A Fog Computing Service Placement for Smart Cities based on Genetic Algorithms

Claudia Canali, Riccardo Lancellotti
Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy
{*claudia.canali, riccardo.lancellotti*}*@unimore.it*

Keywords: Fog computing, Optimization model, Genetic algorithms, Smart Cities

Abstract: The growing popularity of the Fog Computing paradigm is driven by the increasing availability of large amount of sensors and smart devices on a geographically distributed area. The scenario of a smart city is a clear example of this trend. As we face an increasing presence of sensors producing a huge volume of data, the classical cloud paradigm, with few powerful data centers that are far away from the data sources, becomes inadequate. There is the need to deploy a highly distributed layer of data processors that filter, aggregate and pre-process the incoming data according to a fog computing paradigm. However, a fog computing architecture must distribute the incoming workload over the fog nodes to minimize communication latency while avoiding overload. In the present paper we tackle this problem in a twofold way. First, we propose a formal model for the problem of mapping the data sources over the fog nodes. The proposed optimization problem considers both the communication latency and the processing time on the fog nodes (that depends on the node load). Furthermore, we propose a heuristic, based on genetic algorithms to solve the problem in a scalable way. We evaluate our proposal on a geographic testbed that represents a smart-city scenario. Our experiments demonstrate that the proposed heuristic can be used for the optimization in the considered scenario. Furthermore, we perform a sensitivity analysis on the main heuristic parameters.

1 Introduction

The explosive growth in data generation in the context of cyber-physical environments is driven by sensors geographically distributed that produce an ever increasing amount of information that require to be filtered and processed. this evolution is leading to the need of new solutions with respect to the classical cloud paradigm. As data increase in size, pushing them towards the Internet core could cause stress for the network infrastructure and introduce excessive delays for the applications.

A solution to improve scalability and reduce network latency lies in taking advantage of the ever increasing presence of fog computing resources. Fog computing, indeed, is a quite novel paradigm that extends cloud computing by moving some services and tasks to the edge of the network. Basically, an intermediate layer of *fog nodes* is placed at the access network, between the data sources and the cloud data center, to host data filtering, aggregation and preprocessing tasks. The fog paradigm was conceived to address applications and services that do not fit well the cloud paradigm, including [10, 13]:

- Applications that require very low and predictable very latency (gaming, videoconferencing)
- Geo-distributed applications (pipeline monitoring, sensor networks to monitor the environment)
- Fast mobile applications (smart connected vehicle, connected rail)
- Large-scale distributed control systems (smart grid, smart traffic monitoring, support for autonomous driving)

For example, in this paper, we focus on a scenario where the Fog infrastructure is applied to reduce latency and delays experienced by a traffic/air pollution monitoring application in a smart city scenario.

Nevertheless, the use of a distributed and complex infrastructure poses several new challenges [18]. Several studies in literature focus on the issues concerning the level of the infrastructure included between the Fog layer and the cloud data centers, not taking into account the previous level connecting the sensors as data sources and the fog nodes. For example, the studies in [7, 19] address the issue of optimizing the allocation of the processing tasks coming from the fog nodes over the cloud infrastructure, proposing differ-

ent solutions also exploiting fog-to-fog nodes communication to reduce the service delay by sharing the incoming load.

A relevant problem that received less attention in literature regards the issue of determining, within a set of available fog nodes, which nodes should receive and elaborate the workload originated from the data sources. In the state-of-the-art proposed solutions, indeed, a common assumption is that the fog nodes directly communicate with the sensors or mobile users through single-hop wireless connections [7] or that a domain of sensor nodes communicate with a domain of fog nodes associated with the specific domain application(s) [19]. However, to guarantee a high QoS in terms of response time through the reduction of the total latency and processing time, the problem of mapping the data flows coming from sensors over the fog nodes that perform operation on them becomes a critical task.

