Efficient Scheduling of Streaming Operators for IoT Edge Analytics

Patient Ntumba, Nikolaos Georgantas, Vassilis Christophides

▶ **To cite this version:**

Patient Ntumba, Nikolaos Georgantas, Vassilis Christophides. Efficient Scheduling of Streaming Operators for IoT Edge Analytics. FMEC 2021 - Sixth International Conference on Fog and Mobile Edge Computing, Dec 2021, Gandia, Spain. hal-03413549

HAL Id: hal-03413549
https://hal.inria.fr/hal-03413549

Submitted on 3 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC), Author version, Accepted

Efficient Scheduling of Streaming Operators for IoT Edge Analytics

Patient Ntumba*, Nikolaos Georgantas*, Vassilis Christophides†
*MiMove INRIA, Paris, France
Email: firstname.lastname@inria.fr
† ETIS ENSEA, Cergy, France
Email: Vassilis.Christophides@ensea.fr

***Abstract*—Data stream processing and analytics (DSPA) applications are widely used to process the ever increasing amounts of data streams produced by highly geographical distributed data sources such as fixed and mobile IoT devices in order to extract valuable information in a timely manner for real-time actuation. To efficiently handle this ever increasing amount of data streams, the emerging Edge/Fog computing paradigms is used as the middle-tier between the Cloud and the IoT devices to process data streams closer to their sources and to reduce the network resource usage and network delay to reach the Cloud. In this paper, we account for the fact that both network resources and computational resources can be limited and shareable among multiple DSPA applications in the Edge-Fog-Cloud architecture, hence it is necessary to ensure their efficient usage. In this respect, we propose a resource-aware and time-efficient heuristic called SOO that identifies a good DSPA operator placement on the Edge-Fog-Cloud architecture towards optimizing the trade-off between the computational and network resource usage. Via thorough simulation experiments, we show that the solution provided by SOO is very close to the optimal one while the execution time is considerably reduced.**

***Index Terms*—Cloud computing, Edge/Fog computing, IoT, data stream, continuous query, heuristic**

I. Introduction

Today, more and more data are delivered in real-time. *Real-time data* are not kept or stored, but are passed along to the end user as quickly as they are gathered. The *time value* of data is essential to many applications requiring real-time (or near real-time) control and automation such as smart transportation [1], security [2], augmented or virtual reality [3]. Working with real-time data requires *stream processing* compared to batch processing of historical data. *Data stream processing and analytics* (DSPA) engines aim to continuously process unbounded data streams generated by multiple distributed sources to extract valuable information in timely manner via a series of continuous operators such as aggregation, filter, join, etc. [4].

DSPA applications are typically deployed in the Cloud in order to benefit from practically unlimited computational resources on demand. Such centralized solutions favor application availability but may suffer from network congestion and delay issues in case of highly dynamic data streams. Additionally, data propagation to the Cloud may compromise privacy of sensitive data. Recent breakthroughs in network technology allow to realize post-Cloud architectures in order to collect and process real-time data. 5G networks [5] enable an increased network bandwidth capacity up to 10 Gbps, low-latency communications down to 1 ms, and a high connectivity density up to 1 million of IoT devices per km^2. This enables IoT devices to transmit a greater number of data streams at high data rates. However, pushing such amount of real-time data systematically to the Cloud could definitely cause excessive stress on network resources and introduce network delays for real-time IoT applications.

In this respect, the solution lies in taking advantage of the emerging Edge/Fog computing paradigms [6]. They introduce a middle-tier between the IoT devices and the Cloud, in order to extend computational resources close to IoT devices and hence to process data streams near their sources, consequently to reduce network consumption and network delay and to enforce data privacy. The Edge/Fog computing relies on geographical distributed heterogeneous nodes such as mobile or fixed IoT devices (e.g., camera, connected vehicle, smartphone, etc.), small data centers, routers, wireless base stations, etc. that come with stringent resource constraints in terms of limited computational resources (e.g., CPU, RAM, etc.) and limited power supply (e.g., rechargeable batteries, solar energy, etc.) that may have to be shared amongst several DSPA applications.

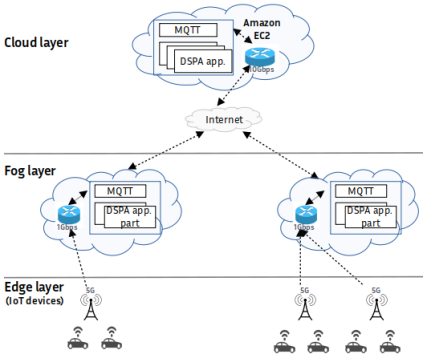

Fig. 1. Edge-Fog-Cloud architecture example

However, the majority of research efforts related to Edge/Fog-based data stream processing focus on optimizing network resource usage and end-to-end network delay, while only ensuring that the capacity of Edge/Fog computational resources is not exceeded [7]–[9]. Clearly, DSPA operators should also account for both the computational capacities available at the heterogeneous Edge/Fog nodes and the highly dynamic workloads related to the spatio-temporal dynamics of real data produced in the wild for multiple DSPA applications [10]. Thus, to distribute computations of a DSPA application across the Edge/Fog and Cloud layers, we need to consider *optimal trade-offs* between the *management of network resources* used to reach the Cloud (i.e., network bandwidth, network delay) and the *management*

of computational resources (CPU, RAM) exploited at Edge/Fog nodes.

In this paper, we assume the Edge-Fog-Cloud architecture as a hierarchical wide area resource network shareable by several DSPA applications [6]. As depicted in Figure 1, at the root of the hierarchy we consider a Cloud node providing computational and network resources on demand. Linked to Cloud node, a set of Fog nodes provide limited computational and network resources at the Fog layer [11]. We assume an overlay network based on the publish-subscribe protocol MQTT [12] to transport data streams across the Edge-Fog-Cloud architecture. In this respect, we consider at the Edge mobile IoT devices (e.g., vehicles) per geographical area that produce and publish data at a certain rate (e.g., 4KB/s [13]) each one to the MQTT instance deployed on the closest Fog node via 5G antennas. We assume that these antennas enable high connectivity density and lossless communication for IoT devices. If a part of the DSPA application is deployed on this Fog node, the MQTT instance transfers these data to the Fog application part and expects the resulting output data back in order to send them to the MQTT instance deployed in the Cloud for further processing. Otherwise if no application part is deployed on the Fog node, the Fog MQTT instance transfers directly these data to the Cloud MQTT instance.

