On Scalable In-Network Operator Placement for Edge Computing

Julien Gedeon, Michael Stein, Lin Wang, Max Mühlhäuser
Telecooperation Lab, Technische Universität Darmstadt
Email: {gedeon,stein,wang,max}@tk.tu-darmstadt.de

***Abstract*—The drawbacks encountered in today's cloud computing infrastructures have led to a paradigm shift towards in-network processing, where resources in the core and at the edge of the network are leveraged to perform computations. This can lead to decreased costs and better quality of service for users, e.g., when latency-critical applications are executed close to data sources and users. Deploying applications or parts thereof on these infrastructures requires to place operators (i.e., functional components of applications) on available resources in the network. Solving large instances of this problem in an optimal way is known to be computationally hard and, thus, practically unfeasible. While heuristic approaches exist, they mostly aim at placing functionalities on homogeneous nodes or make unrealistic assumptions for edge computing environments.**

To address this issue, this paper studies the placement problem in the context of a 3-tier architecture consisting of cloud, fog and edge devices. We provide a comprehensive model and propose a heuristic approach to the problem, in which we introduce constraints on the placement decision to limit the possible solution space, leading to a decrease in the solving time for the problem. These constraints exploit the characteristics of our 3-tier network architecture. To demonstrate the feasibility of the approach, we present a general framework that supports different types of heuristics. We validate the approach by implementing example heuristics for each type. We show that our approach can scale to large instances, i.e., it can significantly reduce the resolution time to find a placement solution while introducing only a small optimality gap.

I. Introduction

With the proliferation and increased number of devices at the extreme edge of the network, today's cloud computing infrastructures suffer from high latencies, congested core networks, and they furthermore lack mobility support and location awareness [1]. The new paradigm of in-network processing can be summarized as what has been named *fog computing* [1], [2] or *edge computing* [3], [4]. In-network processing complements cloud computing by leveraging devices in the network and at its edge for storage [5] and computations [6]. These devices can act as cloudlets [7] to dynamically instantiate required services or applications. This requires to make decisions on where to place functional parts of applications, i.e., *operators*, given potential nodes with free computing resources in the network. In the context of edge computing, these operators might be realized on a varying level of granularity, ranging from unikernel-based implementations [8] and container-based microservices [9] to full-size virtual machines. Being able to make quick and cost-efficient placement decisions in those environments is crucial to minimize the provisioning time of services and therefore, provide good quality-of-service to users. While the general operator placement problem has been extensively researched, especially in the domain of complex event processing, most of the existing works consider homogeneous environments, e.g., the placement of operators or network functions in data centers, and focus on the optimization of either resource consumption of computing nodes or on network metrics. In contrast, we consider a 3-tier architecture, consisting of edge, fog, and cloud nodes, with each of these tiers having different characteristics in terms of placement cost, link quality and computational capacity. When mapped to this heterogeneous environment, the operator placement problem becomes more challenging and existing solutions become non-applicable because many assumptions do not hold any more. The general operator placement problem is a variant of the well-known task assignment problem and therefore is NP-hard [10], meaning that for large instances, it cannot be optimally solved in a reasonable amount of time. In in-network processing scenarios, however, we are required to make quick placement decisions in order to ensure quality-of-service for the users. An example application where this is relevant is the processing of sensor data from Internet-of-Things (IoT) devices at the edge of the network [11]. Being able to make placement decisions quickly also allows for frequent reconfiguration. For instance, this is required in case of user mobility or changes in service demands. Therefore, reducing the time it takes to compute an assignment of operators to network nodes is crucial in large-scale dynamic environments.

In this paper, we target scalable in-network operator placement. In particular, we exploit the heterogeneity of the 3-tier architecture of the in-network processing infrastructure and propose heuristics that impose constraints on the placement decision and therefore limit the possible solution space, leading to a decrease in solving time for the placement problem. To this end, we define two general heuristic approaches that take into account our layered hierarchy of edge, fog and cloud nodes: (a) restricting the placement of a certain operator to a subset of nodes and (b) enforcing the colocation of operators on the same nodes. We also investigate a special case of the first approach by pinning operators, i.e., enforcing their placement on specific nodes. For each of our heuristic approaches, we implement example instances to demonstrate the feasibility of this approach. We show that this approach significantly reduces the time required to compute the placement while only leading to a small optimality gap. To the best of our

knowledge, this is the first paper that studies scalable in-network operator placement heuristics in large-scale heterogeneous edge computing environments. In summary, this paper provides three main contributions:

- We provide a comprehensive model for the heterogeneous placement problem for in-network processing in a 3-tier architecture consisting of edge, fog and cloud nodes.
- We propose a scalable yet efficient approach to reduce the solving time of the placement problem by introducing two general classes of heuristics: placement restriction and operator colocation.
- For each of these classes, we implement representative instances and evaluate their impacts on reducing the solving time for the placement problem in different in-network processing scenarios. We furthermore investigate how the combinations of the proposed heuristics perform.

The remainder of this paper is organized as follows: Section II reviews related work. Section III describes our system model. Our approach for operator placement is presented in Section IV and evaluated in Section V. We conclude the paper in Section VI.

II. Related Work

We summarize the state-of-the-art of the problems relevant to the in-network operator placement problem, from which we motivate our work.

Operator Placement: The problem of operator placement has been studied extensively. We can distinguish between network-agnostic [12], [13] and network-aware approaches [14]–[16]. The former do not consider network characteristics (e.g., the latency and available bandwidth on the links), while the latter do. Considering network characteristics is crucial in the edge computing scenarios we examine because delays have a considerable impact on the delivered quality-of-service. It therefore makes sense to consider both the network and the resource dimension (i.e., how many resources are available at specific nodes and what the costs of placing functionality on the nodes are). Our proposed model therefore considers both the network and the resource dimension in the operator placement problem.

