A Variable Neighborhood Heuristic for Facility Locations in Fog Computing

Thiago Alves de Queiroz[1][0000−0003−2674−3366],
Claudia Canali[2][0000−0001−8448−7693],
Manuel Iori[3][0000−0003−2097−6572], and
Riccardo Lancellotti[2][0000−0002−9470−8784]

[1] Institute of Mathematics and Technology, Federal University of Catalão, 75704-020, Catalão-GO, Brazil, taq@ufg.br.
[2] Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125, Modena, Italy, {claudia.canali, riccardo.lancellotti}@unimore.it.
[3] Department of Science and Methods for Engineering, University of Modena and Reggio Emilia, 42122, Reggio Emilia, Italy, manuel.iori@unimore.it.

Abstract. The current trend of the modern smart cities applications towards a continuous increase in the volume of produced data and the concurrent need for low and predictable latency in the response time has motivated the shift from a cloud to a fog computing approach. A fog computing architecture is likely to represent a preferable solution to reduce the application latency and the risk of network congestion by decreasing the volume of data transferred to cloud data centers. However, the design of a fog infrastructure opens new issues concerning not only how to allocate the data flow coming from sensors to fog nodes and from there to cloud data centers, but also the choice of the number and the location of the fog nodes to be activated among a list of potential candidates. We model this facility location issue through a multi-objective optimization problem. We propose a heuristic based on the variable neighborhood search, where neighborhood structures are based on swap and move operations. The proposed method is tested in a wide range of scenarios, considering a smart city application's realistic setup with geographically distributed sensors. The experimental evaluation shows that our method can achieve stable and better performance concerning other literature approaches, supporting the given application.

Keywords: Smart cities · Fog networking · Facility location problem

1 Introduction

Smart city applications that require the processing of huge volumes of data produced by geographically distributed sensors represent a typical scenario where fog computing is likely to be a winning approach. Its potential has been demonstrated, indeed, by several studies in literature [6,16–18]. The main characteristic of fog computing is the ability to push on the network edge functions such as data filtering and aggregation [17,18], with a twofold advantage. First, it reduces the data volume reaching the cloud data center, thus avoiding the risk of poor performance due to high network utilization and reducing

the non-negligible economic costs related to the cloud pricing model. Second, the fog layer located on the network edge can guarantee a fast response to latency-sensitive applications (e.g., traffic monitoring and support for autonomous driving) that cannot accept delays in the order of hundreds of milliseconds due to the potentially high round-trip-time latency with the cloud data center.

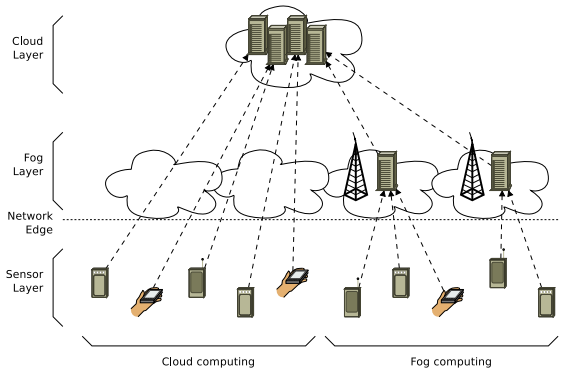


Fig. 1: Cloud and fog infrastructures.

In Figure 1, we compare fog and cloud approaches. In the cloud architecture (left part of the figure), a set of sensors sends data directly to the cloud data center for processing. In the fog case (right part of the figure), a layer of fog nodes is placed on the network edge to host pre-processing, filtering, and aggregation tasks.

The introduction of the intermediate layer of fog nodes represents an additional degree of freedom that arises new issues for the overall infrastructure design. In particular, many studies [6, 19] consider just the fog to cloud communication, adopting a naive approach in the allocation (i.e., mapping) of data flows coming from the sensors over the fog layer, assuming that every sensor sends data to the nearest fog node. On the other hand, recent studies demonstrated that optimized data flow allocation could provide a significant advantage [3]. However, even when some optimization is performed in the sensors-to-fog mapping, no attention has been devoted to minimizing the number of fog nodes required to satisfy the Service Level Agreement (SLA) to reduce the costs and the energy consumption related to the management of the fog infrastructure.

In the operational research field, this issue is named facility location problem [7, 8] and concerns the identification of facilities, so the costs incurred from allocating customers to the selected facilities are minimized, and represents nowadays a very active field of research. A recent survey on service facility location problems can be found in [5]. While this aspect has been widely explored at the level of managing resources in a cloud data center [1, 12], it has not been considered so far in the fog computing field.

In this paper, we formalize the facility location problem through an optimization model aiming to map sensors data flow over the fog layer with a two-fold objective: minimize the number of turned on fog nodes while guaranteeing the respect of a service level agreement on response time; and, minimize the response time for the given

number of selected fog nodes. In the model, we consider both network delays and processing time at the level of fog nodes. Furthermore, a qualifying point of this study is the development of a heuristic, based on the Variable Neighborhood Search (VNS) [9], to solve the facility location problem over a fog computing infrastructure.