We consider for example a DSPA application for country wide traffic monitoring that gives periodically the traffic status per region [14]. This DSPA application aggregates and analyses the data streams (e.g., vehicle identifier, GPS-location, driving speed) produced by the moving vehicles in the different regions of a country. Given the varying number and the mobility of vehicles, data streams can be very dynamic in a geographical area.

The related problem consists in identifying the DSPA operator replication and placement scheme across heterogeneous Edge-Fog-Cloud resources in order to optimize the resource usage while satisfying the maximum resource usage constraints with respect to the evolution of the data stream rates. In this paper, we formulate the corresponding optimization problem and show that it is NP-hard.

In our previous work [15], we introduced a holistic cost model of both network (bandwidth, delay) and computational resources (CPU/RAM of nodes) and formulated the operator scheduling problem as a single objective optimization (SOO) problem of the combined resource cost, and further as an instance of integer linear programming (ILP) subject to inequality constraints. In this respect, we employed the CPLEX tool [16] to find an optimal solution. However, our SOO-CPLEX solution may incur a high execution time for large problem sizes due to the NP-hardness of the optimization problem. Hence, this solution is not appropriate for dynamic scheduling of DSPA operators in response to highly dynamic data streams.

In this paper, we introduce a heuristic solution, which we call SOO, to the streaming operator scheduling problem. SOO leverages the particularities of the problem in hand in order to reduce the execution time while achieving excellent optimality, both in best-case and worst-case executions.

We experimentally evaluate the operator scheduling achieved by SOO via simulation on iFogSim [19] under different configurations of the Edge-Fog-Cloud architecture and numbers and rates of the data streams. In best-case execution, SOO finds the optimal solution (same as SOO-CPLEX) with a performance gain up to 63% in terms of execution time for the problem sizes that we experimented with. In worst-case execution, SOO approximates the optimal solution with up to 0.01% of approximation error, while reducing the execution time up to 55%.

The remainder of this paper is as follows: Section II introduces the system model as well as our modeling of the resource usage cost, and formulates the resource allocation problem. Section III presents the SOO scheduling algorithm. Section IV details the experimental evaluation. Finally, Sections V and VI present respectively the related work and the conclusion.

II. Resource Allocation Problem

A. System model

1) ***DSPA application****:* is modeled as a directed acyclic graph of operators, denoted by G , where the vertices are the set of continuous operators and the edges are the set of data stream flowing between two operators [4]. G topology further includes the sources that produce the raw data streams S_j of rate $|S_j|$ consumed by DSPA operators and sinks that capture the stream of the computed results. To cope with the infinite nature of data streams, we consider that continuous operators are executed in time windows ω_x to process a finite set of data items arising within a time interval. Furthermore, we consider the following parameters:

a) Operator selectivity (sel_x): is the ratio between the input and output data rate of an operator O_x.

b) Cumulated operator selectivity ($csel_x$):: is the product of operator selectivity from a source to a target operator O_x following the topological order of operators in the application graph G.

c) Edge data rate ($\lambda_{x,y}$): is the rate of data stream flow between two operators, from a source to an operator or from the application graph to a sink. This is calculated as the product of the cumulated operator selectivity of the upstream operator (or data source) and the rate of the raw data stream S_j: $\lambda_{x,y} = csel_x \cdot |S_j|$.

d) Edge-cut (ec_j): used in graph theory, is the partitioning of the application graph into two disjoints subgraphs. Any edge-cut contains a set of edges that have one endpoint in each subgraph of the partition. Hence let $|ec_j|$ denotes the value (or rate) of an edge-cut ec_j which is the sum of data rates of the edges crossing this edge-cut.

e) Minimum edge-cut: is the edge-cut that has the smallest value among all the edge-cuts in the application graph G. To calculate the minimum edge-cut we may rely on the Edmond-Karp algorithm [22] that is proven to be efficient for dense graphs.

f) Operator data load (ρ_x): is the aggregation of the input data streams per time window ω_x, we assume a specific window, ω_x=1sec:

$$\rho_j = \sum_{i=1}^{I} \omega_j \cdot \lambda_{i,j} \tag{1}$$

Where I is the maximum number of upstream operators O_x producing data stream at rate $\lambda_{x,y}$ towards the operator O_y.

g) Operator cost (c_x): is the computational cost in terms of CPU and/or memory usage for an operator O_x to process a unitary data load ρ_x.

2) ***Edge-Fog-Cloud architecture****:* is abstracted as hierarchical wide-area resource network as depicted in Figure 1. The Edge layer consists of M IoT devices $E_i, i = \{1, ..., M\}$ moving in N geographic areas, the Fog layer consist of N Fog nodes $F_j, j = \{1...N\}$ where each Fog node F_j provides nearby computational service to the geographic area j. And one Cloud node C on the top of the hierarchy.

In this respect, we consider the following parameters: cm_{E_i} is the maximum CPU/memory of an Edge node E_i. The network link from an Edge node E_i to its nearest Fog node F_j is characterized by a network delay $nd_{E_iF_j}$ and the network bandwidth capacity nb_{EF_j} of all incoming links to the fog node F_j. cm_{F_j} is the maximum CPU/memory of a Fog node F_j. The network link from a Fog node F_j to the Cloud node C is characterized by a network delay nd_{F_jC}

and the network bandwidth capacity nb_{F_jC} for all incoming links of the cloud node. Finally, cm_C is the maximum CPU/memory of the Cloud node.

We consider also cmu_{E_i}, cmu_{F_j} and cmu_C as the CPU/memory usage respectively on a Edge node E_i, a Fog node F_j and the Cloud node C; additionally $nbu_{E_iF_j}$ and nbu_{F_jC} are the network bandwidth usage on network links respectively from an Edge node E_i to a Fog node F_j and from a Fog node F_j to the Cloud node C.

Finally, we consider S_j as the sum of data streams arriving to a Fog nodes F_j and produced by $m_j(t) \leq M$ IoT devices moving at a time t in a geographic area j. Given the above, at time t $M = \sum_{j=1}^{N} m_j(t)$.

B. Computational resource usage cost

Our cost model extends [23] in order to take into account the usage cost of computational resources shared by multiple DSPA applications. More precisely, we weight the usage of a computational resource by an individual application by the inverse of its computational capacity. The overall computational resource usage cost is:

$$cru = \sum_{i=1}^{M} cmu_{E_i} \cdot \frac{1}{cm_{E_i}} + \sum_{j=1}^{N} cmu_{F_j} \cdot \frac{1}{cm_{F_j}} + cmu_C \cdot \frac{1}{cm_C} \quad (2)$$

Where M is the total number of the Edge nodes E_i, N is the total number of the Fog nodes F_j, cmu_{E_i}, cmu_{F_j} and cmu_C are the CPU/memory usage respectively on the Edge node E_i, the Fog node F_j and the Cloud node C.