The main contribution of this paper is twofold. First, we propose a formal model for the optimization problem of mapping the incoming workload (data sources) over the fog nodes: our solution takes into account both the latency due to the communication delay of the geographically distributed infrastructure and the processing time on the fog nodes due to the local load. Second, we propose a heuristic to solve the optimization problem in a scalable way; to this aim, we rely on Genetic Algorithms (GAs), that have been previously and successfully exploited in the context of cloud computing and Software-as-a-Service placement [20]. In this paper we focus on a smart city scenario, which is a typical example of environment where data produced by geographically distributed sensors may require efficient processing for a wide range of possible applications, such as traffic monitoring and control, support for autonomous driving and environmental sensing. We evaluate our proposal on a geographic testbed representing the realistic scenario of a Fog architecture located in a small-sized city in Emilia Romagna (Italy) with roughly 180.000 inhabitants. Our experiments demonstrate that the proposed genetic algorithm represent a viable heuristic to solve the mapping problem in the considered scenario. Moreover, we carry out a sensitivity analysis on the main heuristic parameters.

The remainder of this paper is organized as follows. Section 2 describes the problem formally defines the considered optimization model, while Section 3 presents the heuristic algorithms proposed for solving the problem. Section 4 describes the experimental testbed and results used to prove the viability of our approach. Finally, Section 5 discusses the related work and Section 6 concludes the paper with some final remarks and outlines open research problems.

2 Problem definition

2.1 Problem overview

Our problem concerns the management of data flows in a fog infrastructure such as the one shown in Figure 1. The infrastructure, that we assume to be deployed in a smart-city scenario, is composed of three layers: a *sensor layer* that produces data (represented as a set of wireless sensors at the bottom of the figure), a *fog layer* that is responsible for a preliminary processing of data from the sensors (second layer in the figure), while a *cloud layer* that is the final destination of the data (at the top of the figure). The underlying application logic involves the typical services of a smart city scenario. Sensors collect information about the city status, such as traffic intensity or air quality [12]. Such data should be collected at the level of a Cloud infrastructure to provide value-added services such as traffic or pollution forecast. The proposed fog layer intermediates the communication between the sensors and the cloud to provide scalability and reliability in the smart city services.

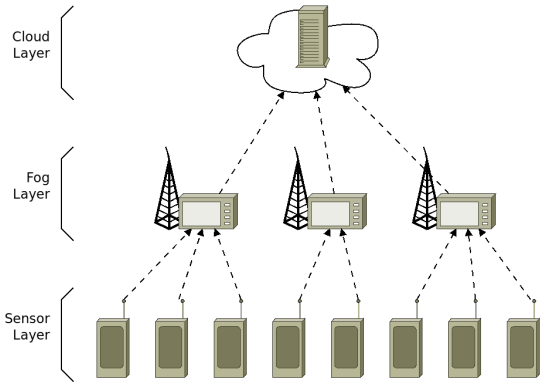

Figure 1: Fog infrastructure

In our model, we assume a stationary scenario where a set of similar sensors $\mathcal{S}$ are distributed over an area (we consider the sensors to be not moving, although a different scenario, where mobility is taken into account can be easily introduced in our model). Furthermore, we assume that sensors are producing data at a steady rate, with a frequency that we denote as λ_i for the generic sensor i (for a summary of the symbols used in the model, the reader may refer to Table 1). The fog layer consists of a set of nodes $\mathcal{F}$ that receive the data from the sensors and performs opera-

tions on them. These operations typically include pre-processing of the data, such as filtering and/or aggregation, or may include some form of analysis to identify anomalies or problems as fast as possible. The refined data samples from the fog nodes are then sent to a cloud platform where additional analysis is carried out and where all the information is stored. These additional analysis tasks are typically highly expensive from a computational point of view. Again, our model presents only one cloud data center, but it is easy to extend it to a multi-cloud scenario.

As the problem concerning the management of large cloud data centers has been widely addressed in literature [14], we do not consider the inner details of the cloud layer in our problem modeling, such as the computation time at the level of the cloud data center. Instead, we focus our attention to the problem of coordinating the communication of the elements in the sensor layer with the nodes in the fog layer. Specifically, we want to guarantee an high QoS, in terms of fast response. To this aim, we must consider that the response time has the following major contributions that should be taken into account:

- Network-based latency due to the communication between the sensor and the fog nodes. We denote this value as $\delta_{i,j}$ where i is a sensor and j is a fog node.
- Network-based latency due to the communication between the fog node and the cloud data center. As this depends just on the fog node due to the single cloud considered in our model, we simply denote this measure as δ_j, where j is the fog node.
- Computation time on the fog node. This time depends on the computation cost of the request (we denote as $1/\mu_j$ the time to process a packet of data from a sensor on fog node j) and on the data rate λ_i of all the sensors i that are communicating with the fog node j – we define as λ_j the incoming data rate at fog node j.