As pointed out in [9], the VNS can be seen as a framework for building heuristics to solve different optimization problems. In the recent survey of [15], it was discussed how the VNS had been successfully applied to solve problems in reverse logistics and closed-loop supply chain networks. Concerning facility location related problems, a VNS with path-relinking was proposed in [20] for the location routing problem, where neighborhood structures based on insertion, swap, 2-opt, and CROSS-exchange moves were used. Recently, a basic variable neighborhood search, based on the less is more concept, was developed in [13] for an obnoxious p-median problem. The obnoxious effect occurs when a facility should be located as far as possible from an inhabited center.

We evaluate the proposed VNS in the realistic scenario of a smart city application with geo-referenced sensors collecting data for traffic monitoring in the city of Modena in Italy. The VNS is compared with the optimization model solved by a state-of-the-art solver in terms of its capability of reducing costs and response times. Moreover, a sensitivity analysis is performed concerning the infrastructure size. The experimental results demonstrate that our proposal can outperform the alternatives with stable results concerning a wide range of scenarios.

The rest of the paper is organized as follows. Section 2 presents the theoretical modeling for the considered problem. Section 3 presents the VNS to solve the facility location problem in fog computing infrastructures. Section 4 presents the experimental setup and the considered scenarios, providing a thorough evaluation of the proposed model against the alternatives. Finally, Section 5 gives some concluding remarks and outlines some future work direction.

2 Problem Definition

In this section, we formalize the location-allocation problem in a fog architecture as a multi-objective optimization problem that aims to minimize two aspects: 1) the delay in the transit of the data from sensors to fog nodes to the cloud data center; 2) the cost associated with the number of fog nodes turned on.

To model this problem, let us assume a stationary scenario with a set $\mathcal{S}$ of geographically distributed sensors producing data at a steady rate: we denote as λ_i the frequency of sensor i. The data are sent to an intermediate layer of a set $\mathcal{F}$ of fog nodes. These nodes can perform operations on data such as filtering and aggregation or specific analysis, such as anomaly detection with low latency. We denote as μ_j the processing rate of the fog node j, and as δ_{ij} the delay from sensor i to fog node j. The model also includes a set $\mathcal{C}$ of cloud data centers that receive data from the fog nodes, where δ_{jk} represents the delay from fog node j to cloud data center k. To formalize the problem, we use the following binary decision variables: a) x_{ij}, indicating whether sensor i sends data to fog node j; b) y_{jk}, indicating whether fog node j sends data to cloud data center k; c)

E_j, defining whether the fog node located at position j is turned on and available to process data from sensors. The main symbols of the model are summarized in Table 1.

Table 1: Notation and parameters for the proposed model.

Model parameters	
$\mathcal{S}$	Set of sensors
$\mathcal{F}$	Set of fog nodes
$\mathcal{C}$	Set of cloud data centers
λ_i	Outgoing data rate from sensor i
λ_j	Incoming data rate at fog node j
$1/\mu_j$	Processing time at fog node j
δ_{ij}	Communication latency between sensor i and fog node j
δ_{jk}	Communication latency between fog j and cloud k
c_j	Cost for locating a fog node at position j (or for keeping the fog node turned on)
Model indices	
i	Index for a sensor
j	Index for a fog node
k	Index for a cloud data center
Decision variables	
E_j	Location of fog node j
x_{ij}	Allocation of sensor i to fog node j
y_{jk}	Allocation of fog node j to cloud data center k

For the problem of sensors allocation on the fog nodes, introduced in [3], we consider the application average response time T_R, which depends on three components (Eq. (1)): the network delay due to the sensor-to-fog latency T_{netSF} (Eq. (2)), the network delay due to the fog-to-cloud latency T_{netFC} (Eq. (3)), and the processing time on the fog nodes T_{proc} (Eq. (4)).

$$T_R = T_{netSF} + T_{netFC} + T_{proc} \tag{1}$$

$$T_{netSF} = \frac{1}{\sum_{i \in \mathcal{S}} \lambda_i} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} \lambda_i x_{ij} \delta_{ij} \tag{2}$$

$$T_{netFC} = \frac{1}{\sum_{j \in \mathcal{F}} \lambda_j} \sum_{j \in \mathcal{F}} \sum_{k \in \mathcal{C}} \lambda_j y_{jk} \delta_{jk} \tag{3}$$

$$T_{proc} = \frac{1}{\sum_{j \in \mathcal{F}} \lambda_j} \sum_{j \in \mathcal{F}} \lambda_j \frac{1}{\mu_j - \lambda_j} \tag{4}$$

In the components T_{netSF} and T_{netFC}, each delay is weighted by the amount of traffic exchanged on the corresponding link, which is λ_i for T_{netSF} and λ_j in T_{netFC}.

The incoming data rate on each fog node λ_j, indeed, can be defined as the sum of the data rates of the sensors allocated to that node:

$$\lambda_j = \sum_{i \in \mathcal{S}} x_{ij} \lambda_i, \quad \forall j \in \mathcal{F} \tag{5}$$

The component concerning the processing time T_{proc} is modeled using the queuing theory considering an M/G/1 system and is consistent with other results in literature [1, 4]. Specifically, the generic fog node j is characterized by an average processing time $1/\mu_j$ and receives an incoming stream of jobs with frequency λ_j (where λ_j is defined as in Eq. (5)). It is worth mentioning that we do not consider the cloud layer's details in our problem modeling, such as the computation time at the cloud data center level, as this aspect has been widely covered in literature [2].