The more computational resources a node has the less is the cost for running a specific computation [25]. Thus, in the Cloud computational resources are practically infinite ($cm_C \rightarrow \infty$), so the weight is very small, practically zero ($\frac{1}{cm_C} \rightarrow 0$). In the Fog, computational resources are limited, so the weight is higher ($\frac{1}{cm_{F_j}} \rightarrow 1$) and the cost for using these resources is high, hence computational resources in the Fog should be used with parsimony as in addition they may have to be shared among multiple applications. Similarly to [25], computational resources of IoT devices at the Edge are not used. An approach to account for mobile and fixed computational resources at the Edge will be addressed in our future work. Then, Formula (2) becomes:

$$cru = \sum_{j=1}^{N} cmu_{F_j} \cdot \frac{1}{cm_{F_j}} \quad (3)$$

cmu_{F_j} is the sum of the CPU/memory usage required by each operator of the subgraph $Gmig_j \in G$, which is replicated on the Fog node F_j:

$$cmu_{F_j} = \sum_{O_x \in Gmig_j} \rho_x \cdot c_x \quad (4)$$

C. Network resource usage cost

We consider that modeling the Edge-to-Fog network as a local-area network (LAN) and the Fog-to-Cloud network as a wide-area network (WAN) as in [25] is too restrictive, since a Fog node may be located at shorter but still considerable distance from the Edge. Thus, in our cost model the Edge-Fog-Cloud architecture is considered as a hierarchical wide area resource network. Hence, two conflicting factors are involved: network bandwidth increases up the hierarchy, however also network delay increases up the hierarchy. In the literature [17], [18], concerning peer node networks, network delay is used as the only weight factor for differentiating network links. We additionally include network bandwidth as a weight factor: using network links of limited capacity with parsimony allows an efficient sharing among several DSPA applications.

Then, the overall network resource usage cost at time t is:

$$nru = \sum_{j=1}^{N} \sum_{i=1}^{m_j(t)} \frac{nbu_{E_iF_j}}{nb_{EF_j}} \cdot \frac{nd_{E_iF_j}}{nd_{min}} + \sum_{j=1}^{N} \frac{nbu_{F_jC}}{nb_{FC}} \cdot \frac{nd_{F_jC}}{nd_{min}} \quad (5)$$

where $m_j(t) \leq M$ is the number of the Edge nodes E_i sending data streams to their closest Fog node F_j at a time t, N is the number of the Fog nodes F_j. $nbu_{E_iF_j}$ and nbu_{F_jC} are the network bandwidth usages on the network links respectively from E_i to F_j and from F_j to the Cloud node C. nd_{min} is the minimum network delay in the resource network ($nd_{min} = \min_{i,j}\{nd_{E_iF_j}, nd_{F_jC}\}$). As the Edge is much closer to the Fog, normally one of the Edge-to-Fog network links has the minimum network delay.

Given that no processing is done at the Edge, each entire raw data stream S_j produced at the Edge reaches the Fog anyway. Thus in Formula (5), the cost part concerning Edge-Fog network links is fixed (set as c constant). Furthermore, although the network delay on a wide-area network link can vary due to the condition of the underlying physical link that is shared by multiple data connections [17], we assume that the network delays between each Fog node and the Cloud can be considered equal and that nd_{F_jC} and nd_{min} can be assigned statically calculated average values. Then, we can consider $\frac{nd_{F_jC}}{nd_{min}}$ as a constant a. In this way Formula (5) becomes:

$$nru = c + a \cdot \sum_{j=1}^{N} nbu_{F_jC} \cdot \frac{1}{nb_{FC}} \quad (6)$$

The network bandwidth effectively used on all the Fog-to-Cloud network links is defined as follows:

$$B = \sum_{j=1}^{N} nbu_{F_jC} \quad (7)$$

While we considered Cloud computational resources as practically infinite in our resource cost model, we opt for considering Cloud network bandwidth as a resource the usage of which incurs a cost that should be taken into account. Indeed, Cloud providers rely on contracts with ISPs for network bandwidth. For distributed data intensive applications, the usage of the Cloud network bandwidth can be a bottleneck when sending huge volumes of data streams. Hence, for such applications the (monetary) cost charged by Cloud providers for network bandwidth usage can be much larger than the cost charged for computational resource usage. Thus, we assume an upper threshold $Bmax$ of B which is set for a specific DSPA application.

D. Problem statement

The operator placement problem is a conflicting optimization problem, since minimizing the (overall Fog) computational resource usage cost, i.e., cru, implies placement of all operators of G in the Cloud, which results in low (zero) cru and high (overall Fog-to-Cloud) network resource usage cost, i.e., nru; while minimizing nru implies placement of all operators of G in the Fog, which results in high cru and low nru. Hence, there is a trade-off to address between the two metrics. Similarly to [15], we transform this conflicting optimization problem to a single-objective optimization (SOO) problem by applying weights to the different optimization metrics. As cru and nru are defined in different scales, we first normalize these two metrics by using the min-max scaling technique. Then, cru defined in Formula (3) is normalized as follows:

$$CRU = \frac{cru - cru_{min}}{cru_{max} - cru_{min}} \quad (8)$$

Where cru_{min} is the minimum value set to 0 and $cru_{max} = N$ is the maximum value set to the number of Fog nodes.

Furthermore, we consider NRU as the normalized form of nru defined in Formula (6), where the constants c and a are eliminated.

Following from the above, we aim to minimize the overall resource usage cost RU defined as the weighted sum of CRU and NRU:

$$\text{minimize} \quad RU = w_c \cdot CRU + w_n \cdot NRU \tag{9}$$

$$\text{subject to} \quad cmu_{F_j} \leq cm_{F_j}\ j = \{1, \ldots, N\}, \tag{10}$$

$$B \leq Bmax. \tag{11}$$

Where $w_c \geq 0$ and $w_n \geq 0$ are respectively the weights for computational and network resource usage cost, which enable to specify a usage preference for one of the two types of resources over the other. In the rest of the paper, we consider that $w_c = w_n = 1$.

Equation (10) represents the computational resource usage (i.e., CPU/RAM usage) constraint for each individual Fog node (max available capacity). Similarly, Equation (11) is the constraint on the Fog to Cloud network bandwidth usage.

Finally, we consider the operator replicability constraint: an operator can be replicated on a Fog node if there is sense in applying its operation only to the local data stream. Otherwise, if this operation must be applied to the global data stream, the operator can only be deployed in the Cloud.