Pietzuch [14] and Rizou [15] present approaches to minimize the network usage. However, they do not consider the differences in placement costs or constraints such as available bandwidth on the network links. User-defined constraints for the placement are introduced in [17], but no network costs are taken into account. Several works such as [16] and [18] do not allow the placement of multiple operators on one node, or consider an equal number of operators and nodes [19]. We argue that these are unrealistic assumptions for edge computing scenarios. Others assume uniform capacity of the processing nodes [10] or restrict the underlying topology, e.g., to a tree topology [20]. In contrast, our heuristics do not have these restrictions.

Lakshmanan et al. [21] survey and classify placement strategies for data stream systems. They also provide an analysis regarding which of the surveyed approaches is applicable in which domain. Similar to our work, Cardellini et al. [22] provide a comprehensive model for the operator placement problem. However, in their scalability analysis of the problem, only a homogeneous environment is considered.

Virtual Network Embedding: The problem of operator placement has some similarities to the virtual network embedding problem, e.g., the placement of virtual network functions (VNF) [23]. However, virtual network embedding mostly deals with the placement of virtual networks or virtual network functions in data center environments [24]. Compared to the problem we address in our paper, the infrastructure on which the network functions are placed are more homogeneous wrt. the resources they offer. In contrast to that, we address the heterogeneity of nodes on the different layers (edge, fog and cloud). Furthermore, existing work on VNF placement considers only one data source [25], whereas our operator graphs can have multiple data sources.

Edge/Fog Resource Management: Compared to resource management in cloud data centers, resource management in edge/fog environments is more challenging because fog nodes are more heterogeneous, and uncertainties are imposed by multiple factors such as user mobility. While it is still not available yet, a general centralized framework for holistic fog resource management is envisioned. Based on this vision, a handful of works have been carried out for fog resource allocation and job scheduling [26]–[29]. Recently, Wang et al. propose a service entity placement strategy for edge computing [30], but it is highly tailored for the social virtual reality application scenario. While falling also into this category, our work considers a new 3-tier heterogeneous architecture and incorporates various application scenarios by allowing more flexibility on both the number and the location of sources and sinks, with the goal of achieving scalability while maintaining efficiency. To the best of our knowledge, this paper is among the first attempts for this problem.

III. System Model and Problem Formulation

In this section, we describe the formal model for the in-network operator placement (INOP) problem. Based on the introduced model, we formulate the placement problem with an integer linear program (ILP).

A. Underlay Network

In the in-network processing scenario, the underlay network provides resources for both computing and communication. We model the underlay network as an undirected graph $G = (V, E)$, where V is the set of underlay (physical) nodes on which data processing and analytics can be carried out and $E = \{\langle u, v\rangle \mid_{u,v \in V}\}$ is the set of links interconnecting the nodes. Each node $v \in V$ is also given a capacity C_v, which denotes the maximum computational load that node v can handle. For each link $e = \langle u, v\rangle \in E$, we consider that its capacity is upper-bounded by its bandwidth B_e. We assume that routing in the network is according to shortest paths, which is in line with current data communication networks.

B. Operator Graphs

In typical in-network processing scenarios, a large volume of data is generated by monitoring devices (known as sources) and will need to be processed step by step to produce final results to feed actuators (known as sinks). To characterize the resource requirements of each job, we introduce *operators*, which are independent data processing components, and all relevant operators constitute a job. Consequently, we represent each data analytics job in the system with an operator graph, which is modeled by a directed acyclic graph (DAG) $H = (O, F)$, where O gives the set of operators (including sources and sinks) and $F \subseteq \{\langle o_1, o_2\rangle \mid_{o_1,o_2 \in O}\}$ represents the data flows between the operators. This operator graph represents the logical flow between data sources, operators and sinks. Note that an operator graph can have multiple sources and sinks simultaneously. Each operator $o \in O$ is characterized by a workload w_o, which represents the computational capacity that is required to execute the operator. For each flow $\langle o_1, o_2\rangle \in F$, f_{o_1,o_2} denotes the average bandwidth requirement for the transfer of data between the two operators o_1 and o_2. Note that for the case of multiple operator graphs, we can actually build a dummy operator graph by obtaining the union of the operators and flow sets of all the operator graphs. Therefore, with a bit abuse of notation we will use H to denote the general dummy operator graph for ease of expression.

C. Operator Placement

The INOP problem aims to make decisions on the placement of operators on the underlay nodes. We introduce decision variables to capture the structure of decision-making. First, we introduce $x_{o,v} \in \{0, 1\}$, which characterizes the placement decision of each operator $o \in O$ to every underlay node $v \in V$. We set $x_{o,v} = 1$ if operator o is placed on node v; otherwise $x_{o,v} = 0$. As every operator can only be placed on exactly one node, we have to enforce the constraint

$$\sum_{v \in V} x_{o,v} = 1, \forall o \in O. \tag{1}$$

In addition, operators have to be placed subject to node capacity constraints, i.e.,

$$\sum_{o \in O} x_{o,v} w_o \leq C_v, \forall v \in V. \tag{2}$$

For a pair of operators $o_1 \in O$ and $o_2 \in O$, where $\langle o_1, o_2\rangle \in F$ and each underlay link $\langle u, v\rangle \in E$, we introduce indicating variables $y^{u,v}_{o_1,o_2} \in \{0, 1\}$ to represent whether flow $\langle o_1, o_2\rangle$ will be routed through underlay link $\langle u, v\rangle$. The bandwidth constraints for the underlay links should not be violated, i.e.,

$$\sum_{\langle o_1,o_2\rangle \in F} y^{u,v}_{o_1,o_2} f_{o_1,o_2} \leq B_{u,v}, \forall \langle u, v\rangle \in E. \tag{3}$$

As we adopt shortest-path-based routing in the underlay network, $y^{u,v}_{o_1,o_2}$ actually depends on both $x_{o_1,v}$ and $x_{o_2,v}$. More specifically, for all $\langle o_1, o_2\rangle \in F$, we have

$$x_{o_1,u} = \sum_{v \in V} y^{u,v}_{o_1,o_2} \text{ and } x_{o_2,v} = \sum_{u \in V} y^{u,v}_{o_1,o_2}. \tag{4}$$

We assume that the sources and sinks are pinned, i.e., these placements are given and cannot be altered. This constraint is practically motivated because we assume no control over the sources of data (e.g., sensors and mobile phones) and the consumers of data. Instead, our challenge is to place the operators in between while ensuring low cost.

