Efficient Application Deployment in Fog-enabled Infrastructures

Lyla Naghipour Vijouyeh*, Masoud Sabaei *, José Santos †, Tim Wauters †,
Bruno Volckaert † and Filip De Turck†
*Computer Engineering Department
Amirkabir University of Technology, Tehran, Iran
Email: lyla.naghipour@aut.ac.ir
†IDLab, Department of Information Technology
Ghent University - imec, Ghent, Belgium

***Abstract*—Fog computing is a paradigm that extends cloud computing services to the edge of the network in order to support delay-sensitive Internet of Things (IoT) services. One of the most promising use-cases of fog computing is Smart City scenarios. Fog computing can substantially improve the quality of citywide services by reducing response delays. Owing to geographically distributed and resource-constrained fog nodes and a multitude of IoT devices in Smart Cities, efficient service deployment and end device traffic routing are quite challenging. Therefore, in this paper, we present an Integer Linear Programming (ILP) formulation for the Joint Application Component Placement and Traffic Routing (JAcPTR) problem in which users' delay requirements and the limited traffic processing capacity of application instances are considered. Besides, the JAcPTR enables users and infrastructure managers to easily enforce their locality and management requirements in the deployment of application instances. To cope with the considerably high execution time in large instances of the JAcPTR problem, we propose a fast polynomial-time heuristic to efficiently solve the problem. The performance of the proposed heuristic has been evaluated through extensive simulation. Results show that in large instances of the problem, while the state-of-the-art Mixed Integer Linear Programming (MILP) solver fails to obtain a solution in 50% of the simulation runs in 300 seconds, our proposed heuristic can obtain a near-optimal solution in less than one second.**

***Index Terms*—Fog Computing, Cloud Computing, Smart City, Application Deployment, Traffic Routing.**

I. Introduction

In recent years, the Internet of Things (IoT) has introduced a new set of ubiquitous services by transforming dumb objects into communicating and smart devices [1]–[3]. According to [4], the number of IoT connected devices is expected to grow to 41.6 billion, generating 79.4 zettabytes of data in 2025, which imposes considerable, yet unsolved, challenges in the existing cloud-centric infrastructure [5]. The high end-to-end delays, lack of bandwidth, and network congestion are among the most critical challenges of current cloud-centric IoT systems. Fog computing was introduced to cope with the aforementioned issues [6]. Fog computing extends cloud computing services, including processing, storage, and network services to the network edge to support geographically distributed and delay-sensitive IoT applications [6], [7].

One of the most promising use-cases of fog computing is Smart City scenarios. Fog computing can considerably improve the quality of citywide services and decrease response time while reducing network load [8]. In this regard, we generally consider Smart City applications, and in particular, Antwerp's City of Things (CoT) and its diverse range of applications, including smart traffic light, smart water, and smart freeways [9]–[11]. Generally, the operator of a Smart City faces several challenges in provisioning IoT applications on hybrid cloud-fog infrastructure. On the one hand, fog nodes are resource-constrained and may not be able to host all components of one application or the components of different applications. On the other hand, a large number of end devices may simultaneously request the applications hosted by fog nodes. As a result, a strategy is needed to enable the Smart City operator to properly deploy applications and route end device traffic to the hosted applications.

In this paper, we focus on providing the Smart City operator with the ability to manage infrastructure, end devices, and applications efficiently. In this regard, we present a mathematical formulation for the Joint Application Component Placement and Traffic Routing (JAcPTR) problem with the objective of maximizing the infrastructure provider profit. The JAcPTR deals with a set of application requests with diverse requirements such as maximum tolerable delay and service locality. IoT applications have been designed based on microservice architecture, and thus, each application is considered as a set of communicating services [12]–[14]. In addition, we assume each application has a limited traffic processing capacity, and sometimes, it may be necessary to create and deploy multiple instances of the application. We offer management capabilities in which users can define a set of preferred locations for the deployment of their requested applications. The JAcPTR problem consists of jointly solving: *(1) Application deployment sub-problem:* this determines the required number of application instances, physical placement of application services, and mapping each virtual link between services to a path in the infrastructure; and *(2) Traffic routing sub-problem:* this consists of assigning the traffic of end devices to the deployed application instances, determining the fog node through which end device traffic enters the network, and routing the end device traffic to the assigned application instance. Moreover, we propose a fast polynomial-

time heuristic to solve the problem efficiently.

In summary, the main contributions of this paper are as follows:

- We formulate the JAcPTR problem as an Integer Linear Programming (ILP) problem that can deploy applications in the infrastructure network and route end device traffic to deployed applications while meeting different user requirements and maximizing the profit of the infrastructure provider.
- We propose a fast polynomial-time heuristic to calculate a feasible and sub-optimal solution to the JAcPTR problem.
- We evaluate the performance of the proposed heuristic through extensive simulations. Results demonstrate that in large instances of the problem, our proposed heuristic can obtain a near-optimal solution in less than one second.

The rest of the paper is organized as follows. In Section II, the related works are discussed. The mathematical formulation of the JAcPTR is presented in Section III, and a heuristic algorithm for solving it is proposed in Section IV. Section V presents the evaluation settings and results. Finally, we conclude the paper in Section VI.