III. Heuristic solution

To solve our streaming operator scheduling problem, we need to search in the graph G the replication and migration point (edge-cut) ec_j of each data stream S_j and Fog node F_j. The edge-cut ec_j delimits the sub-graph $Gmig_j \in G$ to replicate and migrate on the corresponding Fog node F_j. $Gmig_j$ should satisfy the computational resource usage constraint in (10) and the operator replicability constraint. On the other hand, the processed data streams S_j that will be transmitted on the Fog to Cloud network links should jointly satisfy the network bandwidth usage constraint (11).

We observe that, when selecting an edge-cut ec_j as the replication and migration point of a data stream S_j and Fog node F_j, we can individually calculate the effect of this selection on the overall resource usage cost. Thus, we assume that the computational and network resource usage costs can be split for each individual data stream S_j as follows:

$$RU = \sum_{j=1}^{N} RU_j, \text{ where } RU_j = CRU_j + NRU_j \tag{12}$$

We consider RU_j, CRU_j and NRU_j as respectively the contributions to RU, CRU, and NRU for processing a data stream S_j.

Considering our streaming operator scheduling problem with the objective of minimizing RU, given the values of RU_j at each edge-cut ec_j and for all data streams S_j and Fog nodes F_j, under the global constraint of network bandwidth usage, our problem can be mapped to a 0/1 knapsack problem [20], which is NP-hard.

To address this NP-hardness, our proposed heuristic solution called Single Objective Optimization (SOO) algorithm is inspired from BOSe [21]. SOO first applies a parallel search to generate a solution that attempts to minimize the overall resource usage cost RU by minimizing independently RU_j for each data stream S_j. If this solution satisfies the constraint $B \leq Bmax$, then the achieved RU is optimal. Otherwise the problem may not have a solution or, if a solution exists, finding the optimal solution is NP-hard.

Thus in a second step, SOO applies a greedy search that produces local optimal solutions to approximate the global optimal solution in a reasonable amount of time. In this respect, we first use the minimum edge-cut approach to identify the solution that minimizes NRU by minimizing independently NRU_j for each data stream S_j. If the resulting Fog-to-Cloud bandwidth usage does not satisfy the constraint $B \leq Bmax$, the problem has no solution. By relaxing accordingly the $Bmax$ constraint, we may accept this last solution as the best possible one. On the other hand, if the constraint $B \leq Bmax$ is satisfied, SOO applies a greedy search to improve this solution by further reducing RU. We present the pseudo-code of our proposed SOO heuristic in Algorithm 1.

Algorithm 1: SOO

```
Input: G, application graph
Input: Grep, subgraph of replicable operators of G
Input: Bmax, upper threshold for bandwidth usage
Input: Sraw, set of raw data streams S_j
Updated ← ∅, set of S_j on which ΔRU is applied S_j
M ← ∅, set of Gmig_j, replicable subgraph per S_j
RM ← ∅ set of edge-cuts ec_j ∈ Grep per S_j
RU ← 0, overall resource usage cost
B ← 0, Fog-to-Cloud network bandwidth usage
M ← RUminCut()
if B > Bmax then
    Selected ← dataMinCut()
    if B ≤ Bmax then
        ΔRUset ← edgeCutMove(Selected, RM, M)
        Sort ΔRUset in increasing order
        Pull ΔRU on top of ΔRUset
        while ΔRU < 0 do
            Get S_j, e_{j_k}, Gmig_{j_k} corresponding to Δ_RU
            ec_j ← MR[j]
            if B − |ec_j| + |ec_{j_k}| ≤ Bmax and
             Updated.contains(S_j) == False then
                MR[j] ← e_{j_p}
                M[j] ← Gmig_{j_k}
                B ← B − |ec_j| + |ec_{j_k}|
                RU ← RU + Δ_RU
                Updated ← Updated ∪ S_j
            Pull ΔRU on top of ΔRUset
Rewrite G to include all Gmig_j (or Gmig_{j_p}) ∈ M
Deploy the new G
```

A. Parallel search

We use the function **RUminCut**, in which for each data stream S_j we identify $Gsat_j \subseteq Grep$ as the part of $Grep$ that satisfies the computational resource constraint on the Fog node F_j receiving the data stream S_j. Then, we select the edge-cut $ec_j \in Gsat_j$ that produces the minimum RU_j. We set this edge-cut as the replication and migration point of S_j on the Fog node F_j. Then, we identify the candidate sub-graph $Gmig_j \subseteq Gsat_j$ to replicate on Fog node F_j delimited by the identified edge-cut ec_j. Once we identified $Gmig_j$ for all the data streams S_j, we calculate the resulting Fog-to-Cloud network bandwidth usage B and the overall resource usage cost RU. The pseudo code of $RUminCut$ is presented in Algorithm 3.

B. Greedy search

1) ***dataMinCut****:* Similarly to $RUminCut$, for each data stream S_j, we pass by $Grep \subseteq G$ and $Gsat_j \subseteq Grep$ for selecting, in this case, the minimum edge-cut ec_j that produces the minimum NRU_j. In this way, we identify the resulting candidate subgraph $Gmig_j \subseteq Gsat_j$ delimited by the minimum edge-cut ec_j. Then, we compute the overall NRU, RU and B. For space saving, we omit to present its pseudo-code.

*2) **edgeCutMove**:* Departing from the solution of $dataMinCut$ and for each individual data stream S_j, any backward move of an edge-cut ec_j in $Gmig_j$ will decrease CRU_j (consequently also CRU) while it will increase or decrease NRU_j (consequently also NRU).

For example, in Figure 2 we identify $Gmig_2$ as the subgraph to replicate on the Fog at the minimum edge-cut ec_3 for processing the stream S_2. The possible edge-cut backward moves to consider are: ec_{2_1}, ec_{2_2}, ec_{2_3} and ec_{2_4}; the contributions to RU are respectively RU_{2_1}, RU_{2_2}, RU_{2_3}, RU_{2_4}.

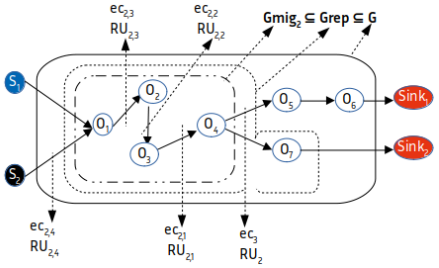

Fig. 2. Example of edge-cut move

Given the above, for any edge-cut backward move ec_j on a data stream S_j, we assume the changes ΔCRU and ΔNRU respectively in CRU and NRU, where ΔCRU is in general negative and ΔNRU may be negative or positive. Thus, we can calculate the change in the resource usage cost RU as follows: $\Delta RU = \Delta CRU + \Delta NRU$, and Formula (9) can be expressed as $RU' = RU + \Delta RU$.