For the sake of clarity we summarize the symbols used throughout the paper in Table 1.

2.2 Optimization model

The main problem in the considered fog scenario is how to map on the fog nodes the data flows coming from the sensors. To this aim, we define an optimization problem where we use as the main decision variable a matrix of boolean flags $x_{i,j}$. In our model $x_{i,j} = 1$ if and only if sensor i is sending data to fog node j, otherwise $x_{i,j} = 0$. As the function of fog nodes is to pre-process the incoming data performing filtering and aggregation, we consider that all the data of a sensor must be sent to the same fog node and cannot be distributed across the fog layer.

Again, the reader may refer to Table 1 for a summary of the parameters used in our model.

Table 1: Notation.

Symbol	Meaning/Role
	Decision variables
$x_{i,j}$	Sending data flow from sensor i to Fog node j
	Model parameters
$\mathcal{S}$	Set of sensors
$\mathcal{F}$	Set of Fog nodes
λ_i	Outcoming data rate from sensor i
λ_j	Incoming data rate at fog node j
$1/\mu_j$	Processing time at fog node j
$\delta_{i,j}$	Communication latency between sensor i to Fog node j
δ_j	Communication latency Fog node j and Cloud data center
	Model variables
i	Index of a sensor
j	Index of a Fog node

The optimization model to address the previously-described problem can be formalized as follows, with an approach similar to the problem of allocating requests over a distributed infrastructure, such as VMs on a Cloud [14, 11, 8]. In particular, we introduce a matrix of boolean decision variables $X = x_{i,j}$ that is used to define the objective function and the constraints as follows:

$$\min obj(X) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} x_{i,j} \cdot \left(\frac{1}{\mu_j - \lambda_j} + \delta_{i,j} + \delta_j \right) \tag{1.1}$$

subject to:

$$\lambda_j = \sum_{i \in \mathcal{S}} x_{i,j} \cdot \lambda_i \quad \forall j \in \mathcal{F}, \tag{1.2}$$

$$\sum_{j \in \mathcal{F}} x_{i,j} = 1 \quad \forall i \in \mathcal{S}, \tag{1.3}$$

$$\lambda_j < \mu_j \quad \forall j \in \mathcal{F}, \tag{1.4}$$

$$x_{i,j} = \{0, 1\}, \quad \forall i \in \mathcal{S}, j \in \mathcal{F}, \tag{1.5}$$

In the problem formalization, the objective function 1.1 aims at reducing the total (and hence the average) latency and processing time from every sensor to the cloud, including the operation carried out at the level of fog computing nodes. The expression of response time used for our objective function is consistent with other studies in literature focusing on distributed cloud infrastructures [2]. Specifically, the average processing time is derived from Little's result applied to a M/G/1 model and considers just the average arrival frequency λ_j and the processing rate μ_j of each fog node j. This definition of the response time has been widely adopted in literature, for ex-

ample in [2]. The second part of the objective function, that is the latency contribution, captures effectively the communication delay of a geographically distributed infrastructure using the latencies $\delta_{i,j}$ and δ_j.

Together with the objective function, we have a set of constraints. Equation 1.2 defines the incoming load λ_j on each fog node j. Constraint 1.3 means that for every sensor i, we direct its output to one and only one fog node. Constraint 1.4 guarantees that, for every fog node j, we avoid a congestion situation, where the incoming load λ_j exceeds the processing capability μ_j of that node. Finally, constraint 1.5 defines the boolean nature of the decision variables $x_{i,j}$.

3 Heuristic algorithm

The previously defined optimization problem is a general definition for mapping sensors over fog nodes. The actual solution of this problem can be carried out using commercial solvers, such as CPLEX or K-NITRO, already applied in similar problems [4], or can be addressed using a specific heuristic.

We consider interesting in this scenario to introduce a solution method based on the Genetic Algorithms (GAs) heuristic and to compare this approach with commercial solvers to validate its viability. In GAs we operate on a *population* of *individuals*, where each individual represents a possible solution of the problem. The solution is encoded in a *chromosome* that defines the individual and the chromosome is composed by a fixed number *genes* that represent the single parameters characterizing a solution of the problem.