Finally, we consider that a fixed cost c_j is associated with the fog node j if it is turned on to model the overall cost due to the fog node activation. In our model, we do no consider other constraints, such as the amount of memory required by the fog nodes' application. Such additional constraints can be straightforwardly added to the model. However, in our experience, the most critical constraint for fog infrastructures deployment is the processing power rather than the available memory. Hence, we opted for a more streamlined model. The complete model for the fog node location-allocation problem may be defined as follows.

Minimize:

$$C = \sum_{j \in \mathcal{F}} c_j E_j \tag{6}$$

$$T_R = T_{netSF} + T_{netFC} + T_{proc} \tag{7}$$

Subject to:

$$T_R \leq T_{SLA} \tag{8}$$

$$\lambda_j < E_j \mu_j, \quad \forall j \in \mathcal{F} \tag{9}$$

$$\sum_{j \in \mathcal{F}} x_{ij} = 1, \qquad \forall i \in \mathcal{S} \tag{10}$$

$$\sum_{k \in \mathcal{C}} y_{jk} = E_j, \qquad \forall j \in \mathcal{F} \tag{11}$$

$$E_j \in \{0, 1\}, \quad \forall j \in \mathcal{F} \tag{12}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \mathcal{S}, j \in \mathcal{F} \tag{13}$$

$$y_{jk} \in \{0, 1\}, \quad \forall j \in \mathcal{F}, k \in \mathcal{C} \tag{14}$$

The two objective functions, (6) and (7), are related, respectively, to the minimization of: **cost**, which is associated with the number of fog nodes turned on; and, **latency**, which is the delay in sensor to fog to cloud data transit expressed through the function introduced as T_R in (1). The second objective is subordinated to the first one, meaning that we aim to minimize (7) as long as the improvement for this objective function does not introduce an increase for (6).

The model includes several constraints. Constraint (8) introduces a limit for the average response time that should not exceed a *Service Level Agreement* (SLA), which is typically defined as a multiple of the average response time $1/\mu$ [1]. We also introduce a term due to the network delays in a distributed architecture (non-negligible) that we consider depending on the average network delays δ. The SLA limit in (15) can be formalized as follows, with K defined depending on the network requirements:

$$T_{SLA} = \frac{K}{\mu} + 2\delta \tag{15}$$

Constraints (9) ensure that no overload occurs on each fog node, imposing that the incoming data flow does not exceed the processing rate. For a node that is powered down, no processing must occur. Constraints (10) guarantee for each sensor that one fog node processes its data, while constraints (11) ensure for each fog node that exactly one cloud data center receives its processed data. Constraints (12), (13) and (14) describe the domain of the decision variables.

3 Variable Neighborhood Heuristic

A *Variable Neighborhood Search* (VNS) is proposed to solve the facility location problem that arises in fog computing infrastructures. The VNS methodology was initially proposed in [14]. It has a systematical change of neighborhoods to look for a globally optimal solution concerning all neighborhoods.

The VNS for the problem has the following main steps: to create an initial solution, which is the current solution; to obtain a neighbor solution of the current solution by using a neighborhood operator (shake-phase); to perform a local search on the neighbor solution (local-search-phase); and, to accept the solution of the local search if it is better than the current solution, besides updating the current solution. If the current solution is updated, the VNS restarts from the first neighborhood; otherwise, it proceeds to the next neighborhood, repeating the steps above until reaching the last neighborhood [9, 10].

In the proposed VNS, a solution x is coded as a vector of lists of integers. Each position of the vector represents a fog node and contains a list of integers with the sensors. There is another integer indicating which cloud data center is serving the fog node. A position with an empty list of integers represents a fog node turned off. Notice the vector has size $|\mathcal{F}|$, and each list of integers has size at most $|\mathcal{S}|$. Moreover, the lists of integers have a null intersection since a sensor is serviced by exactly one fog node.

The initial solution of the VNS is created as follows. For each fog node, we select the closest cloud data center to allocate it. For each sensor, we choose the closest fog node to assign it. If a fog node has already reached the T_{SLA}, no other sensor can be allocated to it. It means the second closest fog node is used to allocate the sensor, and so on until all sensors are allocated to fog nodes. Once the initial solution is created, it has its two objectives calculated: (i) the cost associated with the number of fog nodes turned on; and (ii) the delay in sensor to fog to cloud transit of data, where the first objective is used to guide the VNS. We do not accept solutions that violate constraints (8) to (11) in the optimization process. A solution x' is better than another x'' if (i) the

first objective of x' is smaller than that of x'', or if (ii) the two first objectives are equal but the second objective of x' is smaller than the second of x''.