D. Cost Model

The objective of the INOP problem is to make placement decisions to achieve cost effectiveness. We consider two types of costs, namely, placement costs and link costs. For each node $v \in V$, the cost for handling unit computational load is denoted by p_v. Addressing placement across stakeholder borders, we assume heterogeneous costs for placing operators on nodes. For a given operator $o \in O$ and a given node $v \in V$, the placement cost is given by $p_{o,v}$. Therefore, the aggregated placement cost is given by

$$P = \sum_{o \in O} \sum_{v \in V} p_{o,v} x_{o,v}. \tag{5}$$

For each link $\langle u, v\rangle \in E$ in the network a cost $q_{u,v}$ is associated, which can be used to represent quality-of-service (QoS) attributes such as latency or the combined monetary cost for using the link (e.g., across different network stakeholders). Link costs occur whenever data transfer between two operators uses an underlay link. The total link cost can be represented by

$$Q = \sum_{\langle u,v\rangle \in E} \sum_{\langle o_1,o_2\rangle \in F} q_{u,v} \cdot y^{u,v}_{o_1,o_2}. \tag{6}$$

We believe that the above two cost types are practical and are generic enough to capture a wide range of real-world performance metrics.

To conduct trade-offs between the two costs, we introduce a parameter $\alpha \in [0, 1]$ to weigh the different costs. Given these definitions, the total cost for a placement decision is given by

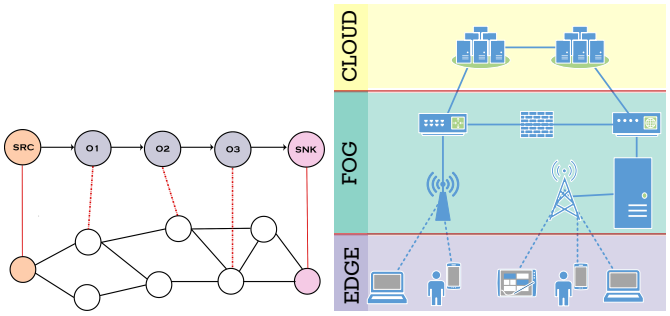
$$Cost = \alpha P + (1 - \alpha) Q. \tag{7}$$

E. Problem Formulation

Based on the presented model, we formulate the INOP problem as follows: For a given underlay network, operator graphs and predefined source-sink pinnings, find an assignment for each operator to an underlay node, such that the cost function (7) is minimal. More formally, we formulate the problem with the following integer linear program.

$$\begin{aligned} \min \quad & Cost \\ \text{s.t.} \quad & (1), (2), (3) \\ & x_{o,v} \in \{0, 1\}, \ \forall o \in O, \ \forall v \in V. \end{aligned}$$

An example result of a simple problem instance can be seen in Figure 1(a). Source-sink pinnings are shown by the solid red lines. Red dotted lines represent a possible result as returned by a placement algorithm. In our evaluation, we will consider a more complex underlay network as well as different types of operator graphs.



(a) Placement example. (b) 3-tier edge-fog-cloud hierarchy.

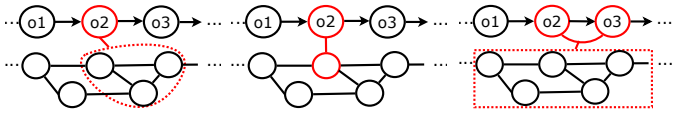
Figure 1. Operator placement in a 3-tier network hierarchy.

IV. Heuristic Approach

To make the placement problem more tractable, we propose an approach in which we use an ILP solver to compute both the optimal solution and a solution that uses our heuristics. For the optimal solution, the input file for the solver contains only the definition of the graphs, alongside with the placement costs and source-sink pinnings. Our heuristics modify the original input problem for the solver in such way that additional constraints are added. More specifically, they impose a set of extra constraints on the placement decision variables $x_{o,v}$. According to [17], adding constraints can lower the time necessary to compute a solution. In detail, we propose two classes of heuristics that follow this approach: placement restriction and operator colocation. We will show that this strategy is more beneficial than the state-of-the-practice, such as greedily placing all operators in the cloud. Before we present the details about the heuristics, we first describe the 3-tier architecture of the system.

A. Edge-Fog-Cloud Hierarchy

Contrary to previous work in this domain, we examine the placement problem specifically in the context of in-network processing, where we consider the network to be in a 3-tier hierarchy, consisting of edge, fog and cloud nodes. Cloud nodes represent data center infrastructures while fog nodes are middleboxes or gateways for end devices. Edge nodes are the end devices, where computations can also take place if sufficient resources are available. We assume cloud nodes to be fully connected, fog nodes to resemble a LAN/WAN topology and edge nodes to have a connection to at least one fog node that acts as a gateway for this edge node. Edge nodes are organized in clusters, depending on which fog node they are connected to. Furthermore, edge nodes have a probability to be connected to more than one fog node and an ad-hoc connection probability to other edge nodes following the device-to-device (D2D) communication technologies. The characteristics in terms of capacity, bandwidth and link costs we assign to these nodes is described accordingly for different evaluation scenarios, as we will see in Section V. Figure 1(b) shows the 3-tier architecture with example devices in each tier. Our motivation for the heuristic approach comes from the observation that the 3-tier architecture of the in-network processing system has some special properties that we could leverage for operator placement. As an example, depending on the locations of the sources and the sinks, we can restrict the placement of operators to a certain subset of underlay nodes, such as underlay nodes in a certain tier. In addition, we could pair operators to achieve optimal decisions before placing the operators.