A population of individuals is typically initialized randomly. A *fitness function*, that describes the objective function of the optimization problem is applied to each individual. The evolution of population through a set of *generations* aims at improving the fitness of the population using the following main operators:

Mutation is a modification of a single or a group of genes in a chromosome describing the individual of the population. Figure 2 presents an example of such operator where the i^{th} gene of the rightmost individual in the K^{th} generation undergoes a mutation. The main parameter of this operator is the probability of selecting an individual to perform a mutation on one of its genes. In the sensitivity analysis in Section 4.3, we will refer to this probability as P_{mut}.

Crossover is a merge of two individuals by exchanging part of their chromosomes. Figure 2, again, provides an example of this operator applied to the two individuals composing the population at the K^{th} generation. In particular, in Figure 2 the child individual is characterized by a chromosome containing the genes from c_0 to c_{i-1} from the rightmost parent and the genes from c_i to c_S from the leftmost parent. The main parameter of this operator concerns the selection of the parents. In the sensitivity analysis in Section 4.3, we will refer to the probability of selecting an individual for a crossover operation as P_{cross}.

Selection concerns the criteria used to decide if an individual is passed from the K^{th} generation to the next. The typical approach in this case is to apply the fitness function to every individual (including new individuals generated through mutation and crossover) and to consider a probability of being selected for the next generation that is proportional to the fitness value. The selection mechanism ensures that the population size remains stable over the generations.

When applying a GAs approach to the problem of mapping sensors over the fog nodes of a distributed architecture, we must encode a solution as a gene. In particular, we aim to formalize the relationship between the model in Section 2.2 and the GA chromosome encoding. Hence, we define a chromosome as a set of S genes, where $S = |\mathcal{S}|$ is the number of sensors. Each gene is an integer number from 1 to F, where $F = |\mathcal{F}|$ is the number of fog nodes in our infrastructure. The generic i^{th} gene in a chromosome c_i can be defined as: $c_i = \{j : x_{i,j} = 1\}$. Due to constraint 1.3 in the optimization model, we know that only one fog node will receive data from sensor i, so we have a unique mapping between a solution of the problem expressed using the decision variable $x_{i,j}$ and the GA-based representation of a solution. As we can map each chromosome into a solution of the original optimization problem, we can use the objective function 1.1 as the basis for fitness function of our problem. Constraints 1.3 and 1.5 are automatically satisfied by our encoding of the chromosomes. The only constraint we have to explicitly take into account is constraint 1.4 about the fog node overload. As embedding the notion of unacceptable solution in a genetic algorithm may hinder the ability of the heuristic to converge towards a solution, we prefer to insert this information into the fitness function, in such a way that the individual providing a solution where one or more fog nodes are overloaded is characterized by a high penalty and is unlikely to enter in the subsequent generation.

Multiple optimization algorithms have been considered before adopting the choice of a genetic algo-

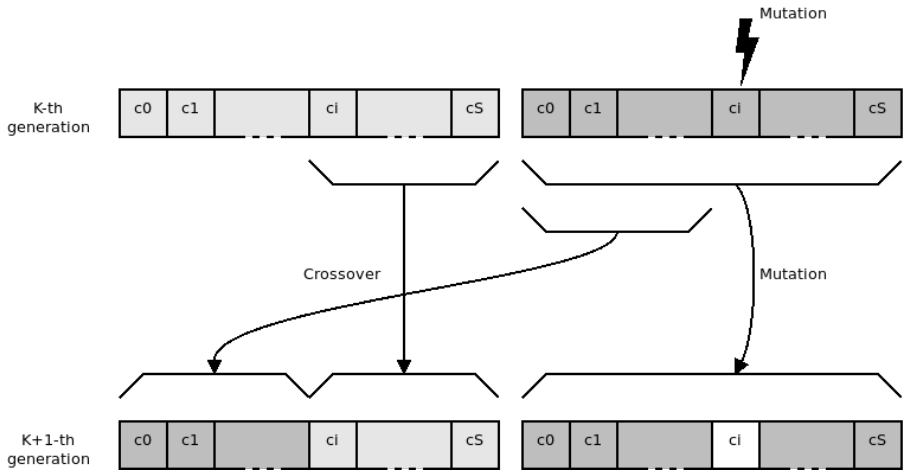

Figure 2: Examples of genetic algorithms operators

rithm. On one hand, greedy heuristics tend to provide performance that heavily depends on the inherent nature of the problem. For example, the non-linear objective function may hinder the application of some greedy approaches, while the number of sensors that may be supported by each fog node may have significant impact on the performance of branch and bound heuristics. As we aim at providing a general and flexible approach to tackle this problem, we prefer to focus on meta-heuristics that are supposed to be better adaptable to a wider set of problem instances [3]. Among these solution, we focus on evolutionary programming in general and on genetic algorithms in particular as this glass of heuristics has been proven a viable option in similar problems such as the problem of allocating VMs on a cloud infrastructure [20].