As the VNS iterates through K neighborhood structures, we propose five structures based on swap and move operations. In particular:

- N_1: select (randomly) a fog node f_1, the farthest sensor s_1 allocated to f_1, the fog f_2 that is the closest to s_1, and the sensor s_2 allocated to f_2 that is the closest to f_1. Hence, swap s_1 and s_2 from their respective fog nodes, if this new solution is feasible.
- N_2: Let $\mathcal{F}_{on}$ be the set of fog nodes turned on. Compute the load of each fog node $j \in \mathcal{F}_{on}$ as $r_j = \lambda_j/\mu_j$ and, then, the average load of fog nodes turned on as: $\bar{r} = \left(\sum_{j \in \mathcal{F}_{on}} r_j\right) / |\mathcal{F}|$. Select (randomly) $f_1 \in \mathcal{F}_{on}$ whose load $r_1 > \bar{r}$. If one exists, select the farthest sensor s_1 allocated to f_1. Then, select the fog node $f_2 \in \mathcal{F}_{on}$ with the lowest load r_2 and closest to s_1. Remove s_1 from f_1 and move it to f_2, if this new solution is feasible.
- N_3: Let $\mathcal{F}_{on}$ be the set of fog nodes turned on. Select (randomly) a fog node $f_1 \in \mathcal{F}_{on}$. Compute the average load with all sensors and fog nodes turned on, except f_1, as: $\tilde{r} = \left(\sum_{i \in \mathcal{S}} \lambda_i\right) / \left(\sum_{j \in \mathcal{F}_{on} \setminus \{f_1\}} \mu_j\right)$. If $\tilde{r} < 1$, then for each sensor s_1 allocated to f_1, remove s_1 from f_1 and move it to the closest fog node in $\mathcal{F}_{on} \setminus \{f_1\}$, if this new solution is feasible.
- N_4: Let $\mathcal{F}_{on}$ be the set of fog nodes turned on. Let $\mathcal{F}_{off}$ be the set of fog nodes turned off. If $\mathcal{F}_{off}$ is not empty, select (randomly) a fog node $f_1 \in \mathcal{F}_{off}$. Select the fog node $f_2 \in \mathcal{F}_{on}$ whose average response time is the highest one. Remove all sensors from f_2 and move them to f_1, then turning off f_2, if this new solution is feasible.
- N_5: if the number of available cloud data centers is greater than one, select (randomly) a fog node turned on and allocate it to the closest cloud data center, if this new solution is feasible.

Regarding the shake-phase of the VNS, the selection of fog nodes in the neighborhood structures occurs randomly if the contrary is not imposed. On the other hand, in the local-search-phase, we use a variable neighborhood descent based procedure, in which the solution from the shake-phase is passed as the input parameter [9]. In this procedure, two neighborhood structures are used. It consists of performing all possible (i) allocations of sensors in fog nodes and (ii) swaps of sensors in fog nodes. The procedure restarts from the first structure whenever the current solution is improved and continues until no improved solution can be achieved.

4 Experimental Results

We discuss next the performance of the VNS by evaluating it on a realistic scenario of fog computing. We start this section with the description of the experimental setup, and we proceed with the comparison between the performance of the proposed VNS and other alternatives.

4.1 Experimental scenario

As a realistic fog computing scenario, we refer to a smart city project developed into the medium-sized Italian city of Modena (around 180.000 inhabitants). We consider a smart city application for monitoring car and pedestrian traffic where geographically distributed sensors collect information comprising data from proximity sensors and possibly low-resolution images. In our scenario, sensors are wireless devices located in the city's main streets: the location of the sensors is obtained by geo-referencing the selected streets. The sensors send the collected data to the fog nodes, which in turn perform pre-processing tasks by filtering the proximity sensor readings and, if available, analyze images from the camera to detect cars and pedestrians. We assume the fog nodes to be located in government buildings. The pre-processed data are then sent to a cloud data center located on the municipality premises.

In the performed experiments, we consider sensors equipped with long-range wireless connectivity, for example, LoRaWAN[4] or IEEE 802.11ah/802.11af [11]. Hence, the sensors can potentially connect to every fog node: we assume that the network delay depends on the physical distance between two nodes as in [3,4], due to the growing delay and decreasing bandwidth limitations as the distance from a sensor to the fog node increases. Specifically, we model the communication latency through the Haversine distance, starting from a given latitude and two locations' longitude.

In the experimental evaluation, we consider scenarios of different sizes to understand the proposed method's capability to scale with growing numbers of sensors and fog nodes. Specifically, we consider a number of sensors $|\mathcal{S}| \in \{50, 100, 200\}$, and a number of fog nodes $|\mathcal{F}| \in \{5, 10, 20\}$.