The objective of the greedy search is first to identify all possible edge-cut backward moves, as described by the function $edgeCutMove$ presented in Algorithm 2, then to apply the edge-cut moves that produce the smallest $\Delta RU < 0$, while satisfying the constraint $B \leq Bmax$. To apply the edge-cut moves, we retrieve ΔRUset, the result of $edgeCutMove$, which is the set of ΔRU's, then we sort this set in increasing order. We pull from the top of the set the smallest ΔRU. If ΔRU is lower than 0, we apply its corresponding edge-cut move ec_{j_k} as the current replication and migration point of the data stream S_j, and $Gmig_{j_k}$ as the current subgraph to deploy in the Fog, only if the constraint $B \leq Bmax$ is satisfied. Then, we pull from ΔRUset the next lowest ΔRU to continue improving RU, as long as the remaining ΔRUset is not empty or we do not yet encounter a $\Delta RU \geq 0$. It worth noting that we update at most once each data stream S_j with its best (lowest) ΔRU. The pseudo-code for applying the backward edge-cut moves is presented in Algorithm 1, lines 11-22.

IV. Evaluations

We use the simulator tool iFogSim [19] driven by realistic parameter settings that are proposed in [27]. As benchmarks, we consider: (i) SOO-CPLEX that produces optimal solutions, and (ii) RCS (Resource Constraint Satisfaction) as baseline approach [15]. RCS enhances pure Cloud-based stream processing by dynamically placing the streaming operators between the Fog and the Cloud in synergy with the evolution of IoT data stream rates. More specifically, assuming an initial deployment of all the operators in the Cloud, RCS minimally uses the Fog computational resources to satisfy the constraints $B \leq Bmax$ and $B \geq Bmin$, where $Bmin$ is a lower threshold of B that we add to avoid the oscillation of operator placement between the Fog and the Cloud. In this respect, we implement SOO, SOO-CPLEX and RCS in Java inside the simulator. To ensure that SOO-CPLEX produces the optimal solution, we set

Algorithm 2: edgeCutMove

```
1  Function edgeCutMove(Selected, RM, M, B):
2      ΔRUset ← ∅
3      for S_j ∈ Sraw do
4          ec_j ← RM[j]
5          Get RU_j corresponding to edge-cut ec_j
6          while ec_{j_k} is valid do
7              Gmig_j ← M[j]
8              Get Gmig_{j_k} ⊆ Gmig_j delimited by ec_{j_k}
9              Compute cmu_{F_j} and normalize to CRU_j
10             Compute nbu_{F_jC} and normalize to NRU_j
11             RU_{j_p} ← CRU_{j_k} + NRU_{j_k}
12             ΔRU ← RU_{j_k} − RU_j
13             ΔRUset ← ΔRUset ∪ ΔRU
14             Keep and map ΔRU, S_j, Gmig_{j_k} and ec_{j_k}
15             ec_j ← ec_{j_k}
16             Get ec_{j_k} preceding ec_j ∈ Gmig_j
17     return ΔRUset
```

Algorithm 3: SOO::RUminCut

```
1  Function RUminCut(Sraw, RM, M, RU, B, Grep):
2      for S_j ∈ Sraw do
3          Get Fog node F_j to serve S_j
4          Identify Gsat_j ⊆ Grep while cmu_{F_j} ≤ cm_{F_j}
5          Find ec_j in Gsat_j with the minimum
             RU_j ← CRU_j + NRU_j
6          Find Gmig_j ⊆ Gsat_j delimited by ec_j
7          RU ← RU + RU_j
8          B ← B + |ec_j|
9          M[j] ← Gmig_j
10         RM[j] ← ec_j
11         Keep and map ec_j to RU_j
12     return M
```

the optimality gap (OG) to 0.0%. OG is a metric used by CPLEX to trade off between optimality and performance.

For our simulation experiments, we build a model of the TLC application (New York City Taxi and Limousine Commission rides) [28]). The TLC application finds the busiest driver every two hours, where each vehicle emits at the end of a ride a data record containing driver identification, pick-up and drop-off times and locations. It comprises 5 operators and 1 sink as depicted in Figure 3.

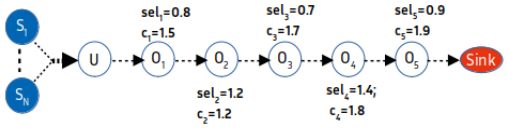

Fig. 3. DSPA application used in the evaluation

We also build a simulation model of the Edge-Fog-Cloud architecture presented in Figure 1, where we scale the numbers of Fog nodes and IoT devices.

We statically simulate the variability of the data stream rates arriving to the Fog nodes by selecting randomly (uniform distribution) 10 values of M (number of IoT devices) in the interval $[5000, 50000]$, where each IoT device produces data at a rate of 4KB/s [13]. Then for each value of M, we set an interval $[0, M]$ in which we randomly (uniform distribution) set $m_j(t)$ IoT devices per geographical area j, so that the sum of $m_j(t)$ be equal to M. In this respect, each Fog

node F_j receives data stream S_j at rate $|S_j| = m_j(t) \times 4KB/s$. We wish to have around 15 results of RU, CRU, and NRU for each value of $M \in [5000, 50000]$ and to plot the average of these results per value of M. Hence, we repeat the splitting of each value of M per geographical area 15 times. In this respect, the total data rate reaching the Fog follows a random (uniform distribution) sequence of the 10 values of M $\times 4KB/s$ repeated 15 times. We feed this sequence to SOO, RCS and SOO-CPLEX. For RCS, we maintain the previous state of the DSPA application between the successive experiments. For SOO and SOO-CPLEX, each experiment is independent, since these two approaches calculate each time a new operator placement without taking into account the previous deployment state.

A. Parameter setting

We evaluate the performance of SOO against RCS and SOO-CPLEX in both best-case and worst-case executions of the SOO algorithm. Best-case execution is related to the set of experiments where SOO applies only $RUminCut$. Worst-case execution is related to the set of experiments where, additionally, SOO applies a greedy search including $dataMinCut$ and $edgeCutMove$.