4 Experimental results

4.1 Experimental testbed

We tested the viability of our approach focusing on a typical Fog scenario consisting of three components that are: (1) a large number of sensors; (2) a set of fog nodes, that we assume to be processing nodes with limited computational power, responsible for data filtering and aggregation; (3) a cloud data center that is the final destination of the pre-processed sensor data. Our testbed is based on a smart city scenario. To guarantee a realistic experimental testbed, we modeled the scenario based on the small city of Modena in Italy, which has roughly 180.000 inhabitants.

Figure 3 provides a map of the sensors, fog nodes and cloud data center considered for the smart city scenario. We assume that our system supports an application for traffic monitoring, with wireless sensors placed on the main streets of the city and collecting data about: the number of cars passing on the street, their speed and other traffic related measures (an example of this application can be found in the Trafair Project [12]). To build the map of sensors, we collected a list of the main streets in the city and we georeferenced them. We assume that in each main street we have at least a sensor producing data. We selected a group of 5 buildings hosting the offices of the municipality and we use them as the location of the fog nodes – this assumption is consistent with the current trend of interconnecting the main public building of each city with high bandwidth links. Our final scenario is composed of 90 sensors and 5 nodes. The euclidean distance between the nodes is used to model the communication latency and the delay is in the order of tens of milliseconds (that is a common value for geographic networks). Finally, there is only one cloud data center placed in the actual location of the municipality data center.

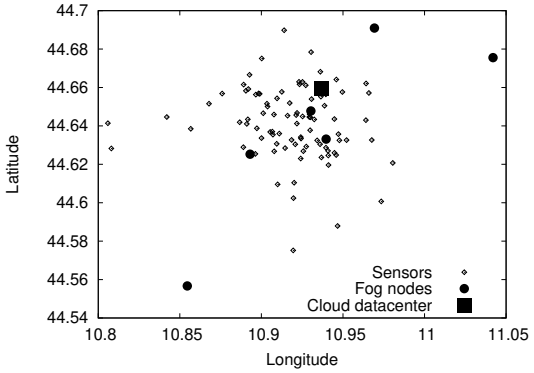

Figure 3: smart city scenario

For the data processing model, we consider a pre-

liminary smart city setup, where we have sensors collecting a large set of samples concerning vehicular traffic and environmental quality indicators. As the data should support a real-time monitoring of the city, we assume to have at least a set of samples available every second. Hence, in our simulation, we consider that $\lambda_i = 1, \forall i \in \mathcal{S}$. For the fog nodes processing capability, we assume that the node have a computational power orders of magnitude higher than the sensors; hence, considering the limited complexity of most filtering and aggregating tasks, we assume that each node can process up to 100 sensor feeds concurrently without risking overload ($\mu_j = 100, \forall j \in \mathcal{F}$).

For the solution of the optimization problem, we first implemented the model using the AMPL language [1] and we use the commercial solver CPLEX. The obtained solution is then used as a comparison for our heuristic implementation. Specifically, the AMPL definition is directly based on the optimization problem discussed in Section 2. The genetic algorithm is implemented using the Distributed Evolutionary Algorithms in Python (DEAP) framework [6] based on the details provided in Section 3.

In the evaluation of the genetic algorithm approach, we run the experiments 10 times and we average the main metrics. In particular, for each run of the genetic algorithm, we consider the best achieved solution at each generation. The algorithm maintains a population of 100 individuals and we force a stop of the algorithm after 300 generations. When evaluating the convergence speed of the algorithm, we define as the optimality-reached criteria the fact that the best individual in the population has a fitness value within 1% of the optimum value obtained using the AMPL solver.

4.2 Genetic algorithm performance

The first analysis in our experiment aims at demonstrating that the genetic algorithms can reach an optimal solution even in presence of a complex problem with integer programming and a non-linear objective function.