(a) Restriction (b) Pinning (c) Colocation

Figure 2. Different heuristic approaches.

B. Placement Restriction

Placement restrictions limit the placement of an operator to a subset of nodes (see Figure 2(a)), i.e., for each $o \in O$, we define a subset $V' \subset V$ and we set

$$\sum_{v' \in V'} x_{o,v} = 1. \tag{8}$$

This heuristic can be used to enforce the placement on nodes with desired properties, such as low placement cost, good link connections, or closer to the sources and the sinks. In real-word deployments, one might also want to restrict certain operations to a specific geographic region. One of the reasons for this might be privacy considerations. For our evaluation, we implement this heuristic as follows: For each operator graph, we determine the locations of the sources and the sinks, i.e., on which network tier they reside. If at least one of the sources or sinks are located in the cloud tier, we restrict the placement of the operator graph to the cloud and fog nodes only. However, if either a source or sink is located at the edge of the network, we try to avoid expensive cloud links and restrict the placement of all the operators to either the edge or fog tier. It is important to note that we do not consider fog nodes to be either the source and sink of data, since we consider them network middleboxes or gateway nodes that do not produce or consume application data. Our placement restriction approach can therefore be formalized as follows:

$$\begin{aligned} \min \quad & Cost \\ \text{s.t.} \quad & (1), (2), (3), (8) \\ & x_{o,v} \in \{0,1\},\ \forall o \in O,\ \forall v \in V,\ \forall v' \in V', \\ & V \cap V' = \emptyset. \end{aligned}$$

A special case of placement restriction is operator pinning. Being the most restrictive heuristic, it restricts the placement of an operator to one particular node (see Figure 2(b)) only. According to our model, for a pinned operator-node pair $(v \in V, o \in O)$, we set $x_{o,v} = 1$ to represent this constraint. For the operator pinning heuristic, we implement the following logic: For each operator graph, we try to pin the first operator, i.e., one adjacent to a(one or multiple) source nodes. For the candidate nodes to place this operator, we have to distinguish between

two cases: (a) a single source (or sink) and (b) multiple sources (or sinks) connected to the operator. In the first case, candidate nodes are the source node and its one-hop neighbors in G. In the second case, we consider the two-hop neighborhoods of each of the sources. Out of these subgraphs, we define the candidate nodes as all nodes that appear in all these 2-hop neighborhoods. Then, for each candidate node v, we compute a penalty function $s = \alpha p_{o,v} + (1-\alpha)\bar{q}_v$, where o is the operator to be placed, $p_{o,v}$ is the placement cost and $\bar{q}_v$ is the average link cost of the edges that are adjacent to v. We then pin the operator to the node with the lowest penalty value, since this node will have the lowest weighted placement and link costs to host this operator. Consequently, we obtain a set $P = \{(o_1, v_1), \dots (o_n, v_n)\}$ of all operator-node pinnings and it must hold that

$$\forall (o_i, v_i) \in P : x_{o_i,v_i} = 1. \tag{9}$$

Applied to our original model, pinning results in the following optimization problem:

$$\begin{aligned} \min \quad & Cost \\ \text{s.t.} \quad & (1), (2), (3), (9) \\ & x_{o,v} \in \{0,1\},\ \forall o \in O,\ \forall v \in V,\ P \cap V = \emptyset. \end{aligned}$$

C. Operator Colocation

While our model naturally allows for operators to be colocated on one node, this heuristic enforces the colocation of certain operators to one underlay node. Formally, we define a pair of operators $(o_1 \in O, o_2 \in O)$, and it must hold true that $\exists v \in V$, $x_{o_1,v} = x_{o_2,v} = 1$. As an example, Figure 2(c) depicts the colocation of operators o_2 and o_3. Colocation of operators requires a trading off between placement costs and communication costs. Consider a scenario where operator o_1 precedes operator o_2. Colocating these two operators on one underlay node is most likely beneficial to the overall utility of the system if operator o_2 is lightweight in terms of resource utilization and placement costs, and the communication between o_1 and o_2 requires high bandwidth. To model this, we compute a score for each pair of neighboring operators o_1 and o_2, i.e., $\langle o_1, o_2 \rangle \in F$, defined by

$$s = \frac{f_{o_1,o_2}}{\alpha(\bar{p}_{o_1,v} + \bar{p}_{o_2,v} + w_{o_1} + w_{o_2})}, \tag{10}$$

where f_{o_1,o_2} is the bandwidth of the flow between the two operators, $\bar{p}_{o,v}$ denotes the average placement cost of the operator and w_o the workload of the operator. This score will favor the colocation of operators with high bandwidth demands, while taking into account their placement costs and workload. We then select n_o operator pairs with the highest score to colocate. Empirically, we determined $n_o = \lfloor |O|/5 \rfloor$. Mapped to our optimization problem, we then get a set $O_C = \{(o_1, o_2), \dots (o_n, o_{n+1})\}$ for which the following must hold:

$$\forall (o_i, o_j) \in O_C : \exists v \in V, x_{o_i,v} = x_{o_j,v} = 1 \tag{11}$$

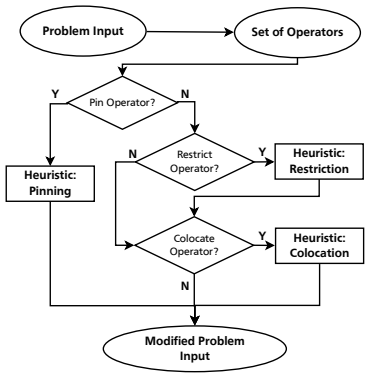


Figure 3. Flowchart of the proposed heuristic framework.