We observed that SOO executes in the worst case only in a narrow subspace of parameter settings. We generate parameter settings inside and outside this subspace by proceeding as follows: assuming that the graph G of the DSPA application is a two-degree graph (as the one of Figure 3), we introduce the following two parameters: i) the cost of operator O_y for processing a unitary data load entering the graph, based on the cumulated selectivity of its upstream operator O_x: $csel\_c_y = csel_x \cdot c_y$, where c_y is the cost of the operator O_y; and ii) the cumulated sum of such costs up to an operator O_y: $sum\_csel\_c_y = sum\_csel\_c_x + csel\_c_y$.

TABLE I
PARAMETERS OF THE DSPA APPLICATION OF FIGURE 3

Operators	O_1	O_2	O_3	O_4	O_5
$csel_x$	0.8	0.96	0.672	0.9408	0.84672
$sum\_csel\_c_x$	1.5	2.78	4.412	5.6216	7.40912

Table I shows the values of $csel_x$ and $sum\_csel\_c_x$ for the DSPA application of Figure 3. Given the formulas of CRU and NRU, we identify the following equilibrium, which represents a balance between the maximum CRU and NRU values over all possible edge-cuts selected as replication and migration points:

$$\frac{\max\limits_{O_x \in G}(sum\_csel\_c_x)}{\sum_{j=1}^{N} cm_{F_j}} \approx \frac{\max\limits_{O_x \in G}(csel_x)}{Bmax} \quad (13)$$

This equilibrium qualifies the case when, while moving from an edge-cut to another, network resources are replaced by computational resources of, more or less, equal cost. Then $RU = CRU + NRU$ will be almost constant.

We have observed that, for a best-case execution of SOO, the Fog computational resources should be less costly (i.e., relatively more abundant) than the Cloud network resources. To do so, we increase cm_{F_j} for each Fog node F_j or decrease $Bmax$ with respect to the equilibrium (13). This ensures that $RUminCut$ produces a solution where the constraint $B \leq Bmax$ is satisfied. Accordingly, we consider N=30 Fog nodes, and we set their computational capacity in terms of RAM so that 20% of the nodes have 256MB, 30% have 1GB and 50% have 2GB. Then we set $Bmax = 125000KB/s$ ($\approx 1Gbps$).

For a worst-case execution of SOO, the Fog computational resources should be more costly (i.e., relatively less abundant) than the Cloud network resources. To this end, we decrease cm_{F_j} for each Fog node F_j or increase $Bmax$ with respect to the equilibrium (13). However, we cannot go very far away from the equilibrium, otherwise $dataMinCut$ produces a solution where $B > Bmax$, i.e., there is no solution to the problem. Accordingly, we consider N=10 Fog nodes where all of them have 256MB of RAM, and we keep $Bmax = 125000KB/s$. Additionally for the baseline solution RCS, we set $Bmin = 75000KB/s$.

B. Evaluation results

1) Best-case execution: In this set of experiments, the Fog computational resources are relatively abundant and hence their usage is less costly than the usage of the Cloud network resources. As depicted in Figure 4, whatever the evolution of the number of IoT devices at the Edge and consequently the data stream rates reaching the Fog nodes, SOO performs like SOO-CPLEX, thus it finds the optimal RU thanks to $RUminCut$, for which the resulting solution satisfies directly the constraint $B \leq Bmax$. When comparing to RCS, SOO outperforms it with a difference ratio that decreases from 43.96% to 2.48% with respect to the evolution of the number of IoT devices at the Edge.

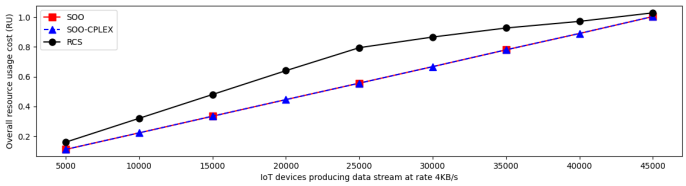

Fig. 4. Overall resource cost, N=30 Fog nodes while scaling the IoT devices

Figure 5 and Figure 6 show that SOO finds a balance between CRU and NRU in order to minimize RU. Given that RCS resorts minimally to the Fog resources, when $M \leq 20000$ IoT devices the resulting data stream rate that reaches the Cloud through the Fog nodes is lower than $Bmax$, hence no processing is moved to the Fog and CRU is null, as depicted in Figure 5. When comparing to SOO, such cost can be considered as cost-efficient, but the counterpart in terms of NRU is very high, as depicted in Figure 6. However when $M > 20000$ IoT devices, CRU for RCS is monotonically increasing for satisfying the constraint $B \leq Bmax$.

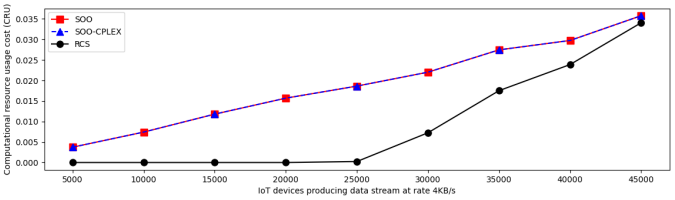

Fig. 5. Computational cost, N=30 Fog nodes while scaling the IoT devices

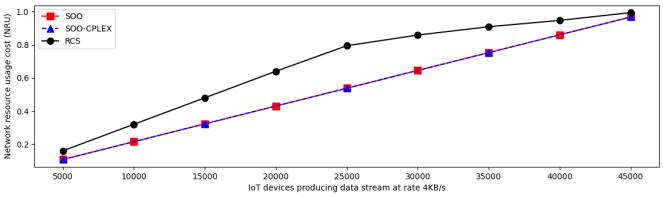

Fig. 6. Network cost, N=30 Fog nodes while scaling the IoT devices

Furthermore, we can see in Figure 5 that CRU values of SOO, SOO-CPLEX and RCS are very small, lower than 0.1. This is due to the specific computational requirements of the DSPA application (Figure 3) and the high capacities of the Fog computational resources.

2) Worst-case execution: In this set of experiments, the Fog computational resources are pretty limited, thus their usage is more

costly than the usage of the Cloud bandwidth. As shown in Figure 7, the RU plots are steeper for RCS, SOO and SOO-CPLEX, with very small differences between them. More specifically, SOO achieves a lower RU compared to RCS with a decreasing difference ratio from 14.77% to 0.76%, following the increase in the number of IoT devices (M) at the Edge, which results in an increase of the data stream rates arriving to the Fog nodes.