Figure 4 shows, for a run of the genetic algorithm, the fitness value (corresponding to the objective function) of the best individual within the population as a function of the generation number. The optimal value obtained using the AMPL-based problem definition and the CPLEX solver is shown as the horizontal thick dashed line in the lower part of the graph. We observe that, for the genetic algorithm, convergence is very fast, with the objective function almost reaching the optimal value in little more than 50 generations. This result is quite interesting because it means that the genetic algorithm is able to explore the solution space in a small amount of time, reaching the proximity of the optimum (even if the actual optimum value may require more generation to be found). Comparing the execution time, the time for the genetic algorithm to reach a value within 1% of the optimum is roughly one order of magnitude lower compared to the complete run of the AMPL-based solution.

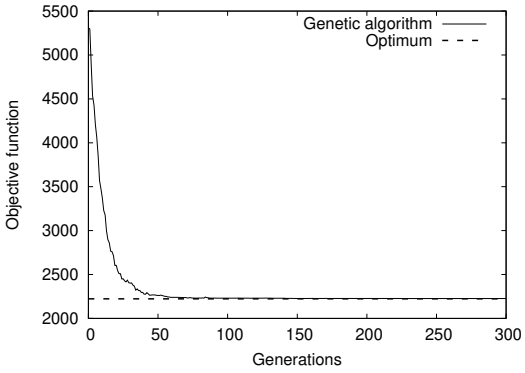

Figure 4: Genetic algorithm performance

4.3 Sensitivity analysis

Our first experiment showed that the genetic algorithm is able to reach an optimal solution rapidly. However, we also consider important to evaluate if this behavior occurs just for a properly tuned algorithm or if the property of fast convergence is maintained. To this aim, we carry out a sensitivity analysis with respect to several parameters of the algorithm and we present the most interesting findings.

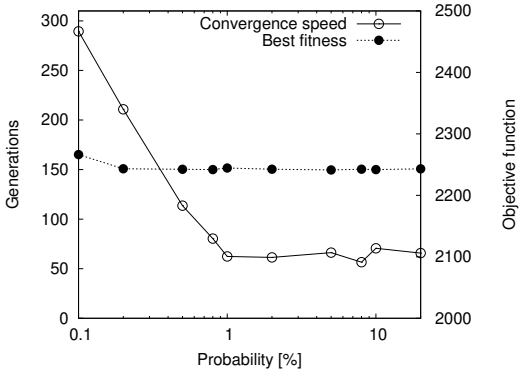

Figure 5: Sensitivity to crossover probability

The first analysis concerns the probability of selecting an individual for a crossover operation P_{cross}. Figure 5 shows the number of generations required to converge (that is obtaining a value within 1% of the optimum) and best value of the objective function

(that is used as the fitness score in our analysis) as a function of P_{cross} that ranges from 0.1% to 20%.

We observe that the time to converge shows a non-negligible dependence from this value: the generations needed to converge start close to the threshold value of 300 and gradually descend as the we approach a value of 1%. After this point, the convergence speed remains stable. If we analyze this behavior, we find that the low crossover probability has a detrimental effect on the time to explore the space of solutions because a low value in this probability hinders the possibility of a good solution to replicate its genes in the population. For very high crossover probabilities, the effect is not so interesting, because the good performing genes becomes rapidly widespread and crossing similar solutions provides a limited performance gain. On the other hand, the best value of the objective function remains quite stable with respect of this parameter (as expected) because in every case we reach convergence, hence, for every probability value we are still within 1% of the optimum.

