Optimization of Function Chaining on the Edge for IoT applications

Mohamed Khaledi, Makhlouf Hadji, Salah Eddine Elayoubi, Dusit Niyato

Optimization of Function Chaining on the Edge for IoT applications

Mohamed-Idriss Khaledi *, Makhlouf Hadji *, Salah Eddine El-Ayoubi † and Dusit Niyato ‡
*IRT - SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France
Email: {mohamed-idriss.khaledi@irt-systemx.fr, makhlouf.hadji@irt-systemx.fr, }@irt-systemx.fr
†Université Paris Saclay, L2S, UMR CNRS 8506, CentraleSupélec, Gif-Sur-Yvette,France,91190
Email: salaheddine.elayoubi@centralesupelec.fr
‡Fellow IEEE
Email: DNIYATO@ntu.edu.sg

***Abstract*—With the rapid deployment of Internet of Things (IoT) applications, video processing and streaming requirements are increasing, and edge computing is a good candidate to cope with strong latency and throughput expectations. In this paper, we consider the optimal routing, placement and scaling of IoT-based service function chains for object detection. We propose an edge networking approach dealing with limited CPU and network bandwidth resources in a joint optimization based on Integer Linear Programming for small problem instances, and a graph-based approximation to cope with scalability issues. We evaluate the efficiency of our algorithms through simulations and show that the graph-based approach converges towards a near-optimal solution in negligible time and is thus suitable for real-time function chain placement.**

***Index Terms*—IoT function chaining, Combinatorial Optimization, Edge Networking**

I. Introduction

With the rapid deployment of smart city and Industrial Internet of Things (IIoT) applications, video streaming and processing is a key issue as many sites are equipped with tens of cameras collecting images and videos for different objectives such as surveillance, security, cyber physical threats, etc. In this context, stakeholders require stringent requirements that include high quality of video streaming with strong latency expectations to rapidly react when an incident such as a physical intrusion. Moreover, Industrial players require efficient deployment solutions for interconnected cameras and video analysis systems. Virtualization techniques are being privileged for this purpose, where both the networking functions are virtualized, using Virtualized Networks Functions (VNF) technology, and application functions are flexibly developed for being deployed on edge nodes.

In order for these functions to be deployed on an infrastructure composed of a multitude of low capacity nodes, they are often presented as a chain of functions, each of them being able to be deployed on an independent physical node, while communicating with the adjacent functions on the chain. The chain has to be deployed on the edge network according to physical infrastructures' available resources in terms of CPU (on servers) and bandwidth (on links). The orchestration of these network services necessitate new algorithms to address the placement, routing and scaling.

In our work, we consider IoT service chains composed by four network functions for object detection. For an object detection application for instance, those functions include video generation, object detection, video compression, object recognition and database update. These IoT applications are deployed on nodes equipped with limited processing capabilities, and interconnected with different communications techniques such as Ethernet, Bluetooth, Wifi, Lora, etc. This heterogeneity of the processing capabilities and communications solutions makes the orchestration and management of the deployed network services harder. We need to investigate efficient optimizations of these resources in near real time to attend systems with good performance (convergence time, solution's cost, etc.).

To deal with these research challenges, we propose two mathematical formulations. One is based on an exact modelling invoking Integer Linear Programming techniques. The second approach is based on graph theory to rapidly attend good solutions in negligible times. Our contribution in this paper is given as follows:

- An exact approach based on integer linear programming approach : this method is based on the description of the convex hull of the addressed joint CPU and network optimization problem for IoT video streaming and processing.
- To cope with scalability issues, we propose an approximation approach based on graph modelling. Indeed, we represent IoT chains by trees and physical substrates by undirected graphs with a certain density in terms of number of edges. Hence, our approximation algorithm consists in finding a good mapping of trees on an undirected graph according to different metrics. This will be detailed in next sections.

Note that the novelty of our approaches will be clearly addressed in next sections and their efficiency will be highlighted and compared to the state of the art addressing close research challenges.

The remainder of this paper is organized as follows: next section (Section II) addresses the related work and most

close papers in terms of research challenges in NFV and IoT domains. Sections III contains the problem description and details, problem complexity issue and the two proposed solutions based on ILP and Graph theory. Our approaches will be assessed using simulations in Section IV. We conclude the paper in Section V, with some future research challenges that have the merit to be addressed in a close future.