Our modified problem for the colocation of operators can therefore be defined as follows:

$$\begin{aligned} \min \quad & Cost \\ \text{s.t.} \quad & (1), (2), (3), (11) \\ & x_{o,v} \in \{0,1\},\ \forall o \in O,\ \forall v \in V,\ O_C \cap V = \emptyset. \end{aligned}$$

D. Strategies for Combining Heuristics

While the three heuristics described above can be applied individually, we can think of ways to combine them, i.e., to apply more than one heuristic for a given input problem. Figure 3 shows a flowchart that describes the different possible ways of combining our heuristics. We first have the option to pin certain operators to an underlay node. Then, the more general, i.e., less restrictive, heuristics *restriction* and *colocation* can be applied independently. It is important to note that these combinations do not work isolated from but influence each other and might be contradictory. This is especially true for the pinning heuristic, as we will see. For our evaluation, we consider the following three combinations of heuristics:

COMB-1: As the first combination, we apply pinning and then restriction of operators. Note that the restrictions will not be applied to already pinned operators, because this might lead to contradictions, e.g., the pinning computes the placement of an operator to a node which is not in the set of possible nodes according to the restriction heuristic.

COMB-2: For this combination, we first apply pinning and additionally the colocation of operators. Similar to COMB-1, operators that are pinned will not be in the candidate set considered for colocation. For example, consider that the pinning heuristic fixes the placement of an operator o_1 on a node A, and the colocation heuristic enforces the colocation of operators o_1 and o_2, but the capacity of node A is less than the workload of o_1 and o_2 combined. Therefore, when pinning is applied before other heuristics, we do not consider the pinned operators for other heuristics applied afterwards.

COMB-3: Lastly, we leave out the pinning heuristic and combine restriction with colocation. Contrary to the previous combinations, this does not require to exclude operators from the heuristic that is applied second.

V. EVALUATION

To evaluate our approach, we built a simulation tool in Python. To model the problem, we use Pyomo[1] and invoke GLPK[2] as a solver. We first provide an overview of our experimental setup before evaluating the benefits in solving time and the cost overhead of our heuristics. We conclude our evaluation by discussing the results with respect to the trade-off between resolution time and cost overhead.

A. Experimental Settings

We consider different underlay networks for our experiments, as described in Table I. To take into account the different characteristics of nodes and connections at the different layers, Table II summarizes the parameters we set for the capacity, link costs and available bandwidth. In our evaluation, we use the link cost to model the latency induced by using a particular link. To represent different applications that are to be deployed in the network, we define different types of operator graphs as shown in Figure 4. Each of the graphs feature a different number of operators (labeled o_i), sources and sinks. In addition, the sources (labeled src) and sinks (labeled snk) are located at different underlay layers. For example, if an operator graph has both its sources and sinks in the edge layer, this models edge analytics of sensor data where the results are also consumed by edge devices (Figures 4(a), 4(d) and 4(i)). In practice, this kind of operator graph could represent an IoT application, such as a smart home automation. As another example, we can also model classic big data analysis by having sources at the edge and data sinks at the cloud layer (see Figures 4(c) and 4(j)). A hybrid case is depicted in Figures 4(b) and 4(e). In practice, this hybrid case refers to applications where parts of the computed results are required by end devices at the edge and at the same time are sent to the cloud for storage or further analysis. In addition, the graphs depicted in Figures 4(f), 4(g) and 4(h) model the case where the data sources are located at both the edge and cloud layer. An example application for this is the combination of sensor data gathered at the edge with knowledge databases hosted on powerful servers. In conclusion, the operator graphs we consider for our evaluation allow to capture a variety of application scenarios where edge computing is relevant. From these different operator graphs, we derive different input sizes by combining a different number of operator graphs to be placed, labeled *gs1* through *gs11*. The properties of those are summarized in Table III. The operators were assigned a random workload between 2 and 5. The required bandwidth between the operators varies between 5 and 20. Placement costs are also randomly generated within the range of 10 to 30. For each configuration (i.e., an underlay network with the different input sizes), we report the average results of 5 simulation runs. We set $\alpha = 0.5$, meaning that placement and link costs are weighted equally. We implemented our heuristics as described in Section IV.

[1]https://www.pyomo.org/

[2]https://www.gnu.org/software/glpk/

Table I
UNDERLAY NETWORK GRAPHS

	Underlay 1	**Underlay 2**	**Underlay 3**
Cloud Nodes	2	5	10
Fog Nodes	5	10	20
Edge Clusters	2	3	5
Edge Nodes	16	36	70
Edge-fog Connection Probability	0.1,0.3	0.1,0.3,0.2	0.1,0.3,0.2, 0.1,0.4
Edge-fog Connection Probability	0.1,0.3	0.1,0.3,0.2	0.1,0.3,0.2, 0.1,0.4