We consider different scenarios, each of them described by three main parameters. First, the *sensor data rate* λ. Based on a preliminary evaluation of smart city applications for traffic monitoring, we consider that each sensor provides a reading every 10 seconds; hence, the data rate $\lambda_i = 0.1, \forall i \in \mathcal{S}$. Second, the *average utilization* of the system ρ, that can be defined as $\frac{\sum_{i \in \mathcal{S}} \lambda_i}{\sum_{j \in \mathcal{F}} \mu_j}$. For this parameter, we consider a wide range of values: $\rho \in \{0.1, 0.2, 0.5, 0.8, 0.9\}$. For each value of ρ, considering sensors and fog nodes homogeneous and knowing the value of λ_i, we derive the value of $\mu_j = \mu$, which is assumed the same for each $j \in \mathcal{F}$. Third, the parameter $\delta\mu$, defining the *CPU-bound or network-bound nature* of the scenario and expressed as the ratio between the average network delay δ and the average service time of a request ($1/\mu$). For this parameter we consider values ranging multiple orders of magnitude: $\delta\mu \in \{0.01, 0.1, 1, 10\}$. In this way, we can explore both CPU-bound scenarios (e.g., when $\delta\mu = 0.01$), where computing time is much higher than transmission time, and network-bound cases (e.g., when $\delta\mu = 10$). We derive the average network delay from the $\delta\mu$ parameter and the previously computed parameter μ_j. It is worth noticing that, even if our analysis may consider very high network delays, these scenarios can still be considered realistic if we consider that the network contribution may involve the transfer of images over low-bandwidth links.

The evaluation of the proposed model considers a wide range of different scenarios related to the introduced parameters. Each scenario is named according to the format

[4] https://lora-alliance.org/

ins-ρ-$\delta\mu$ (e.g., instance ins-0.1-0.01 indicates the scenario with $\rho = 0.1$ and $\delta\mu = 0.01$). Moreover, for the SLA in Eq. (15), the constant K is set to 10, which is a typical value in the literature [1]. Finally, we assume the cost c_j of a fog node at position j equal to 1, for all $j \in \mathcal{F}$. This means that the fog nodes are equal from the operating cost point of view, and the objective function will try to reduce the overall number of active nodes. For the experimental comparison, we evaluate the following three models:

- *Simplified model (SM)*: this is the simplified version of the problem described in Section 2 and presented for the first time in [3]. In this model, all fog nodes are assumed on, that is $E_j = 1, \forall j \in \mathcal{F}$. The energy consumption may be high, but the infrastructure provides good performance from a response time point of view (Eq. (7));
- *Complex model (CM)*: this is the problem described in Section 2 that aims to minimize both energy consumption in Eq. (6) and response time in Eq. (7);
- *Proposed model (VNS)*: this is the heuristic introduced in this study and described in Section 3;

For the Simplified and Complex models' numeric results, we rely on LocalSolver[5] version 9.0, with a time limit of 300 seconds (5 minutes) as a stopping criterion. LocalSolver is a general mathematical programming solver that hybridizes local and direct search, constraint propagation and inference, linear and mixed-integer programming, and nonlinear programming methods. It can handle multi-objective problems, where the objectives are optimized in the order of their declaration in the model. For the sake of fairness, we run the proposed VNS heuristic for 300 seconds or 3000 iterations (the first to reach stops the VNS).

4.2 Performance evaluation

The comparison between the models performance considers two main metrics: the cost related to the number of turned on fog nodes, namely Obj_1, corresponding to Eq. (6); and, the average response time, Obj_2, corresponding to Eq. (7). To facilitate the comparison between models, we also consider a deviation measure expressing the performance of a model with respect to an alternative one. Specifically, the deviation function of a model $M1$ with respect to a model $M2$ is considered for each objective function (Obj_1 and Obj_2), which is defined as:

$$\epsilon(Obj_1^{M1}) = \frac{Obj_1^{M1} - Obj_1^{M2}}{Obj_1^{M2}} \tag{16}$$

$$\epsilon(Obj_2^{M1}) = \frac{Obj_2^{M1} - Obj_2^{M2}}{Obj_2^{M2}} \tag{17}$$

In Table 2, we present the complete results for the scenario with 100 sensors and 10 fog nodes since the others follow the same tendency. The table also shows the number of iterations required by LocalSolver and VNS to reach the reported values. Despite that, we focus the analysis on the deviation measure previously introduced to compare the

Table 2: Results for 100 sensors and 10 available fog nodes.