II. Related work

Paper [1] addressed energy efficiency of the Internet of Things, as many sensors are expected to be completely standalone and able to run for years without battery replacement. The authors proposed a formulation of an optimization problem to jointly design the source coding and transmission strategies for time-varying channels and sources, when considering two criteria based on extending the network lifetime and granting low distortion levels. Authors of this paper applied their offline solution on a IoT network with a dynamic Time Division Multiple Access (TDMA).

Reference [2] is one of the closest work to our paper. It addresses a joint optimization of power consumption of the CPU and the Wireless Network Interface Card (WNIC) of mobile devices while streaming high quality videos. We recall that the two major energy consuming components in mobile video streaming services are the CPU (that supports video data decoding) and the network interface (that supports data communication). Hence, authors of this paper proposed a joint optimization scheme for improved energy efficiency supporting mobile video streaming services. This is based on the adjustment of the number of video chunks to be downloaded and decoded in each packet. However, they only considered joint optimization of CPU and energy consumption for one mobile device. In our work, we consider a joint optimization of CPU and bandwidth in an IoT network with multiple physical objects and where the number of paths between each couple of nodes can be exponential.

Authors of paper [3] discussed the problem of real time video analytics in an edge computing environment. Recall that real-time video analysis is used nowadays in several domains (traffic control, surveillance and security, retail store monitoring) and because of the high data volumes, compute demands and latency requirements, cameras represent the most challenging of things in IoT, and large-scale video analytics may well represent the killer application for edge computing. The authors proposed a real-time video analytics system with low resource cost to produce outputs with high accuracy. The proposed approach can be used in several application such as self-driving and smart cars, etc. Nevertheless, the proposed approach do not consider a joint optimization for different resources such as CPU and bandwidth when satisfying latency expectations.

Paper [4] proposes a mathematical optimization approach over virtual network services for better scaling, placement and routing on a physical substrate. Network services (video streaming, online gaming) are placed and deployed in the network based on fixed predefined descriptors. With the large number of degrees of freedom for finding the best adaptation, deciding scaling, placement, and routing can result in sub-optimal decisions for the network and for the running services. They proposed JASPER (Joint optimization of scaling, placement and routing) in which, each network service is described by a service template containing information on the component network services, their interconnection, and resources requirements. Hence, the solutions of reference [4] are applied using IETF use-cases [5] and examples for an embedding of virtual networks on physical substrates. In our work, we consider limited IoT nodes in terms of available resources, which makes the problem harder and finding feasible solutions more complicated.

Another reference dealing with CPU and bandwidth optimization is given by [6]. Authors of this paper proposed a technique to manage computation offloading in a local IoT network under bandwidth constraints. They proposed approaches to separately optimizing CPU and Bandwidth resources, which can lead to sub-optimal solutions of the problem. We propose a joint optimization of the two mentioned limited resources and discuss the scalability of our approaches for large instances.

In the context of VNF orchestration, the Stratos project [7] proposed a detailed architecture to orchestrate VNFs outsourced to a remote cloud taking considering traffic engineering, VNFs scaling, etc. Nevertheless, this project has addressed only VNF placement solution based only on workloads, and then, chaining constraints are not considered.

References [8] and [9] propose mathematical models for the VNF chains placement with routing constraints. The proposed models, however, describe a limited number of linear constraints and can only characterize a small portion of the problem convex hull. The proposed exact solutions do not scale for large problem instances. We need deeper modeling that can characterize better (completely) the convex hull of the VNF placement and chaining problem to find near optimal solutions in few seconds and scale with problem size. We propose a competitive graph theory approach with the desired properties of converging quickly to near optimal solutions even for large problem instances.

III. System Model and Problem Formulation

A. System description

We consider a set of sensors that collect data for monitoring and analytics purposes. Each sensor is connected to another one via a link that runs on a wired or wireless technology such as Wifi, Ethernet, Bluetooth, etc. This leads to considering different bandwidth capacities and delay profiles on the links related to each couple of sensors. Moreover, we consider limited processing resources (in our case, represented by CPU cores) at each connected sensor. The montoring and analytics service is composed of a chain of n network functions (denoted by $f_1 \rightarrow f_2 \rightarrow ... \rightarrow f_n$) equivalent to Service Function Chains as described by [10].