II. Related Work

In recent years, various studies have been conducted on resource allocation to IoT applications in fog computing environments, more specifically, service placement and traffic routing. In [15], the authors have investigated the Virtual Machines (VMs) placement problem in Mobile Edge Computing (MEC) architecture to minimize the delay. The authors in [16] have addressed the dynamic service placement problem on edge servers for the group virtual reality games to minimize operational costs while respecting the players' Quality-of-Service (QoS) requirements. In [17], a Mixed Integer Nonlinear Programming (MINLP) model for the joint VMs resource allocation and user request assignment problem has been presented to minimize the average response time. Kherraf et al. [18] have formulated the problem of resource provisioning and workload assignment for IoT service as a Mixed Integer Programming (MIP) problem, to minimize the cost of edge servers and applications deployment while ensuring the IoT services delay requirement. An ILP model has been presented in [19] for the IoT application placement problem with the aim of maximizing profit while the users' QoS is guaranteed.

Mouradian et al. [20] have proposed an ILP formulation for the IoT application component placement problem in Network Function Virtualization (NFV)-based fog-cloud systems aiming to minimize cost and makespan. In [21], the VNF placement and user traffic routing problem has been addressed to minimize delay while respecting the limitation of resource capacity and delay requirements. The problem of application component placement and mapping links between them with the objective of minimizing the maximum costs is studied in [22]. In [23], by assuming the application as a chain of services with limited processing capacity, the authors have presented a multi-objective Integer Nonlinear Programming (INLP) model for the joint service placement and data flow management problem to minimize deployment costs and service delay. Moreover, in [24], a Mixed Integer Linear Programming (MILP) model for the Service Function Chain (SFC) deployment problem in edge servers has been presented to find the optimal placement of services and route traffic between the edge servers while minimizing cost. In [25], the authors assumed that each gateway could have limited associated stations and proposed an ILP model for IoT application resource provisioning with different objective functions while taking the wireless constraint into account.

A comparison of the most important reviewed studies is presented in Table I. We compare them according to the following criteria: (1) decision variables in the defined problems, (2) the objective function of the defined problems, (3) the most crucial assumptions used in the problem definition, and (4) whether delay and locality are considered in users' requests or not.

In this paper, we investigate the JAcPTR problem by considering both delay and locality requirements. In this regard, we take several real-world assumptions, including the limited processing capacity of applications and the limited connection capacity of fog nodes into account. As shown in Table I, our approach is different from previous studies in considering both delay and locality requirements at the same time. In addition, the JAcPTR considers the limited traffic processing capacity of applications as well as end device traffic routing to the deployed instances of the application.

III. System Model and Problem Formulation

We consider the high-level fog-cloud architecture of a Smart City, as illustrated in Fig. 1. The details and specifications of the architecture are based on the CoT project [9], [11]. According to [9], wireless gateways, in addition to communication resources, have processing and memory resources. These gateways serve as fog nodes in the infrastructure. In the assumed architecture, each user prepares and submits a set of application requests to the Smart City provider. The fog and cloud infrastructure provider must deploy the requested applications and route the traffic of end devices to them according to the requirements outlined in the users' request and the available resources in the infrastructure. In the following, first, the infrastructure network model, the user request model, and the problem assumptions are explained. Next, the formulation of the JAcPTR problem is presented. The notations used for variables and parameters in this paper are also listed in Table II and Table III, respectively.

A. Infrastructure Network Model

The hybrid fog-cloud infrastructure is modeled as a directed graph $G = (V, E)$, wherein V is the set of cloud nodes, fog nodes, and switches, and E is the set of links among the infrastructure nodes. We denote the set of cloud nodes, fog nodes, and switches by V^C, V^F, and V^S, respectively, and we define $V = V^C \cup V^F \cup V^S$. A directed link from node n to node m is shown by $(n, m) \in E$ with bandwidth capacity $B_{(n,m)}$ and propagation delay $d_{(n,m)}$. Cloud and fog nodes have limited processing and memory resources represented by

TABLE I: Summary of Related Research

Ref.	Decision Variables					Objective function	Assumptions		User requirements	
	Application deployment		End device traffic assignment to		End device/ user traffic routing		Limited processing capacity of application instances	Limited connection capacity of fog nodes	Delay	Locality
	Service/ application placement	Link mapping	Fog node	Application instance						
[20]	✓			✓		Minimizing costs and time	✓			
[21]	✓				✓	Minimizing delay			✓	
[22]	✓	✓				Minimizing the maximum costs				
[23]	✓	✓				Minimizing cost and delay	✓			
[25]	✓	✓	✓			Seven different objective function		✓		✓
This paper	✓	✓	✓	✓	✓	Maximizing profit	✓	✓	✓	✓

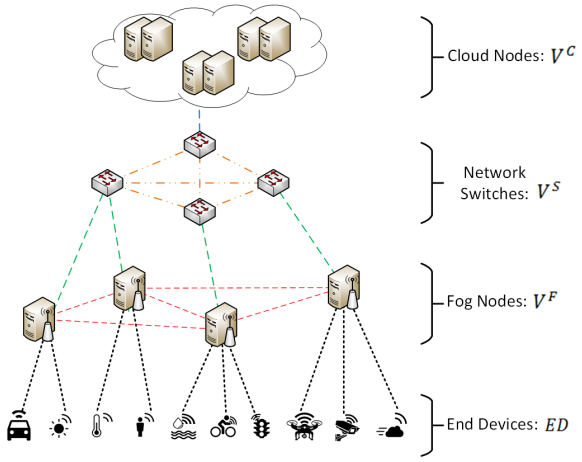


Fig. 1: High-level View of Hybrid Fog-cloud Infrastructure Considered for Smart City.