	Simplified Model			Complex Model (Dev. CM vs. SM)					VNS (Dev. VNS vs. CM)				
Instances	Iter.	Obj_1	Obj_2	Iter.	Obj_1	Dev. (%)	Obj_2	Dev. (%)	Iter.	Obj_1	Dev. (%)	Obj_2	Dev. (%)
ins-0.1-0.01	23655	10	0,1163	52421	2	-80,00	0,2337	100,96	1	2	0,00	0,2332	-0,23
ins-0.1-0.1	31809	10	0,1544	50876	2	-80,00	0,5520	257,45	1	2	0,00	0,5305	-3,90
ins-0.1-1	29173	10	0,5219	61189	2	-80,00	3,7795	624,22	1	2	0,00	3,2555	-13,86
ins-0.1-10	36088	10	4,1912	31853	6	-40,00	8,6976	107,52	1	3	-50,00	17,9568	106,46
ins-0.2-0.01	26833	10	0,2613	25242	3	-70,00	0,6482	148,07	1	3	0,00	0,6443	-0,61
ins-0.2-0.1	19049	10	0,3429	30661	3	-70,00	1,0125	195,30	6	3	0,00	1,0125	0,00
ins-0.2-1	28671	10	1,0829	33141	3	-70,00	4,9492	357,05	4	3	0,00	4,5140	-8,79
ins-0.2-10	38641	10	8,4215	46185	3	-70,00	45,6711	442,31	1	3	0,00	38,9263	-14,77
ins-0.5-0.01	39481	10	1,0300	13903	6	-40,00	3,1153	202,46	1	6	0,00	3,1148	-0,01
ins-0.5-0.1	24610	10	1,2825	15566	6	-40,00	3,5829	179,37	176	6	0,00	3,5344	-1,35
ins-0.5-1	21598	10	3,3132	7802	7	-30,00	5,9867	80,70	86	6	-14,29	8,1437	36,03
ins-0.5-10	25093	10	21,9581	10851	7	-30,00	34,4636	56,95	315	6	-14,29	44,9171	30,33
ins-0.8-0.01	52087	10	4,0480	11032	9	-10,00	8,3199	105,53	40	9	0,00	8,3160	-0,05
ins-0.8-0.1	51989	10	4,4799	14790	9	-10,00	8,8266	97,03	295	9	0,00	8,7628	-0,72
ins-0.8-1	38901	10	8,7654	14729	9	-10,00	13,1785	50,35	305	9	0,00	13,2132	0,26
ins-0.8-10	32297	10	44,1912	7335	9	-10,00	60,2917	36,43	455	9	0,00	63,1833	4,80
ins-0.9-0.01	57507	10	9,0540	11832	10	0,00	9,0540	0,00	16	10	0,00	9,0540	0,00
ins-0.9-0.1	45581	10	9,5399	15801	10	0,00	9,5399	0,00	20	10	0,00	9,5399	0,00
ins-0.9-1	54009	10	14,3987	10055	10	0,00	14,3987	0,00	16	10	0,00	14,3987	0,00
ins-0.9-10	50609	10	62,9869	12502	10	0,00	62,9869	0,00	50	10	0,00	62,9869	0,00

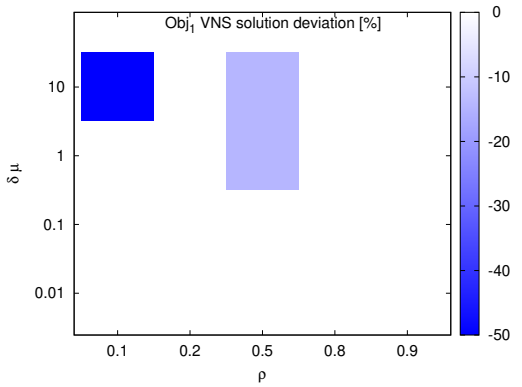
models. Specifically, we consider the CM model's deviation to the SM and the deviation of the VNS to the CM.

In comparing the CM model with the SM, we note that the differences strongly depend on ρ. On the one hand, the SM model uses all the available fog nodes, even if, especially when ρ is low, the processing of sensors data may require just a fraction of the computational infrastructure power. On the other hand, the system load has a significant impact on the number of fog nodes used by the CM model. For low values of ρ, indeed, the deviation on Obj_1 shows a reduction of the costs related to the activated fog nodes up to 80%. A higher number of active fog nodes can provide lower response time, as testified by the CM's positive deviation versus the SM on Obj_2. While in the SM model, we have an abundance of computational power due to all fog nodes' use, the CM uses the minimum amount of resources to satisfy the SLA constraint to reduce costs effectively.

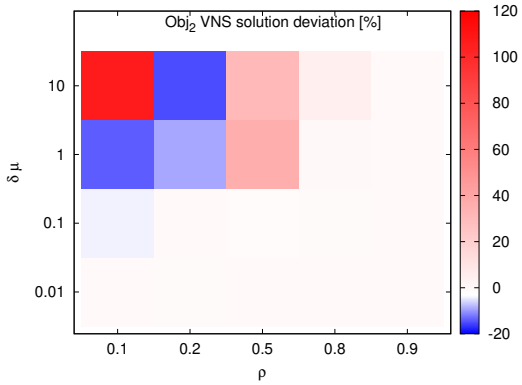
Figures 2a and 2b have the deviation of the VNS with respect to the CM model in terms of cost ($\epsilon(Obj_1^{VNS})$ and response time ($\epsilon(Obj_2^{VNS})$), respectively. Data are represented as heat maps, with red hues when the solution performs worse than the CM model, white color when the performances are similar, and blue hues in the opposite case.

Focusing on $\epsilon(Obj_1^{VNS})$ in Fig. 2a, we observe how the VNS achieves equal or better performance for the CM model in every considered scenario. In all white areas of the chart, the number of nodes used by the VNS is the same concerning the CM model (see third and fourth columns of Table 2). Moreover, we observe that in some Network-bound scenarios (e.g., for $\rho = 0.1$ and $\delta\mu = 1$ as well as $\rho = 0.5$ and $\delta\mu \in \{1, 10\}$)