Figure 1 illustrates the different components of our system model for a specific service of object detection via surveillance cameras in a restricted area. In this example, each camera

captures videos and images and may be equipped by a processing node. Cameras are interconnected and the edge network also incorporates processing servers. Depending on the orchestrator's decision, the videos captured by each of the cameras can be processed locally or send to another processing node. Moreover, the object detection service is composed of a chain of four virtual functions as follows:

1) Function f_1 (**Object/Face Detection**): detects and extracts in frames all faces in the restricted area (industrial site, for instance) before sending the data to f_2
2) Function f_2 (**Object/Face compression**): It characterizes each face by limited number of features
3) Function f_3 (**Object/Face recognition**): the features extracted for each object are compared to a given database
4) Function f_4 (**Database enhancement**): Once an object/Face is recognized with a high reliability, the new corresponding features are added to the database to improve the quality of future identification

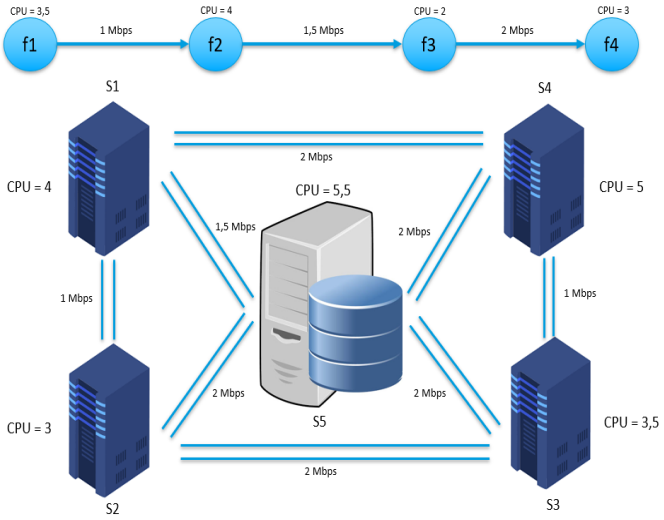

Fig. 1. IoT based virtual network functions mapping for object detection

B. Problem description

The objective of this paper is to find an optimal embedding of the virtualized IoT Service Function Chains (IoT-Chains) on the edge networks subject to processing and bandwidth constraints.

Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be the graph representing the IoT-substrate (interconnected sensors and processing nodes), where $|\mathscr{V}|$ and $|\mathscr{E}|$ represent the network sizes in terms of number of nodes and edges, respectively. Let $\mathscr{G}_v = (\mathscr{V}_v, \mathscr{E}_v)$ be the virtual request graph composed by n chained VNFs. Note that in the particular example of Figure 1, $|\mathscr{V}_v| = 4$ and $|\mathscr{E}_v| = 3$.

Each physical object (IoT) j has a limited CPU processing noted by C_j. Each physical edge (or link) (j, j') has a limited amount of bandwidth noted by $B_{j,j'}$. If there is no edge between j and j', then the amount of available bandwidth between these two nodes is represented by the smallest available bandwidth on the path between j and j'. Each virtual arc (i, i') has a request of $b_{ii'}$ of necessary bandwidth.

We aim at developing new algorithms to reach optimal trade-offs between the two criteria of processing and bandwidth with respect to latency requirements. Before proposing these algorithms, we investigate the complexity of the problem.

C. Problem complexity

The addressed problem in this paper is close to the well known Virtual Network Embedding problem (VNE) (see [11], and [12] for more details), when the chaining and routing constraints are completely relaxed in the requested graphs. Different research papers have already addressed the question of VNE problem's complexity. For instance, reference [13] discussed the general cases of the problem's NP-Hardness, while reference [14] has shown a special case where the requested graphs are represented as cycles. Hence, in this case, the VNE problem can be similar to the 2-Factor (see [15] for more details) problem which is polynomially solvable. This particular case imposes strong hypothesis and assumptions on the virtual graph. Moreover, a recent paper (see [16]) proved the NP-Completeness (using linear reductions from the well known NP-Complete 3-SAT problem) of the VNE problem in almost of the considered virtual and physical graphs architectures. Hence, relaxing routing constraints of our problem is leading to (in general) an NP-Hard VNE problem, which is hampering solving our problem in polynomial time unless $\mathcal{P} = \mathcal{NP}$.