Furthermore, SOO achieves equal RU compared to SOO-CPLEX, for the experiments where $M < 40000$ IoT devices at the Edge. This corresponds to best-case execution for SOO, thus, SOO produces optimal results like SOO-CPLEX. On the other hand, when M is equal to 40000 and 45000 IoT devices, SOO-CPLEX outperforms SOO, with a small difference ratio up to 0.01%. This corresponds to worst-case execution for SOO, where SOO approximates the optimal solution.

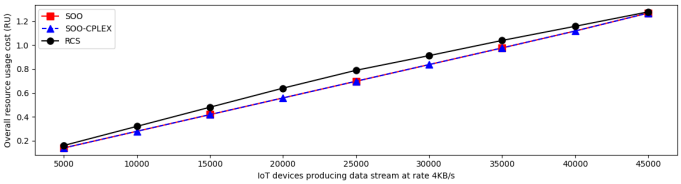

Fig. 7. Overall resource cost, N=10 Fog nodes while scaling the IoT devices

Figure 8 and Figure 9 show that, for RCS, CRU is null and constant and NRU monotonically increases when $M \leq 20000$ IoT devices, since RCS has the preference of using the Cloud network bandwidth over the Fog computational resources when $B \leq Bmax$. On the other hand, when M increases higher than 20000 IoT devices, CRU monotonically increases to ensure the constraint $B \leq Bmax$.

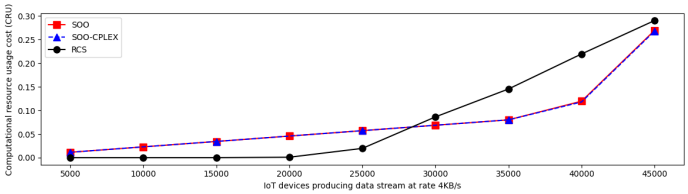

Fig. 8. Computational cost, N=10 Fog nodes while scaling the IoT devices

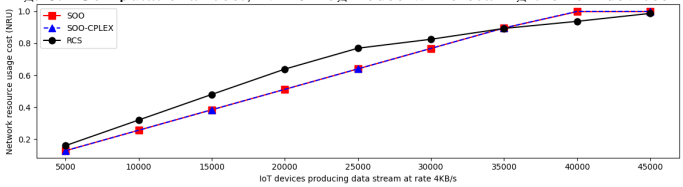

Fig. 9. Network cost, N=10 Fog nodes while scaling the IoT devices

For SOO and SOO-CPLEX, Figure 8 and Figure 9 show that SOO achieves to balance between CRU and NRU to minimize the overall resource usage cost RU.

3) Scalability analysis: We assess the scalabilty of SOO against SOO-CPLEX in terms of execution time, i.e., the time it takes for either algorithm to find the best streaming operator placement, to rewrite the application graph based on this placement, and to deploy the resulting application graph between the Fog and Cloud. In this respect, we use as starting point the parameter settings for SOO worst-case execution with $N = 10$ Fog nodes, $M = 40000$ IoT devices and $Bmax = 125000KB/s$. To maintain the same ratio in parameter setting with respect to the equilibrium (13), we consider the scaling up ratio $\frac{N}{N_{init}}$, where $N_{init} = 10$ Fog nodes, then we scale $Bmax$ and M by multiplying their initial values with the scaling up ratio for each increase of N, where initially $N = N_{init}$.

Besides the setting of SOO-CPLEX with $OG = 0.0\%$, we additionally consider the setting of SOO-CPLEX with $OG = 50\%$ to enable SOO-CPLEX to balance between optimality and execution time. As depicted in Figure 10, for the different scaling up ratios, SOO has lower execution times, with a difference ratio ranging between 54% and 70%, compared to SOO-CPLEX with $OG = 0\%$. Even when compared to SOO-CPLEX with $OG = 50\%$, SOO has lower execution times with a difference ratio between 1% and 30%.

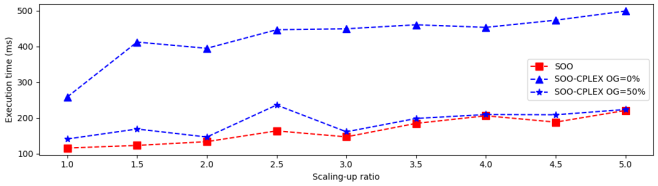

Fig. 10. execution time by by scaling N, Bmax and M

In terms of the produced RU solution, Figure 11 shows that SOO fails to find the optimal solution compared to SOO-CPLEX with $OG = 0\%$, but the approximation error is very small, ranging from 0.01% to 0.0003%, following the scaling-up ratio increase. On the other hand, when comparing to SOO-CPLEX with $OG = 50\%$, SOO produces a better RU solution whatever the scaling-up ratio.

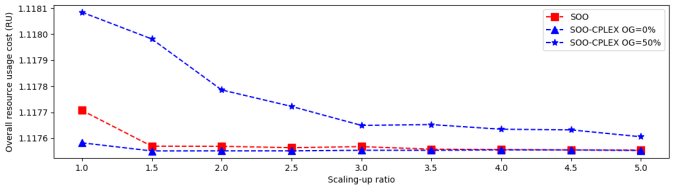

Fig. 11. Resource usage cost by scaling N, Bmax and M

V. Related Work

Algorithms [17], [18] solve the general resource allocation problem, where placement of operators is performed on peer resource nodes. These algorithms optimize some DSPA application performance metric (e.g. response time, availability, etc.) or some network-related metric (e.g. network bandwidth, network delay) and show a positive impact on application performance. Optimizing computational resource usage is not considered by these algorithms.

Neophytou et al. [21] propose the BOSe heuristic, which inspired our SOO heuristic. BOSe splits the processing load of continuous queries (CQs) between a central server and several mobile user devices in order to optimize the trade-off between data communication cost and data processing cost. These costs are expressed in terms of energy consumption of mobile devices. The results of the CQs take the form of individual data streams disseminated to the mobile devices over a shared broadcast medium. Similarly to SOO, BOSe relies on a greedy search of edge-cut moves to split the load of CQs. Unlike from SOO, in BOSe, moving an edge-cut of a data stream for a mobile user has an impact on the energy consumption of the other mobile users due to the broadcast medium; in SOO, dependency among streams is due to the shared Cloud bandwidth. Furthermore, BOSe does not consider a constraint on the computational resource usage of the mobile users, and it was not evaluated in terms of optimality.