Table II
PROPERTIES OF NODES

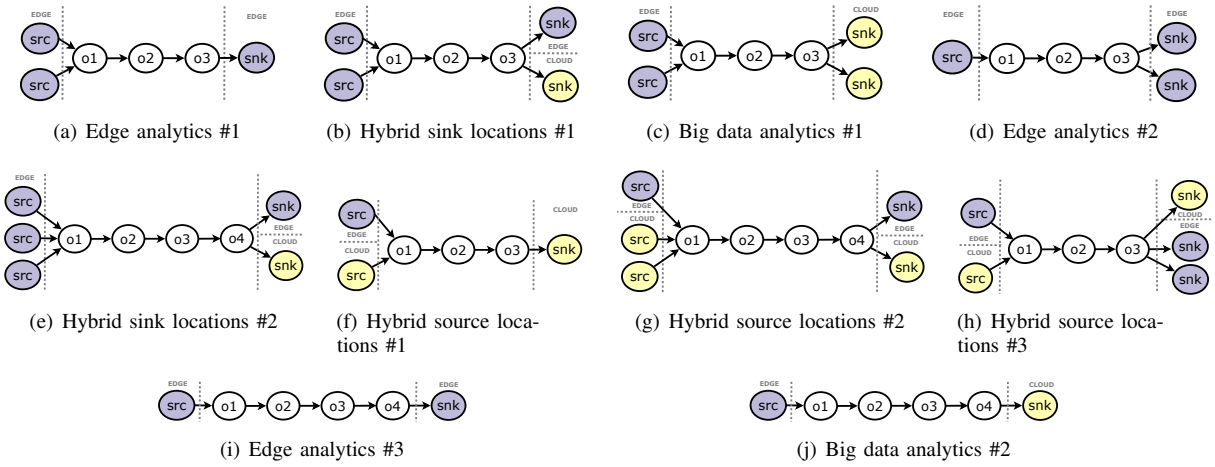
	Cloud Nodes	**Fog Nodes**	**Edge Nodes**
Capacity	100	random(20,60)	random (5,20)
Link Cost	$\mathcal{N}(200, 2500)$	$\mathcal{N}(20, 4)$	$\mathcal{N}(5, 1)$
Bandwidth	10,000	random(1000,5000)	random(10,30)

Table III
INPUT SIZES FOR THE OPERATOR GRAPHS

	Operator Graphs/ Operators	**Sources**	**Sinks**	**Source locations (Edge/Cloud)**	**Sink locations (Edge/Cloud)**
gs1	5/17	8	6	8/0	4/2
gs2	6/20	10	8	10/0	4/4
gs3	7/24	11	9	10/1	4/5
gs4	8/28	14	11	11/3	5/6
gs5	9/32	15	12	12/3	6/6
gs6	10/35	17	15	13/4	8/7
gs7	11/39	19	16	15/4	9/7
gs8	12/44	20	17	16/4	10/4
gs9	13/47	21	19	17/4	12/4
gs10	14/50	22	20	17/5	12/5
gs11	15/54	25	22	20/5	13/6

B. Performance Analysis

First, we analyze the performance of our proposed heuristics, i.e., the reduction in solving time for the placement problem. Figure 5 shows the results of this analysis for the different network sizes. The time it takes to run the heuristics, i.e., the overhead they introduce, is included in the measurement times. Except for applying the colocation heuristic alone, all the approaches reduce the resolution time, regardless of the size of the underlay network or operator graph input size. Colocation alone can increase the resolution time because it takes time to determine the possible combinations to consider. All our other approaches drastically reduce the resolution time. For the different problem input sizes, we were able to achieve decreases of 20 to 80 percent. It is important to emphasize the general trend of a decrease in resolution time as the graph size increases. This validates that our heuristics scale and become even more beneficial in complex problem instances. Out of the implemented approaches, applying pinning together with restriction was the fastest most of the time. However, we will see in the next subsections that this is also the approach that introduces the largest optimality gap. Pinning alone gives us less benefit for the resolution time because

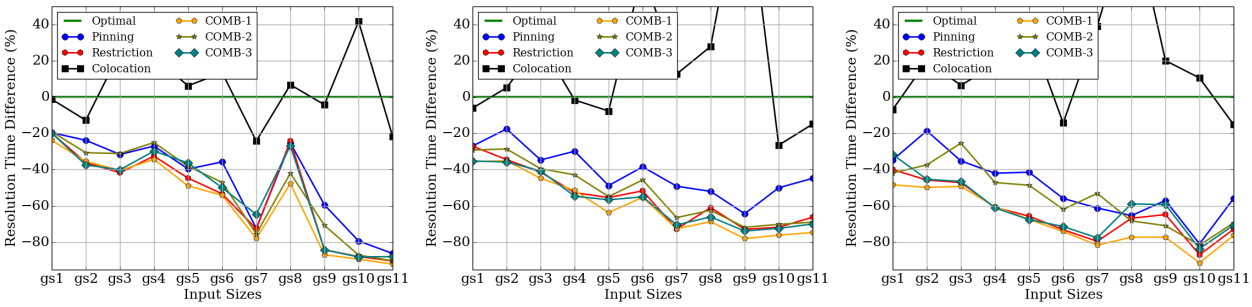


(a) Edge analytics #1 (b) Hybrid sink locations #1 (c) Big data analytics #1 (d) Edge analytics #2

(e) Hybrid sink locations #2 (f) Hybrid source locations #1 (g) Hybrid source locations #2 (h) Hybrid source locations #3

(i) Edge analytics #3 (j) Big data analytics #2

Figure 4. Different operator graphs.



(a) Underlay 1 (b) Underlay 2 (c) Underlay 3

Figure 5. Evaluation results on resolution time.

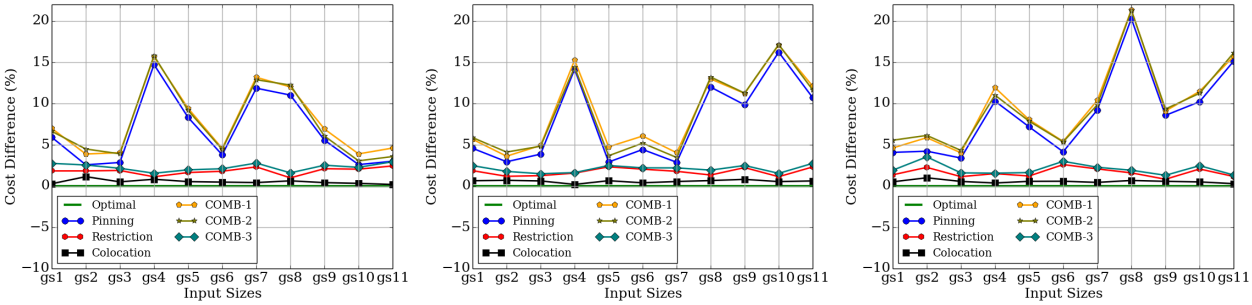
only the first few operators are pinned. Furthermore, fixing the first operator might add complexity for the placement of the remaining operators in the graph. When also applying colocation after pinning, the resolution time is lowered because fewer reasonable placement options are available. In our experiments, restriction and restriction alongside colocation performed similarly wrt. the benefit in resolution time.