D. Integer Linear Programming approach

Let $\min_{i,j,j',k}$ be the minimum available bandwidth on the k^{th} path between physical nodes j and j' for the virtual arc $(i, i+1)$. Moreover, and for the sake of clarity, we introduce $\mathscr{P}(j, i)$ as the set of all physical nodes j' such that there exists a shortest path (a simple Dijkstra algorithm can be used) between nodes j and j' in which the minimum available bandwidth (on its arcs) is at least equal to the required amount of bandwidth.

Before going through the details of our mathematical formulation describing the Integer Linear Programming (ILP) of our problem, we propose to describe decision variables that will be used in our joint optimization formulation.

- $x_{i,j}$ is a binary variable equals to 1 if the virtual node/function f_i (of the requested chain) is deployed on a physical node j, and equals to 0 otherwise.
- $y_{(i,i+1);(j,j',k)}$ is a binary variable, the value of which is 1 if a virtual arc $(i, i+1)$ for a link between two virtual functions of the demand is placed on a physical path k joining two physical nodes j and j', and 0 otherwise.
- z_j is a binary variable, the value of which is 1 if the physical node j hosts at least one virtual function, and 0 otherwise.

Our objective function is composed by three terms and consists of jointly optimizing the placement and routing of

the four chained virtual functions when minimizing the total CPU core consumption and number of used objects. This is given by:

$$\max \quad \mathscr{F} =$$

$$\sum_{j \in \mathscr{V}} (C_j z_j - \sum_{i \in \mathscr{V}_v} c_i x_{i,j}) + \tag{1}$$

$$\sum_{i \in V_v} \sum_{j \in \mathscr{V}} \sum_{j' \in \mathscr{P}} \sum_{k \in \mathscr{K}} (\min_{ijj'k} - b_{(i,i+1)}) y_{(i,i+1)(j,j',k)}$$

where $\mathscr{K}(j,j')$ is the set of all available and **feasible** paths between nodes j and j'.

Our optimization problem for the placement and routing of virtual chains has a set of constraints described in the sequel:

$$\sum_{j \in \mathscr{V}} x_{i,j} = 1, \forall i \in \mathscr{V}_v \tag{2}$$

Constraints (2) guarantee that each virtual node/function function f_i, is deployed on exactly one physical node/IoT object.

$$\sum_{i \in \mathscr{V}_v} c_i x_{i,j} \leq C_j z_j, \forall j \in \mathscr{V} \tag{3}$$

Constraints (3) ensure that the placement of virtual functions cannot consume more resources than that available on the selected physical node/IoT object.

$$\sum_{i \in \mathscr{V}_v} x_{i,j} \geq z_j, \forall j \in \mathscr{V} \tag{4}$$

Constraints (4) ensure that only selected servers will be taken into account in the first term of the objective function.

$$x_{i,j} \leq \sum_{j' \in \mathscr{P}(j,i)} x_{i+1,j'}, \forall i \in \{1,...,n-1\}, \forall j \in \mathscr{V} \tag{5}$$

Constraints (5) are guaranteeing the chaining of the n virtualized functions f_1 to f_n.

$$\sum_{j \in \mathscr{P}(j',i)} \sum_{k \in \mathscr{K}(j,j',i)} y_{(i,i+1)(j,j',k)} = x_{i+1,j'}, \tag{6}$$

$$\forall i \in \{1,...,n-1\}, \forall j' \in \mathscr{V}$$

$$\sum_{j' \in \mathscr{P}(j,i)} \sum_{k \in \mathscr{K}(j,j',i)} y_{(i,i+1)(j,j',k)} = x_{i,j}, \tag{7}$$

$$\forall i \in \{1,...,n-1\}, \forall j \in \mathscr{V}$$

Constraints (6) and (7) are provided to ensure that if a virtual node/function i is deployed on a physical node j, i.e. $x_{i,j} = 1$, and the virtual node $i+1$ is hosted by a physical node j', i.e. $x_{i+1,j'} = 1$, then the virtual arc $(i, i+1)$ should be deployed on the k^{th} **physical** path starting from node j to node j', i.e. $y_{(i,i+1);(j,j',k)} = 1$.