TABLE II: Notations: Decision Variables

Symbol	Description
$accept^a \in \{0,1\}$	Equals 1 if the request a is accepted
$placed^{a,i} \in \{0,1\}$	Equals 1 if the instance i of the application a is created
$x_n^{a,i,s} \in \{0,1\}$	Equals 1 if service s of the instance i of the application a is placed on the node n
$y_{(n,m)}^{a,i,(s,s')} \in \{0,1\}$	Equals 1 if the virtual link (s,s') of the instance i of the application a uses link (n,m) of the infrastructure
$z_{ed}^{a,i} \in \{0,1\}$	Equals 1 if end device ed is assigned to the instance i of application a
$f_{ed,(n,m)}^{a,i} \in \{0,1\}$	Equals 1 if the link (n,m) is used to route the traffic of end device ed to the instance i of application a
$k_{ed,n} \in \{0,1\}$	Equals 1 if the traffic of end device ed enters to the network through fog node $n \in V^F$

Ω_n and Γ_n, respectively. In addition, fog nodes can serve a limited number of associated end devices depending on the technology they use [25]. We use θ_n, henceforth referred as a fog node $n \in V^F$ connection capacity, to denote the maximum number of end devices that can be served by fog node $n \in V^F$.

B. User request Model

Let A show the set of all application requests received by the infrastructure provider. We assume each request contains all the information required by the infrastructure provider. Each application request $a \in A$ is modeled by $a = \langle G^a, \tau^a, ED^a, R^a, D^a \rangle$ where G^a and τ^a are a graph describing the components of the requested application and the processing capacity of the requested application, respectively. ED^a presents the set of end devices whose traffic is assumed to be processed by the requested application. R^a and D^a represent the set of preferred locations for the services constituting the requested application and the maximum tolerable delay, respectively.

TABLE III: Notations: Parameters

Symbol	Description
$G = (V, E)$	The infrastructure graph
A	The set of all requests
ED	The set of all end devices
$G^a = (V^a, E^a)$	Graph of application a
ED^a	The set of all end devices in the request a
E'	The set of virtual links between end devices and fog nodes
I^a	The maximum number of instances that can be created from the application a
D^a	The maximum tolerable delay for request a
τ^a	The processing capacity of application a
$\gamma^{a,s}$	The memory requirement of service s of application a
$\omega^{a,s}$	The processing requirement of service s of application a
λ_{ed}^a	The traffic rate of end device ed in request a
$profit_{ed}^a$	The profit of end device ed in request a
$\varphi^{a,(s,s')}$	The bandwidth requirement of virtual link (s,s')
$R_n^{a,s} \in \{0,1\}$	Equals to 1 if it is permitted to deploy service s from application a on node n
$B_{(n,m)}$	The bandwidth of link (n,m)
$d_{(n,m)}$	Link (n,m) propagation delay
θ_n	The connection capacity of fog node $n \in V^F$

Since each request contains just one application, we reuse the notation of request a for the application that the request describes. The application a is modeled by a linear directed graph $G^a = (V^a, E^a)$, wherein V^a contains a source, a destination, and a set of services constituting application a, and E^a is the set of virtual links between the services. The source node acts as a classifier determining the services through which the traffic of each end device passes. Therefore, the source node is modeled similarly to other application services. The destination node represents the exit point of the application traffic. The physical location of the destination nodes on the infrastructure are predefined and can be an egress switch of the network. An egress switch is a switch through which traffic leaves the network. A virtual link $(s,s') \in E^a$ indicates a data stream from service s to service s', and its bandwidth requirement is shown by $\varphi^{a,(s,s')}$. The processing

and memory requirements of the service s in the application a are denoted by $\omega^{a,s}$ and $\gamma^{a,s}$, respectively. We assume only one configuration file exists for the deployment of each application.

Let ED be the set of all end devices in all requests. We define a virtual link (ed, n) between each fog node $n \in V^F$ and each end device $ed \in ED$ within the service region of that fog node. E' represents the set of all virtual links between end devices and fog nodes. Each virtual link has a limited bandwidth capacity, shown by $B_{(ed,n)}$, which is determined by communication technology. The propagation delay of each virtual link (ed, n) is indicated by $d_{(ed,n)}$.

C. Assumption

To deploy an application, the infrastructure provider has to deploy all its constituent services and map each virtual link between them to a path in the infrastructure. The infrastructure provider or the user requesting an application can determine whether service s of application a can be placed on the infrastructure nodes $n \in V$ or not by setting parameter $R_n^{a,s} \in \{0, 1\}$. If $R_n^{a,s} = 1$, the service s of application a can be placed on node n. This parameter is used to guarantee user locality requirements and infrastructure provider management capabilities. We assume each application instance to be limited in terms of the amount of traffic it can process. If the input traffic to an application instance exceeds its processing capacity, the infrastructure provider has to create and deploy more instances of the application. By creating a new instance of an application, a new instance of all its services is created. The infrastructure provider has to assign the traffic of end devices to the application instances. To this end, the decision variable $z_{ed}^{a,i}$ is used. If $z_{ed}^{a,i} = 1$ the end device ed is assigned to the instance i of application a.

We assume that there is a fixed revenue and cost for each end device. Therefore, the infrastructure provider can calculate the profit of accepting the traffic of each end device. $profit_{ed}^{a}$ is the profit gained by accepting the traffic of end device ed of request a.