C. Optimality Gap

Applying our heuristics will inherently lead to a decrease in the system utility, i.e., the value of the cost function (7) will increase. However, given the benefits regarding the resolution time, this is a trade-off one might be willing to accept in practice. Especially in highly dynamic scenarios, reconfiguring placement decisions is time-critical due to quick changes in network characteristics. In this section, we analyze the optimality gap of our implementation, i.e., we quantify the increase of the cost function for our approaches compared to solving the problem optimally. In Figure 6, we plot this optimality gap. In general, the maximum increase we can observe is only slightly above 20% but is much lower on average. The highest increase is observed whenever pinning is involved since it does not consider the link cost that it might imply for the operators placed later on. The cost difference is the lowest for the colocation heuristic alone because we have more nodes than operators to choose from. Therefore, even though we enforce the colocation of operators, we can find a node that has a combined placement cost close to the optimum. Restriction alone and restriction combined with colocation perform only slightly worse. This is because compared to colocation alone, not all nodes are considered for the placement.

D. Dissecting the Placement Decisions

Figure 7 shows how our proposed approaches influence the placement decisions with respect to the different layers, i.e., how many operators are placed on the cloud, fog or edge layer. We plot the number of operators on different layers for the largest problem instance (Underlay 3, $gs11$). From the results, we can make the following observations: Compared to the optimal solution, pinning is more aggressive in terms of placing operators on the edge, i.e., close to where most of our data sources are located. Since restriction always allows the placement on fog nodes (regardless of the location of data sources and sinks), we can clearly see a trend towards fog placement when applying the restriction heuristic either alone or in combination (COMB-1 and COMB-3). Colocation applied alone leads to nearly the same results as the optimal solution; recall however that this increases the resolution time in most cases. With applying colocation after pinning (COMB-1), we see a slight shift towards cloud placements since after pinning,



(a) Underlay 1 (b) Underlay 2 (c) Underlay 3

Figure 6. Evaluation results on optimality gap.

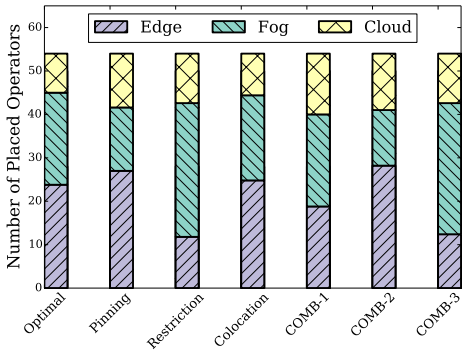


Figure 7. Dissecting the operator placement decisions.

we might have less placement possibilities—especially on the fog layer.

E. Discussion

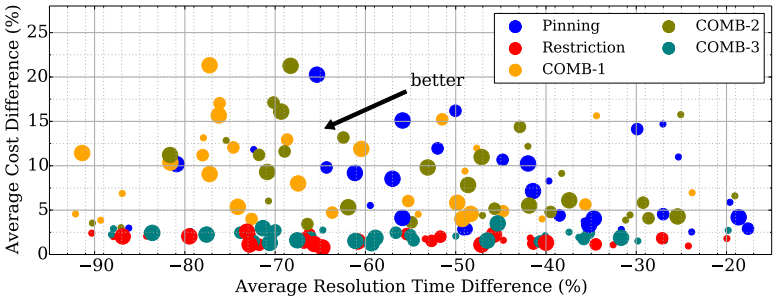
Based on the results obtained, we now discuss the trade-off between the saving in resolution time and the cost overhead of our heuristics. Figure 8(a) depicts this trade-off with the resolution time on the x-axis and the cost overhead on the y-axis. Each dot represents one result for the different graph sizes. The size of each dot represents the size of the underlay network. Since the colocation heuristic alone often increases the resolution time, sometimes dramatically and therefore does not offer a good trade-off, we omit it out in the plot. From the figure, we can first observe the scalability of our approach since there is a trend towards a bigger saving in resolution time for larger underlay sizes. Second, we can see the trade-off between cost optimality gap and saving in time. For instance, COMB-1 leads to good results in terms of time saving but also has one of the highest optimality gaps and a large variance in the optimality gap. Applying restriction alone consistently leads to low cost, however, there is more variance in the saving in resolution time. If we apply restriction and colocation (COMB-3), the results are similar, but for larger underlay sizes, we can see a slight benefit for COMB-3. The similarity of these two is an interesting observation, as we imagine this to be highly dependent on the actual implementation of the colocation heuristic and we plan to examine this in future work. Compared to these, pinning and COMB-2 were found to be the weakest in terms of the trade-off.

We also compare our results with a greedy algorithm for operator placement. This algorithm places every operator on cloud nodes only and chooses the cloud node with the lowest placement cost for each operator. This is the current practice one would employ to place cloud services without considering the edge or fog as possible processing layers. We plot the results of this greedy placement strategy for the largest underlay network (Underlay 3) in Figure 8(b). We observe that while the saving in resolution time is comparable to our heuristics (with a maximum saving of around 90%), the greedy algorithm performs much worse in terms of cost. For larger graph sizes, the costs are three to four times higher than the optimal solution. This is mainly because for the greedy solution, link costs are substantially higher since all data has to be transferred to the cloud, even for operator graphs where data sources and sinks all reside in the edge layer. Compared to that, when applying our heuristics, we saw a maximum increase in the total costs of way below 10% on average.