TABLE I
MATHEMATICAL FORMULATIONS' PARAMETERS

Parameters	***Definition***
x_{ij}	binary variables indicating if VNF_i is hosted in j
$y_{i,i';j,j'}$	a binary variable indicating if the virtual arc (i,i') is hosted on the physical path between j, j'
z_j	a binary variable indicating if node j is solicited
c_i	the requested CPU amount by VNF_i
C_j	The available CPU amount in node j
$K_{(j,j',i)}$	a set of all available and feasible (a path with the minimum bandwidth on all its edges is higher than the requested bandwidth for the arc $(i, i+1)$) paths between nodes j and j' for the virtual arc $(i, i+1)$
$min_{i,j,j',k}$	Minimum available bandwidth on the k^{th} feasible path between physical nodes j and j' for the virtual arc $(i, i+1)$

$$\sum_{j \in \mathscr{V}} \sum_{j' \in \mathscr{P}(j,i)} \sum_{k \in \mathscr{K}(j,j',i)} y_{(i,i+1)(j,j',k)} = 1, \forall i \in \{1,...,n-1\} \tag{8}$$

Constraints (8) guarantee that each virtual arc $(i, i+1)$ is deployed on exactly one physical path. There is no flow split considered in our formulation.

$$\sum_{i \in \{1,2,3\}} \left(y_{(i,i+1)(j,j',k)} + y_{(i,i+1)(j',j,k)} \right) b_{(i,i+1)} \leq \min_{ijj'k}$$
$$\forall i \in \{1,...,n-1\}, \forall j \in \mathscr{V}, \forall j' \in \mathscr{P}(j,l), \forall k \in \mathscr{K}(j,j',i) \tag{9}$$

Constraints (9) are provided to guarantee a minimum bandwidth availability to satisfy the requested routing between each couple of selected physical nodes j and j'.

Table I summarizes the whole of parameters and variables used in our mathematical formulation.

E. Graph-based approach

ILP approach suffers from scalability issues and then cannot address large problem instances. Hence, to cope with this issue, we propose a scalable method based on graph theory. This approach is based on a weighted extended 4-level graph. At each level, we represent a set of physical nodes/IoTs able to host (in terms of available CPU cores) some virtual functions f. There is an arc between a node S_j^l (a physical node j at the level $l (l \leq 4)$) and $S_{j'}^{l+1}$ (a physical node j' at the level $l+1$) with a weight $w_{j,j'}$ representing the available amount of bandwidth between servers j and j'. Figure 2 is built according to the example provided by Figure 1. This provides more details on the construction of the extended multi-level graph.

The constructed multi level graph of Figure 2 allows easily to select physical servers to host virtualized functions when guaranteeing (thanks to the multi-level property) the routing

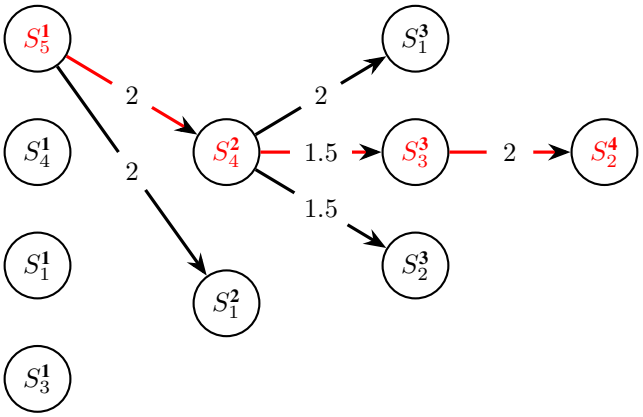

Fig. 2. Extended multi-level graph construction

expected in the virtual graph request.

This approach converges rapidly to near optimal solutions with a complexity of $O(|\mathscr{V}|^4)$ and this will be illustrated in next section.

As it is illustrated in Figure 2, the first level of the extended graph, is used to select a server (according to a given criteria such as high available CPU resources) to host f_1. Before identifying the second server (in level 2 of the extended graph) to host f_2 using the same criteria, we compute a shortest path, in terms of available bandwidth, between the selected server in the previous level, and all of the available servers (in terms of available CPU) of level 2. We iterate these operations for levels 3 and 4 of the extended graph.

Note that the selected strategies using servers with high available CPU and shortest paths to guarantee a bandwidth efficient routing, can be replaced by other techniques to attend different objectives.

In the following, we provide the pseudo-code (see Algorithm 1) of the graph based heuristic and discuss its performance and convergence through simulations.