D. Problem Formulation

Given the assumptions mentioned above, the JAcPTR problem is formulated as follows. The objective function is the maximization of the infrastructure provider's profit and is expressed via (1). The profit of the infrastructure provider is calculated as the sum of each end device profit in the accepted requests.

$$max \sum_{a \in A} \sum_{ed \in ED^a} accept^a \times profit_{ed}^{a} \tag{1}$$

Several constraints have to be respected to meet this objective. These constraints can be classified as follows:

Determining the number of required instances of each application: Constraint (2) determines the number of required instances of each application. It also ensures the total traffic assigned to each instance does not exceed the processing capacity of that instance.

$$placed^{a,i} \times \tau^a \geq \sum_{ed \in ED^a} z_{ed}^{a,i} \times \lambda_{ed}^{a} \quad \forall a \in A, i \in I^a \tag{2}$$

Application service placement: Constraint (3) respects that each service is placed on at most one node in the infrastructure. As previously mentioned, each service can only be placed on one of the locations specified for it. If an instance of an application is deployed, all its constituent services also have to be placed, which is ensured by constraint (4).

$$\sum_{n \in V} x_n^{a,i,s} \times R_n^{a,s} \leq 1 \quad \forall a \in A, i \in I^a, s \in V^a \tag{3}$$

$$\sum_{s \in V^a} \sum_{n \in V} x_n^{a,i,s} \times R_n^{a,s} = placed^{a,i} \times |V^a| \quad \forall a \in A, i \in I^a \tag{4}$$

The resource capacity of infrastructure nodes: Given the limited processing and memory resources of cloud and fog nodes, services have to be placed such that their total resource requirements do not exceed the capacity of the nodes hosting them. Constraints (5) and (6) respect the limited memory and processing resources of cloud and fog nodes.

$$\sum_{a \in A} \sum_{i \in I^a} \sum_{s \in V^a} x_n^{a,s,i} \times \gamma^{a,s} \times R_n^{a,s} \leq \Gamma_n \quad \forall n \in V^C \cup V^F \tag{5}$$

$$\sum_{a \in A} \sum_{i \in I^a} \sum_{s \in V^a} x_n^{a,s,i} \times \omega^{a,s} \times R_n^{a,s} \leq \Omega_n \quad \forall n \in V^C \cup V^F \tag{6}$$

Mapping virtual links between services: Constraints (7) and (8) ensure that each virtual link between services is mapped to one path in the infrastructure. Constraints (7) enforces flow conservation for cloud and fog nodes, and Constraints (8) applies flow conservation for switches.

$$\begin{aligned} &\sum_{(n,m) \in E} y_{(n,m)}^{a,i,(s,s')} - \sum_{(m,n) \in E} y_{(m,n)}^{a,i,(s,s')} = \\ &x_n^{a,i,s} \times R_n^{a,s} - x_n^{a,i,s'} \times R_n^{a,s'} \\ &\forall a \in A, i \in I^a, (s, s') \in E^a, n \in V^C \cup V^F \end{aligned} \tag{7}$$

$$\begin{aligned} &\sum_{(n,m) \in E} y_{(n,m)}^{a,i,(s,s')} - \sum_{(m,n) \in E} y_{(m,n)}^{a,i,(s,s')} = 0 \\ &\forall a \in A, i \in I^a, (s, s') \in E^a, n \in V^S \end{aligned} \tag{8}$$

Traffic assignment: Constraint (9) guarantees if a request is accepted, the traffic of all its end devices are accepted.

$$\sum_{ed \in ED^a} \sum_{i \in I^a} z_{ed}^{a,i} = |ED^a| \times accept^a \quad \forall a \in A \tag{9}$$

The traffic of each end device has to be assigned to one of the application instances to be processed. Constraint (10) ensures that each end device is assigned to at most one instance of the requested application. If the assignment of the end device traffic to one of the application instances is not possible, its traffic cannot be accepted. Constraint (11) guarantees that if the end device traffic is assigned to one of the requested application instances, that instance of the application is placed.

$$\sum_{i \in I^a} z_{ed}^{a,i} \leq 1 \quad \forall a \in A, ed \in ED^a \tag{10}$$

$$z_{ed}^{a,i} \leq placed^{a,i} \quad \forall a \in A, i \in I^a, ed \in ED^a \tag{11}$$

Each end devices must be assigned to a fog node to be able to send its traffic. Constraint (12) ensures that if the end device traffic is accepted, it is assigned to a fog node. Constraint (13) respects the limited bandwidth of the virtual link between the end device and the fog node. Constraint (14) assures the number of end devices assigned to a fog node does not exceed its connection capacity.

$$\sum_{n \in V^F} k_{ed,n} = \sum_{a \in A} \sum_{i \in I^a} z_{ed}^{a,i} \quad ed \in ED^a \tag{12}$$

$$\lambda_{ed}^{a} \times k_{ed,n} \leq B_{(ed,n)} \quad \forall a \in A, ed \in ED^a, (ed,n) \in E' \tag{13}$$

$$\sum_{ed \in ED} k_{ed,n} \leq \theta_n \quad \forall n \in V^F \tag{14}$$

Traffic routing: Following the determination of the application instance to which the end device traffic is assigned, the end device traffic has to be routed to the given application instance. Constraints (15) and (16) present the flow conservation for routing the traffic of end devices.

$$\begin{aligned} &\sum_{(n,m) \in E} f_{ed,(n,m)}^{a,i} - \sum_{(m,n) \in E} f_{ed,(m,n)}^{a,i} = 0 \\ &\forall a \in A, i \in I^a, ed \in ED^a, n \in V^S \end{aligned} \tag{15}$$

$$\begin{aligned} &\sum_{(n,m) \in E} f_{ed,(n,m)}^{a,i} - \sum_{(m,n) \in E} f_{ed,(m,n)}^{a,i} = \\ &z_{ed}^{a,i} \times k_{ed,n} - x_n^{a,i,0} \times R_n^{a,0} \times z_{ed}^{a,i} \\ &\forall a \in A, i \in I^a, ed \in ED^a, n \in V^C \cup V^F \end{aligned} \tag{16}$$