Nardelli et al. [9] propose several heuristics to solve the operator placement problem with the objective to minimize response time, network resource usage, and to maximize the DSPA application availability. Some of these heuristics rely on meta-heuristics, such as tabu search, local search, etc. They use a penalty function that

captures the cost of using a computing resource. However, this set of heuristics does not optimize the computational resource usage. Furthermore, Hiessl et al. [24] extend the operator placement problem of [9] by including the operator enactment cost and operator migration cost. However, this work does not consider the optimization of the computational resource usage, and it does not propose any efficient heuristic. It is worth noting these approaches address the operator placement problem formulated by Cardellini et al. in [18] as an ILP problem and solved with CPLEX. This work also inspired our SOO-CPLEX solution.

Similarly to our approach, Planner [8] introduces a heuristic for deploying DSPA applications between the Cloud and the Edge/Fog in order to minimize the network resource usage for reaching the Cloud. Planner relies on a minimum edge-cut algorithm to split separately each data stream processing path.

Compared with Planner, we account for the fact that not only the network resources to reach the Cloud but also the computational resources at the Edge/Fog are limited and shareable, thus it is necessary to ensure their efficient usage; hence we propose a more sophisticated resource model and algorithmic approach.

Finally, SpanEdge [29] aims at reducing the response time and the network resource usage cost of data stream processing applications, however, it does not consider the computational resource usage. Additionally, it mainly proposes a software engineering approach, while it does not solve any instance of the operator placement problem.

VI. Conclusion

In this paper, we addressed the problem of scheduling streaming operators between Fog and Cloud nodes, in synergy with the variability of data stream rates produced at the Edge. We propose SOO, a resource-aware and time-efficient heuristic that takes into account the limited Fog computational resources, as well as congestion and delay issues on Fog-to-Cloud network resources. Evaluation results showed that SOO approximates the optimal resource usage cost solution while achieving excellent optimality, by managing the trade-off between the Fog computational resource usage cost and the Fog-to-Cloud network resource usage cost. Regarding scalability, SOO is time-efficient comparing to the optimal algorithmic solution, whatever the problem size. We are currently working on extending our problem formulation so as to take into account real-time response constraints of IoT applications. Additionally, as DSPA workloads may highly vary, we need to monitor at run-time the allocated resources and continuously schedule the deployment of DSPA computations over the Edge/Fog/Cloud.

References

[1] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, W. Zhao, "An edge computing based public vehicle system for smart transportation," IEEE Transactions on Vehicular Technology, 2020

[2] P. Dangal, G. Bloom, "Towards industrial security through real-time analytics," IEEE 23rd International Symposium on Real-Time Distributed Computing, 2020.

[3] Z. Lv, J. Lloret, H. Song, " Real-time image processing for augmented reality on mobile devices," Journal of Real-Time Image Processing, 2021.

[4] H. Röger, R. Mayer, "A comprehensive survey on parallelization and elasticity in stream processing," ACM Computing Surveys,2019.

[5] S. Li, L. Da Xu, S. Zhao, "5G Internet of Things: A survey, Journal of Industrial Information Integration," 2018.

[6] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," Internet of Things, 2018.

[7] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog," IEEE Internet of Things Journal, 2017.

[8] L. Prosperi, A. Costan, P. Silva, G. Antoniu, Planner: cost-efficient execution plans placement for uniform stream analytics on edge and cloud, IEEE/ACM Workflows in Support of Large-Scale Science, 2018.

[9] M. Nardelli, V. Cardellini, V. Grassi F. L. Presti, Efficient operator placement for distributed data stream processing applications, IEEE Transactions on Parallel and Distributed Systems, 2019.

[10] C. H. Hong B. Varghese, Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms, ACM Computing Surveys, 2019.

[11] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen M. Satyanarayanan, Quantifying the impact of edge computing on mobile applications, 7th ACM SIGOPS Asia-Pacific Workshop on Systems,2016.

[12] S. Katsikeas, K. Fysarakis, A. Miaoudakis, A. Van Bemten, I. Askoxylakis, I. Papaefstathiou, A. Plemenos, Lightweight secure industrial IoT communications via the MQ telemetry transport protocol, IEEE Symposium on Computers and Communications, , 2017.

[13] F. Xhafa, B. Kilic, P. Krause, Evaluation of IoT stream processing at edge computing layer for semantic data enrichment, Future Generation Computer Systems, 2020.

[14] A. Tiwari, B. Ramprasad, S. H. Mortazavi, M. Gabel, E. D. Lara, Reconfigurable streaming for the mobile edge, 20th International Workshop on Mobile Computing Systems and Applications,2019.

[15] P. Ntumba, N. Georgantas, V. Christophides, Scheduling Continuous Operators for IoT Edge Analytics, 4th International Workshop on Edge Systems, Analytics and Networking,2021.

[16] Studio-CPLEX, Users manual-version 12 release 6. IBM ILOG CPLEX Division: Incline Village, NV, USA, 2013.

[17] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, M. Seltzer, Network-aware operator placement for stream-processing systems, 22nd International Conference on Data Engineering, 2006.

[18] V. Cardellini, V. Grassi, F. Lo Presti M. Nardelli, Optimal operator placement for distributed stream processing applications, 10th ACM International Conference on Distributed and Event-based Systems,2016.

[19] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, Software: Practice and Experience, 2017.

[20] H. M. Salkin, C. A. De Kluyver, The knapsack problem: a survey. Naval Research Logistics Quarterly, 1975.

[21] P. Neophytou, M. A. Sharaf, P. K. Chrysanthis, A. Labrinidis, Power-aware operator placement and broadcasting of continuous query results, 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access,2010.

[22] J. Edmonds, R. M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM, 1972.

[23] B. Donassolo, I. Fajjari, A. Legrand P. Mertikopoulos, Fog based framework for IoT service provisioning, 16th IEEE Annual Consumer Communications Networking Conference, 2019.

[24] T. Hiessl, V. Karagiannis, C. Hochreiner, S. Schulte M. Nardelli, Optimal placement of stream processing operators in the fog, 3rd International Conference on Fog and Edge Computing, 2019.

[25] L. Li, M. Guo, L. Ma, H. Mao Q. Guan, Online workload allocation via fog-fog-cloud cooperation to reduce IoT task service delay, Sensors, 2019.

[26] R. Deng, R. Lu, C. Lai, T. H. Luan H. Liang, Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption, internet of things journal, 2016.

[27] F. R. de Souza, M. D. de Assunçao, E. Caron, A throughput model for data stream processing on fog computing, International Conference on High Performance Computing Simulation, 2019.

[28] P. Silva, A. Costan, G. Antoniu, Investigating edge vs. cloud computing trade-offs for stream processing, International Conference on Big Data, 2019.

[29] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, V. Vlassov, Spanedge: Towards unifying stream processing over central and near-the-edge data centers, IEEE/ACM Symposium on Edge Computing,2016.