IV. Numerical Results

Our proposed approaches are evaluated through a Python implementation using CPLEX [17] solver for the exact approach. We use a laptop with 8Gb of RAM and 2.7Ghz of CPU. The following metrics are used to quantify the efficiency of the two proposed approaches (exact and heuristic):

1) **Gap** between the objective functions of the proposed approaches: this metric is provided to measure the efficiency of the approximation algorithm benchmarked by the exact method. It is given as follows:

$$Gap(\%) = \frac{Exac.Sol - Heur.Sol}{Exac.Sol} \times 100$$

2) **Convergence time**: The necessary time to converge to a (near) optimal solution
3) **Scalability**: The ability of the algorithm to scale to a large graph and provide a solution within an acceptable convergence time.

Algorithm 1: Graph-based approach

$SelectedNodes \leftarrow []$;
$ObjF \leftarrow 0$;
$i \leftarrow 1$;
Get the list of physical nodes that can host the first VNF f_1;
$S_j^1 \leftarrow$ Best node to select (in terms of CPU) that can host f_1;
while $i \leq 3$ **do**
  Subtraction of CPU of c_i from node S_j^i;
  if $i \geq 2$ **then**
    Subtraction of $b_{i-1,i}$ from the selected physical path joining S_j^{i-1} to $S_{j'}^i$;
  end
  Get the list of nodes that can host f_{i+1};
  if *(S_j^i has connections with other nodes $S_{j'}^{i+1}$ that can host f_{i+1})* **then**
    Append S_j^i to $SelectedNodes$;
    if $i \geq 2$ **then**
      $ObjF \leftarrow ObjF + min_{S_j^i, S_{j'}^{i+1}, k} - b_{i-1,i}$;
    end
    Get $min_{S_j^i, S_{j'}^{i+1}, k}$ (minimum bandwidth on the best paths that connect S_j^i to each of the nodes $S_{j'}^{i+1}$ in the next layer $(i+1)$);
    Sorting the list of nodes reached by S_j^i and that can host f_{i+1} in a descending order in terms of CPU and Bandwidth optimization;
    $S_{j'}^{i+1} \leftarrow$ Best node to select in layer $(i+1)$;
    Increment i by 1;
  else
    while *there are still nodes that can host f_i* **do**
      Update the CPU and Bandwidth;
      $S_j^i \leftarrow$ The new selected node that can host f_i;
      Re-update the CPU and Bandwidth;
      Check if this node has connections with other nodes that can host f_{i+1} (if True then **break** the loop);
    end
    if *it doesn't exist any node* **then**
      return The Network is overloaded;
    else
      Append S_j^i to $SelectedNodes$;
      if $i \geq 2$ **then**
        $ObjF \leftarrow ObjF + min_{S_j^i, S_{j'}^{i+1}, k} - b_{i-1,i}$;
      end
      Get $min_{S_j^i, S_{j'}^{i+1}, k}$;
      Sorting the list of nodes reached by S_j^i and that can host f_{i+1} in a descending order in terms of CPU (verifying Bandwidth feasibility) optimization;
      $S_{j'}^{i+1} \leftarrow$ Best node to select in layer $(i+1)$;
      Increment i by 1;
    end
  end
end
$ObjF \leftarrow ObjF + min_{S_j^3, S_{j'}^4, k} - b_{3,4}$;
Update (CPU and Bandwidth) for the 4^{th} layer;
Append S_j^4 to $SelectedNodes$;
$ObjF \leftarrow ObjF +$ the remaining CPU of the selected nodes;
return $SelectedNodes, ObjF$;

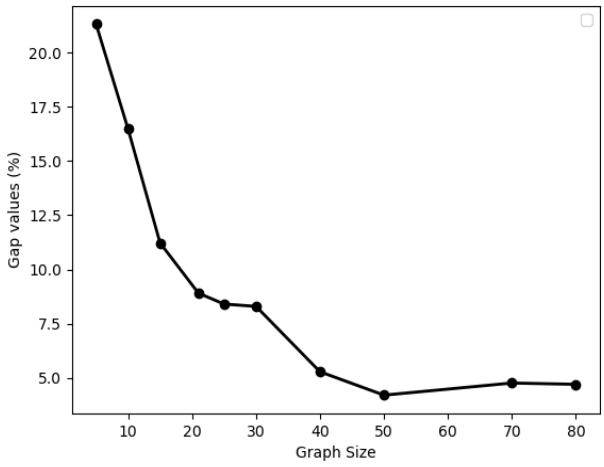

Fig. 3. Evolution of the gap (objective functions) with graph sizes

4) **Rejection rate**: This metric represents the capacity of the system in terms of a number of requests that can be processed without creating instability.