The bandwidth capacity of infrastructure links: Constraint (17) guarantees the traffic passing through each infrastructure link does not exceed its capacity.

$$\begin{aligned} &\left(\sum_{a \in A} \sum_{i \in I^a} \sum_{(s,s') \in E^a} y_{(n,m)}^{a,i,(s,s')} \times \varphi^{a,(s,s')} \right) + \\ &\left(\sum_{a \in A} \sum_{i \in I^a} \sum_{ed \in ED^a} f_{ed,(n,m)}^{a,i} \times \lambda_{ed}^{a} \right) \leq B_{(n,m)} \ \ \forall (n,m) \in E \end{aligned} \tag{17}$$

Delay constraint: Constraint (18) respects the delay requirement of the requests. The left side of the constraint is, respectively, the sum of (1) propagation delay between cloud and fog nodes for mapping virtual links between services, (2) propagation delay between cloud and fog nodes for routing the traffic of end devices, and (3) propagation delay between end devices and fog nodes. The right side of the constraint presents the maximum tolerable delay determined by the user request.

$$\begin{aligned} &\left(\sum_{(s,s') \in E^a} \sum_{(n,m) \in E} y_{(n,m)}^{a,i,(s,s')} \times z_{ed}^{a,i} \times d_{(n,m)} \right) + \\ &\left(\sum_{(n,m) \in E} f_{ed,(n,m)}^{a,i} \times d_{(n,m)} \right) + \left(\sum_{n \in V^F} k_{ed,n} \times d_{(ed,n)} \right) \\ &\leq D^a \quad \forall a \in A, ed \in ED^a, i \in I^a \end{aligned} \tag{18}$$

Constraints (16) and (18) contain multiplications of two variables, and linearization techniques must be used to convert the formulation to an ILP problem. These constraints can be easily linearized. However, due to the lack of space, we omit the details here.

IV. Proposed Heuristic

The presented ILP formulation for the JAcPTR problem is complex and computationally intractable. To cope with the considerably high execution time in large instances of the JAcPTR problem, we propose a heuristic named First Nearest Then Farthest (FNTF) that solves the JAcPTR problem faster. The general procedure of the FNTF is illustrated in Alg. 1. First, a weight is assigned to each request according to the Value Factor (VF) function. Three factors are involved in the calculation of the VF: (1) the profit gained from the request, (2) the sum of the bandwidth requirement of all virtual links, and (3) the maximum tolerable delay. We define the VF function of the request a as follows:

$$\begin{aligned} VF(a) = &\left(\frac{\sum_{ed \in ED^a} profit_{ed}^{a}}{MaxP} \right) \times \left(\frac{\sum_{(s,s') \in E^a} \varphi^{a,(s,s')}}{MaxBW} \right) \\ &\times \left(\frac{MaxD}{D^a} \right) \end{aligned} \tag{19}$$

where $MaxP$, $MaxBW$, and $MaxD$ are the maximum profit of requests, the maximum bandwidth capacity of the infrastructure links, and the maximum propagation delay in the infrastructure network, respectively. We use the aforementioned parameters in the VL function to consider profit, delay, and bandwidth in request deployment. Next, the requests are sorted in descending order of VF and are scheduled one by one. The FNTF is a two-phase algorithm: (1) the Nearest Search (NS), and (2) the Farthest Search (FS). Each phase consists of two steps: (1) application service placement and (2) traffic routing. If a request is rejected in the first phase, NS, we move it to the next phase, FS, and attempt to deploy it. A request may be rejected in the placement or routing step in the NS phase. If both steps of one phase are completed successfully, the request is accepted, and the infrastructure resources are updated; otherwise, the request is rejected.

A. Applications Service Placement

In the NS and FS phases, different strategies are utilized for application service placement. The application service placement algorithm for the NS phase is illustrated in Alg. 2. First, the required number of instances for the requested application is calculated in line 2; then instances are deployed one after another. In the deployment of instances, the services are placed one by one, and the links between them are mapped. After placing each service, we update the available infrastructure resources. The source node of an application is deployed according to the following procedure: first, candidate nodes for placing the source node are identified. The candidate nodes are nodes that have adequate resources and placement of the service on them is permitted. Then, we assign a weight to each candidate node based on its available resources and select the node with the highest weight for placement in line 6. Eq. (20) presents the weighting function, where $MaxCPU$

and $MaxMem$ are the maximum available CPU and memory, respectively.

$$NW(n) = \frac{availableCPU}{MaxCPU} + \frac{availableMem}{MaxMem} \tag{20}$$

The procedure for deploying other services is different from the source node, which is described below. Line 13 checks the possibility of placing the service on the same node on which the previous adjacent service is placed. If it is possible, we place the service on the same node. The aim of placing services on the same nodes is to save the bandwidth and reduce the delay. If a service cannot be placed on the node that the previous service is deployed on, we have to select another node. For selecting another node, a path is calculated using the weighted shortest path from the node whereon the previous service is placed to all the candidate nodes for the placement of the current service in lines 20 and 21. In line 22, the node whose path has the lowest cost is selected for service placement. Besides, we use that path to map the virtual link between the two subsequent services. We implement the weighted shortest path using the Dijkstra algorithm. A pruned-graph of the infrastructure network is used as the input to the Dijkstra method. The cost of link (n, m) is a function of its propagation delay and remaining bandwidth, and it is calculated using (21). The reason for placing the service on the node with the lowest cost is to save bandwidth and reduce delay.

$$LC\,(n, m) = \frac{d_{(n,m)}/MaxD}{AvailableBW\,(n, m)/MaxBW} \tag{21}$$

The application service placement algorithm in the FS phase is similar to the placement algorithm in the NS phase. The only difference between these two algorithms is in the selection of another node for service placement in line 22 of Alg. 2. As mentioned before, If a service cannot be placed on the node that the previous service is deployed on, we have to select another node. Although in the NS, we select the node with the lowest path cost, in the FS placement algorithm, we choose the node with the highest path cost. We use the FS to improve the profit by reconsidering the rejected requests. Therefore, unlike the NS, we disperse services in order to use every last remaining resource. Since the FS placement algorithm is similar to the NS placement algorithm, it is not presented here.