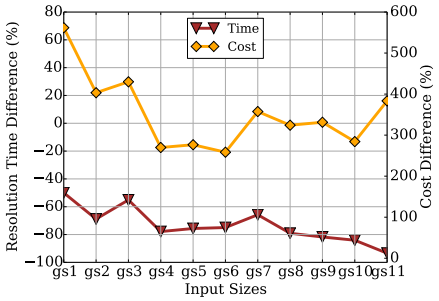
VI. Conclusion and Future Work

In this paper, we presented an approach to tackle the problem of operator placement in a 3-tier edge-fog-cloud infrastructure by applying placement constraints. We introduced two general classes of heuristics to do so. For each of these classes, we implemented sample representatives to demonstrate their feasibility. By evaluating our approach, we were able to show that the resolution time can be decreased dramatically while only leading to a small optimality gap. Our findings can be applied to a variety of practical problems in the emerging domain of edge computing, one of which is the placement of containers. For instance, the popular container-based virtualization framework Docker[3] only considers the available resources and disregards network characteristics when orchestrating different containers in a cluster. In future work, we will implement our proposed heuristics in such real-world systems and incorporate additional heuristics that allow the re-use of operators across different operator graphs. Furthermore, we plan to provide a more detailed cost model for the different

[3]https://www.docker.com/



(a) Time-cost trade-off for the heuristics.



(b) Performance and optimality gap for greedy cloud placement.

Figure 8. Trade-offs and greedy comparison.

processing layers and examine methods to adapt a given placement at runtime. This might be necessary due to changes in the underlay network (e.g., due to user mobility) or a new application being deployed in the network.

Acknowledgement

This work has been co-funded by the German Federal Ministry for Education and Research (BMBF, Software Campus project DynamicINP), by the German Research Foundation (DFG) as part of subproject A1 of the CRC 1053 - MAKI, and by the DFG and the National Nature Science Foundation of China (NSFC) joint project under Grant No. 392046569 (DFG) and No. 61761136014 (NSFC).

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. of Mobidata*, 2015, pp. 37–42.

[2] J. Gedeon, J. Heuschkel, L. Wang, and M. Mühlhäuser, "Fog Computing: Current Research and Future Challenges," in *Proc. of 1. GI/ITG KuVS Fachgespräche Fog Computing*, 2018, pp. 1–4.

[3] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *IEEE Internet Comput.*, vol. 17, no. 5, pp. 70–73, 2013.

[4] M. Satyanarayanan, "The Emergence of Edge Computing," *IEEE Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[5] J. Gedeon, N. Himmelmann, P. Felka, F. Herrlich, M. Stein, and M. Mühlhäuser, "vStore: A Context-Aware Framework for Mobile Micro-Storage at the Edge," in *Proc. of MobiCASE*, 2018, pp. 1–18.

[6] H. Chang, A. Hari, S. Mukherjee, and T. V. Lakshman, "Bringing the cloud to the edge," in *Proc. INFOCOM Workshops*, 2014, pp. 346–351.

[7] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[8] A. Madhavapeddy and D. J. Scott, "Unikernels: The rise of the virtual library operating system," *Commun. ACM*, vol. 57, no. 1, pp. 61–69, Jan. 2014.

[9] D. Jaramillo, D. V. Nguyen, and R. Smart, "Leveraging microservices architecture by using docker technology," in *SoutheastCon 2016*, March 2016, pp. 1–5.

[10] R. Eidenbenz and T. Locher, "Task allocation for distributed stream processing," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[11] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge Analytics in the Internet of Things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, Apr 2015.

[12] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. B. Zdonik, "Scalable distributed stream processing." in *Proc. of CIDR*, 2003.

[13] B. Peng, M. Hosseini, Z. Hong, R. Farivar, and R. Campbell, "R-Storm: Resource-aware scheduling in Storm," in *Proc. of ACM/IFIP/USENIX Middleware*, 2015, pp. 149–161.

[14] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Proc. of IEEE ICDE*, 2006, pp. 1–12.

[15] S. Rizou, F. Dürr, and K. Rothermel, "Solving the multi-operator placement problem in large-scale operator networks," in *Proc. of ICCCN*, 2010.

[16] Q. Zhu and G. Agrawal, "Resource allocation for distributed streaming applications," in *Proc. of ICPP*, 2008, pp. 414–421.

[17] C. Thoma, A. Labrinidis, and A. J. Lee, "Automated operator placement in distributed data stream management systems subject to user constraints," in *Proc. of IEEE ICDE*, 2014, pp. 310–316.

[18] Y. Huang, Z. Luan, R. He, and D. Qian, "Operator placement with QoS constraints for distributed stream processing," in *Proc. of CNSM*, 2011, pp. 309–315.

[19] N. Mohan and J. Kangasharju, "Edge-fog cloud: A distributed cloud for Internet of Things computations," in *Proc. of CIoT*, 2016, pp. 1–6.

[20] F. Starks and T. P. Plagemann, "Operator placement for efficient distributed complex event processing in MANETs," in *Proc. of IEEE WiMob*, 2015, pp. 83–90.

[21] G. T. Lakshmanan, Y. Li, and R. Strom, "Placement strategies for internet-scale data stream systems," *IEEE Internet Comput.*, vol. 12, no. 6, pp. 50–60, 2008.

[22] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, "Optimal operator placement for distributed stream processing applications," in *Proc. of ACM DEBS*, 2016, pp. 69–80.

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

[24] L. Wang, A. F. Anta, F. Zhang, J. Wu, and Z. Liu, "Multi-resource energy-efficient routing in cloud data centers with network-as-a-service," in *ISCC*, 2015, pp. 694–699.

[25] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 98–106.

[26] L. Wang, L. Jiao, D. Kliazovich, and P. Bouvry, "Reconciling task assignment and scheduling in mobile edge clouds," in *Proc. ICNP*, 2016.

[27] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. INFOCOM*, 2016.

[28] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. INFOCOM*, 2016.

[29] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. ICDCS*, 2017.

[30] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. INFOCOM*, 2018.