The assessment scenarios correspond to physical infrastructures (graphs) with a number of nodes. We develop two flavors of simulations: a static version that implements the optimization algorithm on a graph for one function chain, and a dynamic version where requests for function chain placement arrive according to a Poisson process. The latter version is used to assess the capacity of the system, in terms of low rejection rate.

A. Placement of a single function chain on an edge graph

For each scenario in our simulations, we represent each point by the average value of 50 runs of different collected metrics of the two proposed approaches.

Table II depicts the obtained results when comparing the graph-based approach to the exact solution using convergence time and the gap between the two obtained solutions. In these simulations, we consider graphs with number of nodes in the $[5; 80]$ range. As expected, the exact algorithm outperforms the graph based approach in terms of objective function metric (see columns 3 and 4 in Table II). Nevertheless, the graph based approach converges to near optimal solutions. This is confirmed by the Gap values (see column 7 in Table II) which are decreasing when graph sizes are increasing. Figure 3 depicts the evolution of the gap metric (in terms of CPU/Bandwidth) with the considered graph sizes. The figures illustrate a weak gap value for the two scenarios which indicates and confirms the efficiency of the proposed graph-based method to reach near-optimal solutions. The gap values are decreasing and are vanishing to 5% when considering graphs or IoT substrates of 50 nodes.

Moreover, columns 5 and 6 of Table II show the execution times for the exact and heuristic approaches, respectively. Exponential time execution for the exact approach can be observed, reaching more than 900 ms for small graphs (30 nodes) and 91 seconds for large ones (80 nodes). The graph based approach has negligible time convergence even for larger graphs. Indeed, the exact approach is time consuming especially when investigating an explosive number of paths to guarantee the requested routing of virtual functions. The graph-based approach uses the already described multi-level extended graph, to easily and rapidly detect a feasible/optimal path for routing strategies.

TABLE II
EXACT VS APPROXIMATION ALGORITHMS PERFORMANCE

# nodes	# links	Exa. Obj	Heu.Obj	Exa.Time(ms)	Heu.Time(ms)	Gap%
5	7	14.58	11.47	14.72	0.24	21.33
10	16	19.65	16.41	72.64	1.56	16.48
15	25	21.49	19.08	171.72	2.75	11.22
21	36	23.16	21.10	418.86	9.43	8.90
25	43	23.92	21.91	632.81	11.34	8.40
30	52	24.17	22.17	937.66	17.33	8.29
40	70	25.27	23.94	14607.15	23.56	5.27
50	88	26.28	25.18	26110.11	30.78	4.19
70	124	26.47	25.21	58235.02	45.78	4.75
80	142	26.52	25.28	91514.11	53.32	4.69
150	268	24.89	23.85	308015.30	83.41	4.18

To highlight the scalability of the heuristic based approach, we provide in the last row of Table II the results for a graph with 150 nodes and 268 edges, randomly generated. The heuristic approach converges to near optimal solutions in less than 84 ms for the worst case, when the exact approach necessitates more than 25 minutes to attend the optimal solution.

B. Dynamic placement of function chains

In the following, we address simulations with dynamic setting where demands for service function chain placement arrive according to a Poisson process with inter-arrival times (noted by λ^{-1}) and departures are represented by exponential distribution with a parameter (lifetimes) represented by μ^{-1}, called also service rate. This dynamic setting allows to compute the system capacity, in terms of rejection rate. Hence, we introduce $\rho = \frac{\lambda}{\mu}$ to assess these performances. Note that the event of request rejection occurs when a new request arrives while the system his still processing a large number of previous requests. This is modeled in our case by a buffer for requests of limited size (10 in the numerical results). When this buffer is full, new arriving requests are rejected. This rejection is thus related to the algorithm's necessary convergence time: The faster the convergence, the lowest is the probability of buffer overflow.

Figure 4 reports the rejection rate evolution for different values of system load. We selected graphs with 15, 30 and 50 nodes for this simulation. As the convergence time for the exact (ILP) approach increases exponentially with the graph size, the rejection rate is non negligeable for a larger graph size. When the load increases (i.e. the average number of new requests increases), the rejection rate also increases. Note that for the graph-based approach, the rejection rate is close to zero. This is due to the negligible necessary time to converge to near optimal solution using the heuristic approach.