B. Traffic Routing

In the traffic routing step, the assignment of end devices to fog nodes and deployed application instances, and end device traffic routing are performed. The traffic routing procedure is depicted in Alg. 3. The end device traffic routing is performed according to the following procedure: (1) First, candidate fog nodes and candidate instances are determined for the end device traffic assignment in lines 3 and 4. The candidate fog nodes are nodes that have adequate communication resources and the end device is in their coverage area. Communication resources refer to the remaining connection capacity of the fog node and the virtual link bandwidth between the fog node and the end device. The candidate instances are the deployed instances of the requested application whose remaining pro-

Algorithm 1 Main procedure of the FNTF

```
1: Input: The set of all requests
2: Requests ← sort the requests in descending order of VF
3: for each req in Requests do
4:     placement ← NSplacement(req)
5:     if (placement ≠ null) then
6:         routing ← routing (req, placement)
7:         if (routing ≠ null) then
8:             accept req and update remaining resources
9:         else
10:            placement ← FSplacement(req)
11:            if (placement ≠ null) then
12:                routing ← routing (req, placement)
13:                if (routing ≠ null) then
14:                    accept req and update remaining resources
15:                else
16:                    reject req
17:            else
18:                reject req
19:    else
20:        placement ← FSplacement(req)
21:        if (placement ≠ null) then
22:            routing ← routing(req, placement)
23:            if (routing ≠ null) then
24:                accept req and update remaining resources
25:            else
26:                reject req
27:        else
28:            reject req
```

Algorithm 2 The NSplacement procedure of the FNTF

```
1: Input: request, Output: placement
2: instances ← calculate the number of required instances
3: for each i in instances do
4:     for each s in instanceServices do
5:         if (s is the source node) then
6:             n ← candidate node with the minimum NW
7:             if (n = null) then
8:                 return null
9:             else
10:                place s on the n and update placement
11:        else
12:            n ← node that hosting the previous service
13:            if (we can place s on the n) then
14:                place s on the n and update placement
15:            else
16:                CN ← candidate nodes
17:                if (CN = null) then
18:                    return null
19:                else
20:                    prune infrastructure graph G
21:                    find a path from n to each node in CN
22:                    n ← node in CN with minimum path cost
23:                    place s on node n and reserve that path
24:                    update placement
25: return placement
```

Algorithm 3 The routing procedure of the FNTF

1: Input: $request$ and $placement$, Output: $routing$
2: **for** each ed in $request$ **do**
3: $CFN \leftarrow$ determine candidate fog nodes
4: $CAI \leftarrow$ determine candidate application instances
5: **if** ($CFN \neq null \ \& \ CAI \neq null$) **then**
6: $\bar{G} \leftarrow$ prune infrastructure graph G
7: **for** each n in CFN **do**
8: **for** each i in CAI **do**
9: $dest \leftarrow$ node hosting the first service of i
10: $p \leftarrow$ Dijkstra ($\bar{G}$, n , $dest$)
11: NIV.add(n,i,p.cost(),p)
12: $r \leftarrow$ select the entry with minimum path cost from NIV
13: update $routing$ based on n,i,p in r
14: **else**
15: return $null$
16: return $routing$

cessing capacity exceeds the end device traffic. (2) After that, the shortest path is calculated from each candidate fog node to all nodes whereon the source node of candidate instances is placed using the Dijkstra method in lines 7-11. The links cost and the Dijkstra method inputs are similar to the placement step. (3) Finally, the path with the lowest cost is selected and reserved for the end device traffic. This path connects two entities: The fog node that carries the traffic of the end device into the network, and the node hosting the source node of an application instance that is responsible for processing the traffic of the end device, i.e., the traffic of the end device is assigned to that application instance.

V. Numerical Results

In this section, we conduct an extensive simulation to evaluate the performance of our proposed heuristic against the CPLEX Branch and Cut algorithm (B&C) [26] and a Random-based Placement and Routing (RPR) method in terms of average execution time and profit. In the RPR, first, the application services are placed randomly on the infrastructure nodes. Then the Dijkstra algorithm is used to find a path between communicating services. The traffic routing part of the RPR is similar to the FNTF routing step. Their only difference is the cost of infrastructure links. In RPR, propagation delay is used as the cost of links.

A. Simulation Settings

The Fog-Cloud infrastructure used in this paper is based on [14], [27]. The topology consists of twelve fog nodes, three cloud nodes, and five switches. The specifications of the infrastructure nodes and links are summarized in Table IV.