[5] http://www.localsolver.com



(a) $\epsilon(Obj_1^{VNS})$

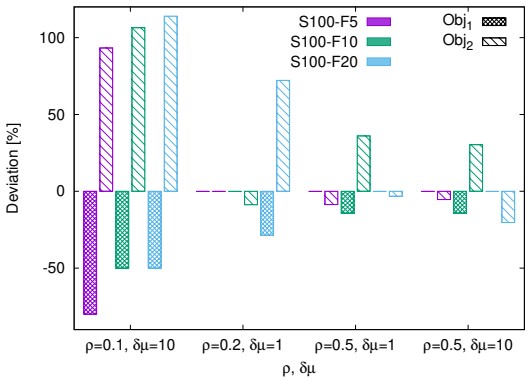


(b) $\epsilon(Obj_2^{VNS})$

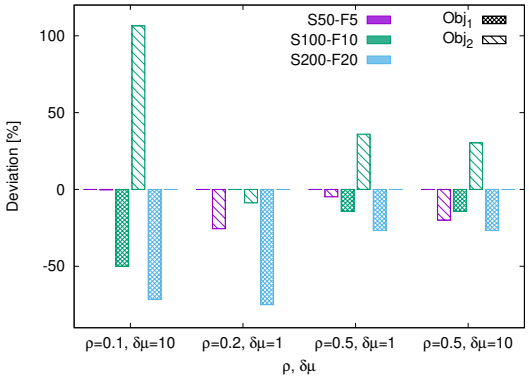
Fig. 2: Deviation between the VNS and CM model.

the VNS allows a further reduction of fog nodes concerning the CM model, which decreases the costs up to the 50% in case of low system load ($\rho = 0.1$). The results of $\epsilon(Obj_2^{VNS})$ in Fig. 2b show that the VNS can reduce the response times in scenarios where the number of turned on fog nodes is the same as the CM model (blue areas in Fig. 2b corresponding to white areas in Fig. 2a) thanks to a more optimized mapping between sensors data flow and fog nodes. This can be explained by the fact that the CM, in the time limit of 300 seconds, could not reach such an optimized mapping as the proposed VNS. On the other hand, we observe some red areas corresponding to the scenarios where the number of fog nodes activated by the VNS is lower concerning the CM model: in this case, as expected, the response time increased, but it remains within the defined SLA. The effect is particularly evident for the scenario with $\rho = 0.1$ and $\delta\mu = 1$, where the VNS presents an increase in response times. Still, it can achieve a very significant reduction (50%) of the required fog nodes.

We also present two sensitivity evaluations of the proposed VNS. First, we consider a varying number of sensors $|\mathcal{S}|$ while keeping constant the number of fog nodes $|\mathcal{F}| = 100$ in Fig. 3a. Second, we evaluate how the performance changes for different sizes of scenarios, keeping constant the ratio between sensors and fog nodes ($|\mathcal{S}| = 10 \cdot |\mathcal{F}|$) and varying the number of sensors ($|\mathcal{S}| \in \{50, 100, 200\}$) in Fig. 3b.



(a) Sensitivity to the number of fog nodes $|\mathcal{F}|$.



(b) Sensitivity to number of sensors $|\mathcal{S}|$ and fog nodes $|\mathcal{F}|$.

Fig. 3: Sensitivity analysis of the VNS.

The results of the first sensitivity analysis are presented in Fig. 3a. They show the deviation of the VNS concerning the CM model in terms of costs (Obj_1) and response time (Obj_2) for different numbers of fog nodes, where $|\mathcal{F}|$ varies from 5 to 10 up to 20, and the number of sensors $|\mathcal{S}|$ is fixed to 100. For each case, we evaluate scenarios defined by different values of ρ and $\delta\mu$. These parameters' values have been chosen among the more interesting points that emerged in the previous analysis. These are the points where the VNS behaves differently from the CM model. We observe that for a network-bound scenario with low system load ($\rho = 0.1$ and $\delta\mu = 10$), the VNS can

significantly reduce the costs due to turn on fog nodes for every number of sensors, with an increase of the response times that remain within the SLA. As ρ increases, we note a different behavior depending on the number of fog nodes available. If this number is low ($|\mathcal{F}| = 5$), the VNS activates all the fog nodes as the CM model; however, it achieves in some cases ($\rho = 0.5$) a reduction of the response time thanks to better mapping of sensor data flows on the fog nodes. For a higher number of fog nodes ($|\mathcal{F}| = 10, 20$), the VNS can reduce the number of required fog nodes with a low increase in the average response time.

The second sensitivity analysis is presented in Fig. 3b, where we consider systems with the ratio between the number of sensors and fog nodes fixed to 10. This analysis confirms the conclusions of the previous one, which is our proposal's stability concerning different scenarios. Also, in this case, the most significant gain in terms of required fog nodes is achieved for low system loads ($\rho \in \{0.1, 0.2\}$) and bigger sizes of the infrastructure ($|\mathcal{F}| \in \{10, 20\}$). In the smallest scenario, ($|\mathcal{S}| = 50$ and $|\mathcal{F}| = 5$), it is interesting to note that, in three scenarios out of four, the VNS can reduce the response time while using the same number of fog nodes concerning the CM model. Furthermore, in the largest scenario, ($|\mathcal{S}| = 200$ and $|\mathcal{F}| = 20$), the VNS can significantly reduce the number of required fog nodes without increasing the response time.