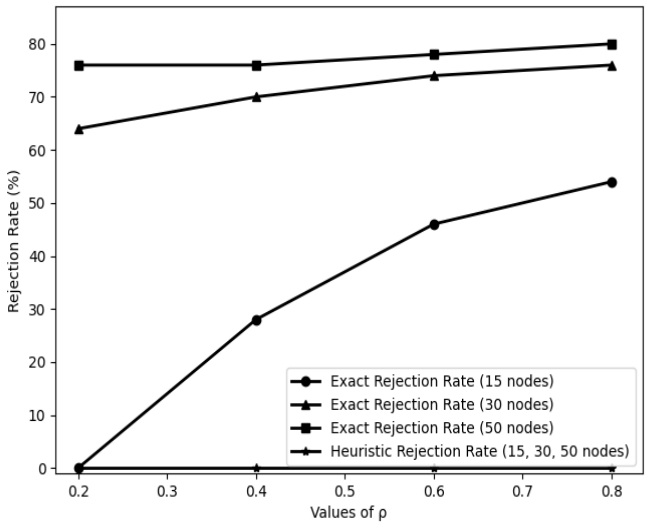

Fig. 4. Exact algorithm's rejection rate for different graph sizes

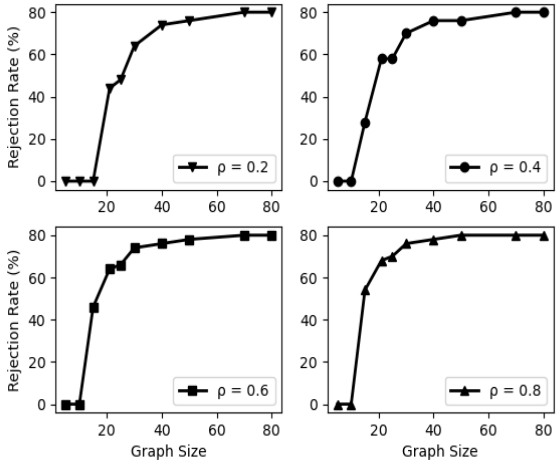

Fig. 5. Rejection rate for different system loads (ρ)

This result is also confirmed by simulations of Figure 5 illustrating the rejection rate of the exact algorithm for increasing physical infrastructures size. With no surprises, systems with high values of ρ are saturated rapidly compared to systems with small values of ρ. In fact, high values of ρ lead to bursts of arrivals that can not be processed in acceptable times, as the exact algorithm suffers from slow convergence time.

V. Conclusion and future work

This paper addressed the problem of resource allocation on an edge network for IoT-based services and discussed the trade-off in terms of processing and bandwidth optimization. We considered a service composed of a chain of functions and developed optimization algorithms for the placement of the functions on the graph composed of heterogeneous nodes and links. We investigated the optimal approach based on Linear Integer Programming and a heuristic based on Graph theory. We highlighted the efficiency of our approaches through simulations, using different graph requests and instances.

Future work will consider reinforcement learning techniques to improve the optimization process and hence minimize the rejection rate of the graph-based algorithm.

References

[1] C. Pielli, A. Biason, A. Zanella, and M. Zorzi, "Joint optimization of energy efficiency and data compression in tdma-based medium access control for the iot," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[2] S. Jo and J. Chung, "Joint optimized cpu and networking control scheme for improved energy efficiency in video streaming on mobile devices," *Mobile Information Systems*, vol. 2017, 2017.

[3] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[4] S. Dräxler, H. Karl, and Z. Mann, "Jasper: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 946–960, 2018.

[5] "https://www.ietf.org/," 2020.

[6] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power iot edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 7–12.

[7] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar, "Stratos: A network-aware orchestration layer for middleboxes in the cloud," *CoRR*, vol. abs/1305.0209, 2013. [Online]. Available: http://arxiv.org/abs/1305.0209

[8] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014, pp. 418–423.

[9] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1346–1354.

[10] "https://www.etsi.org/technologies/689-network-functions-virtualisation," 2020.

[11] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[12] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 816–827, 2014.

[13] E. Amaldi, S. Coniglio, A. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electron. Notes Discret. Math.*, vol. 52, pp. 213–220, 2016.

[14] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Computer Networks*, vol. 114, pp. 95 – 110, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617300087

[15] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 5th ed. Springer Publishing Company, Incorporated, 2012.

[16] M. Rost and S. Schmid, "On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 791–803, 2020.

[17] "https://www.ibm.com/fr-fr/analytics/cplex-optimizer," 2020.