We randomly generated a set of requests using the data presented in Table V and Table VI. Each request has 5 to 10 end devices, and the generated traffic of each end device was randomly selected between [0.02-0.2] Mbps. Each application graph contained 3 to 5 services. The specifications of application services were based on Smart City applications studied in [25], [27]. The traffic processing capability of each application was chosen randomly between [1-20] Mbps. The

TABLE IV: Infrastructure Specifications

Parameter	Value
CPU (GHz)	{4,8,12,16}
RAM (GB)	{8,12,16,24,48}
Fog nodes connection capacity	{38,80,100}
Link bandwidth (Mbps)	{25,50,125,150,500}
Links propagation delay (ms)	{1,3,5,15,25}

TABLE V: Requests Parameters

Parameter	Value
Number of end devices	5 to 10
End device traffic rate (Mbps)	Uniform [0.02-0.2]
The maximum tolerable delay (ms)	Limited scenario: Uniform [60-180]
	Exact scenario: Uniform [10-180]

TABLE VI: Services Specifications

Parameter	Value
CPU requirement (GHz)	{0.1,0.2,0.5,1}
Memory requirement (GB)	{0.128,0.256,0.512,1}
Bandwidth (Mbps)	Uniform [2-8]

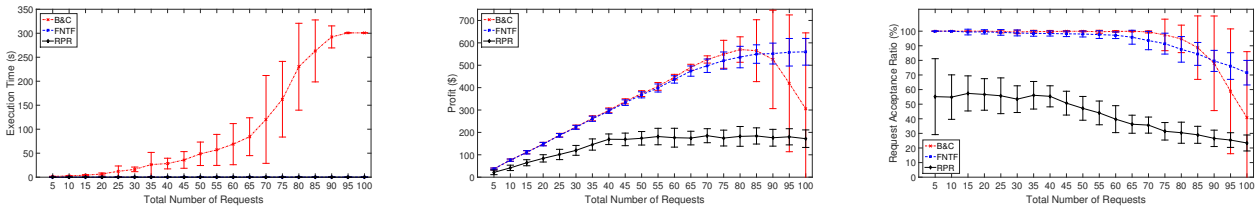
value of the $R_n^{a,s}$ parameter was randomly set to one. The value of the $profit_{ed}^{a}$ parameter was set to one dollar for all end devices. Each end device was in the coverage area of two fog nodes, which were selected randomly. The maximum connection capacity of fog nodes was chosen randomly from the set {38,80,100}. Virtual wireless links bandwidth and delay were random numbers between [0.5-1] Mbps and [0.5-1] ms, respectively.

We used two scenarios to evaluate the performance of the proposed heuristic: Limited and Exact. In the Limited scenario, the maximum tolerable delay of each request is a random number between [60-180] ms. We limited the execution time of B&C to 300s in this scenario. Accordingly, the termination condition of the B&C was reaching 300s or achieving the optimality. In the Exact scenario, we did not limit the execution time and the range of maximum tolerable delay was set to [10-180] ms. All simulations were carried out on a server with 20 GB of memory and four E5-2690 CPU cores operating at 2.60 GHz. All simulations results have been averaged over 30 successive runs.

B. Performance Evaluation

The performance of different methods in the Limited scenario is shown in Fig. 2. As shown in Fig. 2a, by increasing the number of requests from 5 to 100, the execution time of the B&C grows from 1.65s to 300s. In contrast, the execution time of the other two methods is constantly below 1s despite their ascending trend. Therefore, both the FNTF and RPR methods calculate the solution faster than the B&C method.

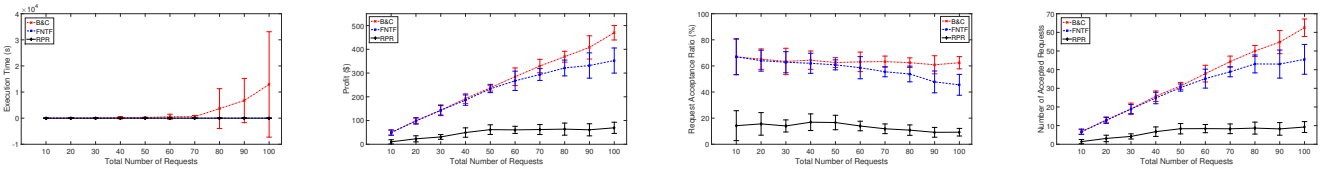
According to Fig. 2b, by increasing the number of requests from 5 to 80, the profit of both the B&C and FNTF methods gradually grows from 37.12$ to 570.064$ and 536.538$, respectively. However, the profit of RPR remains almost constant after reaching 40 requests. As we continue to increase the number of requests, the execution time of B&C reaches the termination limit in some of the simulation runs. Therefore,



(a) Execution Time (b) Objective Value (c) Request Acceptance Ratio

(d) Accepted Requests (e) The Optimality Gap of B&C Method (f) Failed Runs in B&C Method

Fig. 2: The Performance of B&C, FNTF and RPR Methods in the Limited Scenario.



(a) Execution Time (b) Objective Value (c) Request Acceptance Ratio (d) Accepted Requests

Fig. 3: The Performance of B&C, FNTF and RPR Methods in the Exact Scenario.

the B&C algorithm ends before obtaining the optimal solution, which leads to a decrease in profit and the number of accepted requests. Fig. 2f represents the percentage of failed runs in the B&C method. A failed run is defined as a run that B&C failed to accept at least one request of the requests set. The obtained optimality gap in other runs is depicted in Fig. 2e. As shown in Fig. 2f, the percentage of failed runs increases exponentially to 50%; therefore, B&C needs more time to find an acceptable solution. In the Limited scenario, the FNTF outperforms the B&C and RPR by 45.5% and 69.29% more profit in 100 requests, respectively. Also, the FNTF deploys 43% and 67.27% more requests than the B&C and RPR methods in 100 requests, respectively. The B&C fails to obtain acceptable results in limited time due to the increase in the problem size, which shows the importance of developing a faster method.