5 Concluding Remarks

The facility location problem related to the management of a fog infrastructure is investigated in this work. Specifically, we propose a VNS to solve the optimization problem of mapping sensors data flows to the fog nodes to reduce costs and response time. A qualifying point of our proposal is that starting from a list of *potential* fog nodes, it selects a minimal subset of them to guarantee the satisfaction of a service level agreement.

We test the proposed heuristic against alternative models from the literature. The VNS is evaluated in a realistic scenario of a smart city application over a wide range of scenarios characterized by different load intensities and varied nature of the application (network-bound vs. CPU-bound). The experiments show that the VNS outperforms the best alternative model in 13 out of 20 instances, finding equivalent or very close solutions in the other seven. Moreover, we perform two sensitivity analyses concerning infrastructure size.

From the sensitivity analysis, we conclude that the VNS has a very stable behavior for varying the size of the considered fog infrastructure and smart city application characteristics. In each scenario, the proposed VNS had equal or better performance concerning a state-of-the-art solver in optimizing the mapping of sensor data flows and fog nodes. It reduced the costs due to the turned-on fog nodes keeping the response time within the SLA limits.

In future works, we plan to extend the VNS to handle the fog nodes' heterogeneity and dynamic scenarios in which the load can change through time.

Acknowledgments

This research was partially funded by the National Counsel of Technological and Scientific Development (CNPq - grant 308312/2016-3) and the State of Goiás Research Foundation (FAPEG).

References

1. Ardagna, D., Ciavotta, M., Lancellotti, R., Guerriero, M.: A hierarchical receding horizon algorithm for QoS-driven control of multi-IaaS applications. IEEE Transactions on Cloud Computing pp. 1–1 (2018)
2. Canali, C., Lancellotti, R.: Scalable and automatic virtual machines placement based on behavioral similarities. Computing **99**(6), 575–595 (2017)
3. Canali, C., Lancellotti, R.: A Fog Computing Service Placement for Smart Cities based on Genetic Algorithms. In: Proc. of International Conference on Cloud Computing and Services Science (CLOSER 2019). Heraklion, Greece (May 2019)
4. Canali, C., Lancellotti, R.: GASP: Genetic Algorithms for Service Placement in fog computing systems. Algorithms **12**(10) (2019)
5. Celik Turkoglu, D., Erol Genevois, M.: A comparative survey of service facility location problems. Annals of Operations Research **292**, 399–468 (2020)
6. Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H.: Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet of Things Journal **3**(6), 1171–1181 (2016)
7. Eiselt, H.A., Laporte, G.: Objectives in location problems. In: Drezner, Z. (ed.) Facility Location: A survey of application and methods, pp. 151–180. Springer (1995)
8. Farahani, R.Z., Fallah, S., Ruiz, R., Hosseini, S., Asgari, N.: OR models in urban service facility location: a critical review of applications and future developments. European Journal of Operational Research **276**(1), 1 – 27 (2019)
9. Hansen, P., Mladenović, N., Todosijević, R., Hanaf, S.: Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization **5**, 423–454 (2017)
10. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighbourhood search: methods and applications. Annals of Operations Research **175**(1), 367–407 (2010)
11. Khorov, E., Lyakhov, A., Krotov, A., Guschin, A.: A survey on IEEE 802.11 ah: An enabling networking technology for smart cities. Computer Communications **58**, 53–69 (2015)
12. Marotta, A., Avallone, S.: A Simulated Annealing Based Approach for Power Efficient Virtual Machines Consolidation. In: Proc. of 8th International Conference on Cloud Computing (CLOUD). IEEE (2015)
13. Mladenović, N., Alkandari, A., Pei, J., Todosijević, R., Pardalos, P.M.: Less is more approach: basic variable neighborhood search for the obnoxious p-median problem. International Transactions in Operational Research **27**(1), 480–493 (2020)
14. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
15. Sifaleras, A., Konstantaras, I.: A survey on variable neighborhood search methods for supply network inventory. In: Bychkov, I., Kalyagin, V.A., Pardalos, P.M., Prokopyev, O. (eds.) Network Algorithms, Data Mining, and Applications. pp. 71–82. Springer International Publishing (2020)
16. Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H., Yang, Q.: A hierarchical distributed fog computing architecture for big data analysis in smart cities. In: Proceedings of the ASE BigData & SocialInformatics 2015. Association for Computing Machinery, New York, NY, USA (2015)

17. Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., Rovatsos, M.: Fog orchestration for internet of things services. IEEE Internet Computing **21**(2), 16–24 (2017)
18. Yi, S., Li, C., Li, Q.: A survey of fog computing: Concepts, applications and issues. In: Proc. of 2015 Workshop on Mobile Big Data. pp. 37–42 (2015)
19. Yousefpour, A., Ishigaki, G., Jue, J.P.: Fog computing: Towards minimizing delay in the internet of things. In: 2017 IEEE International Conference on Edge Computing (EDGE). pp. 17–24 (2017)
20. Yu, V.F., Maghfiroh, M.F.: A variable neighborhood search with path-relinking for the capacitated location routing problem. Journal of Industrial and Production Engineering **31**(3), 163–176 (2014)