In Fig. 3, the performance of different methods in the Exact scenario is illustrated. As shown in Fig. 3a, the B&C obtained the optimal solution after spending 12917.58s on solving the problem whit 100 requests. However, the obtained solution is just 24.9% better than the proposed heuristic and accepts only 17% more requests. Therefore, it is justifiable to use the proposed heuristic in large instances of the JAcPTR problem since it can obtain a sub-optimal solution in less than one second while the B&C method cannot solve the problem efficiently.

VI. Summary and Conclusion

In this paper, we investigate the JAcPTR problem that jointly solves the application component placement and traffic routing problems. Each IoT application was considered as a set of communicating services. We assumed each application instance to be limited in terms of the amount of traffic it can process. Therefore, if the amount of entering traffic is larger than an application instance processing capacity, it is necessary to create multiple instances of the application. The JAcPTR took management capabilities into account that capable users to define a set of preferred locations for the deployment of their requested application. We proposed an ILP model for the JAcPTR problem with the objective of maximizing the fog-cloud infrastructure provider's profit while respecting the users' requirements and infrastructure resource limitations. We also proposed a fast polynomial-time heuristic named FNTF to solve the problem efficiently. We evaluated the FNTF performance through simulation. According to the simulation results, by considering a time limit of 300 seconds, the B&C cannot obtain a solution in 50% of the simulation runs with 100 requests. However, The FNTF can provide a near-optimal solution in less than one second. Therefore, it is justifiable to use the proposed heuristic in large instances of the JAcPTR problem if execution time matters.

REFERENCES

[1] B. Negash, A. M. Rahmani, P. Liljeberg, and A. Jantsch, "Fog Computing Fundamentals in the Internet-of-Things," in *Fog Computing in the Internet of Things: Intelligence at the Edge*. Springer International Publishing, 2018, ch. 1, pp. 3–13.

[2] J. Yao and N. Ansari, "QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 167–175, Mar. 2019.

[3] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Fog Computing: Enabling the Management and Orchestration of Smart City Applications in 5G Networks," *Entropy*, vol. 20, no. 1, p. 4, Jan. 2018.

[4] "The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast." [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS45213219

[5] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Efficient resource management for virtual desktop cloud computing," *The Journal of Supercomputing*, vol. 62, no. 2, pp. 741–767, Nov. 2012.

[6] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 2018.

[7] M. Mukherjee, L. Shu, and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 2018.

[8] A. Chowdhery, M. Levorato, I. Burago, and S. Baidya, "Urban IoT Edge Analytics," in *Fog Computing in the Internet of Things: Intelligence at the Edge*. Springer International Publishing, 2018, ch. 6, pp. 101–120.

[9] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for IoT smart city experiments," in *Proc. IEEE ISC2*, Sep. 2016, pp. 1–8.

[10] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications," in *Proc. IEEE NetSoft*, Jun. 2019, pp. 351–359.

[11] J. Santos, T. Vanhove, M. Sebrechts, T. Dupont, W. Kerckhove, B. Braem, G. V. Seghbroeck, T. Wauters, P. Leroux, S. Latre, B. Volckaert, and F. D. Turck, "City of Things: Enabling Resource Provisioning in Smart Cities," *IEEE Commun. Mag.*, vol. 56, no. 7, pp. 177–183, Jul. 2018.

[12] F. Chiti, R. Fantacci, F. Paganelli, and B. Picano, "Virtual Functions Placement With Time Constraints in Fog Computing: A Matching Theory Perspective," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 3, pp. 980–989, Sep. 2019.

[13] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[14] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Resource Provisioning in Fog Computing: From Theory to Practice †," *Sensors*, vol. 19, no. 10, p. 2238, Jan. 2019.

[15] L. Zhao and J. Liu, "Optimal Placement of Virtual Machines for Supporting Multiple Applications in Mobile Edge Networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, Jul. 2018.

[16] Y. Zhang, L. Jiao, J. Yan, and X. Lin, "Dynamic Service Placement for Virtual Reality Group Gaming on Mobile Edge Cloudlets," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1881–1897, Aug. 2019.

[17] Q. Fan and N. Ansari, "Application Aware Workload Allocation for Edge Computing-Based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[18] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized Provisioning of Edge Computing Resources With Heterogeneous Workload in IoT Networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 459–474, Jun. 2019.

[19] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated Fog–Cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 177–190, Jan. 2020.

[20] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1130–1143, May 2019.

[21] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.

[22] S. Wang, M. Zafer, and K. K. Leung, "Online Placement of Multi-Component Applications in Edge Computing Environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[23] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An Ant Colony Optimization-Based Multiobjective Service Replicas Placement Strategy for Fog Computing," *IEEE Trans. Cybern.*, pp. 1–14, 2020.

[24] Z. Zhou, Q. Wu, and X. Chen, "Online Orchestration of Cross-Edge Service Function Chaining for Cost-Efficient Edge Computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.

[25] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Resource provisioning for IoT application services in smart cities," in *Proc. IEEE CNSM*, Nov. 2017, pp. 1–9.

[26] "CPLEX Optimization Studio V12.8," Jan. 2018. [Online]. Available: https://www.ibm.com/support/pages/cplex-optimization-studio-v128

[27] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Towards delay-aware container-based Service Function Chaining in Fog Computing," in *Proc. IEEE/IFIP NOMS*, Nov. 2020, pp. 1–9.