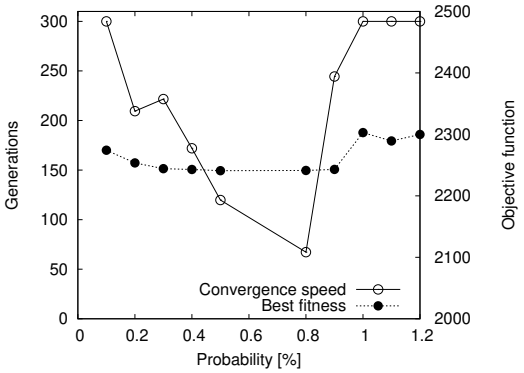

Figure 6: Sensitivity to mutation probability

The second significant analysis carried out concerns the impact of the mutation probability P_{mut}. Figure 6 shows the convergence speed and the objective function corresponding to the best fitness as a function of the mutation probability. Once again the most significant metric to observe is the number of generation needed to reach a value within 1% of the optimum (that measures the convergence speed). In this case we observe that both very low values ($P_{mut} \leq 0.1\%$) and high values ($P_{mut} \geq 1\%$) result in the algorithm failing to converge within the threshold of 300 generations. In general, we observe a clear V-shaped curve with a point of fast convergence for a probability close to 0.8%. To understand this behavior, we must consider the two-fold effect of mutations. On one hand, a low mutation probability hinders the ability to explore the solutions pace, simply because solution not present in the initial randomly-generated population may be reached only through mutation. On the other hand, a mutation in an already good solution may simply reduce the ability of the algorithm to converge, because the population keeps changing too rapidly. If we observe the objective function values as a function of the mutation probability, we observe that, when convergence is reached, the achieved fitness is quite stable; on the other hand, when no convergence occurs, the output of the genetic algorithm may provide a solution significantly worse compared to the potential optimal solution.

5 Related work

The explosive growth in the generation of data and the need for their processing to provide innovative services and applications has recently led researchers to focus on fog computing solutions to complement the cloud systems capabilities. To always exchange localized data from and to the remote cloud, indeed, tends to be inefficient under different points of view, thus motivating fog computing to partially process workload and data locally on fog nodes [7, 16, 18, 17].

A survey discussing representative application scenarios and identifying various issues related to design and implementation of fog computing systems can be found in [18], while the study in [17] provides an overview of the core issues, challenges, and future research directions in fog-enabled orchestration for IoT services, focusing on smart cities as main motivating example of the research. Also our study considers the smart cities as a meaningful scenario where large amount of sensors and smart devices produce a huge volume of data on a geographically distributed area. Specifically, we focus on the specific issue of distributing the incoming workload over the fog nodes to minimize communication latency while avoiding overload.

Some existing studies focus on the issue of allocating the processing tasks coming from the fog nodes to the cloud nodes to optimize performance and reduce latency. Among these studies, Deng et al. [7] explore the tradeoff between power consumption and transmission delay in the fog-cloud computing system, formulating an optimization of the allocation problem among fog and cloud nodes. The study in [19] explicitly focuses on the issue of minimizing the service delay in IoT-fog-cloud application scenarios, proposing a delay-minimizing policy for fog nodes: in contrast to other proposals in literature, the proposed policy employs fog-to-fog communication to reduce the service delay by sharing load. It is worth to note that in both these studies the issue

of mapping data sources on the fog nodes is not taken into account. In [7], indeed, the fog nodes directly communicate with the mobile users through single-hop wireless connections using the off-the-shelf wireless interfaces, such as WiFi, Bluetooth, etc., while in [19], the communication among the IoT nodes and fog nodes works as follows: a domain of IoT nodes (in a factory, for instance) communicate with a domain of fog nodes associated with the specific domain application(s). On the other hand, our study focuses on the issue of optimizing the mapping of the workload coming from data sources over the fog nodes.

Among the studies focusing on fog computing applied to the same context of our paper, in [16] a hierarchical 4-layer Fog Computing architecture is proposed for big data analysis in smart cities. The layered Fog computing network exploits the natural characteristic of geo-distribution in big data generated by massive sensors, performing latency-sensitive applications and providing quick control loop to ensure the safety of critical infrastructure components. In this paper, the mapping between fog nodes and sensors is fixed: each fog node is connected to and responsible for a local group of sensors that cover a neighborhood or a small community.

The study in [5] considers Data Stream Processing (DSP) applications and, specifically, the so called operator placement problem, that is the allocation of DSP operator on fog nodes with the goal of optimizing the applications Quality of Service (QoS). The optimal DSP placement is modeled as an Integer Linear Programming (ILP) problem. In this case the authors made the assumption that it is possible to split the incoming data flow for parallel processing, while we consider generic applications where this assumption may not be true.

Finally, genetic algorithms (GAs) have been successfully applied to the context of cloud computing in recent literature. The study in [20] exploits GAs to produce a suitable and scalable solution for the Software as a Service (SaaS) Placement Problem [20], while Karimi et al. [9] proposes a QoS-aware service composition for cloud computing systems based on GAs.

6 Conclusions and future work

Throughout the present paper, we faced a typical scenario that motivates a fog computing infrastructure: a smart city where sensors or smart devices disseminated over a geographic area produce a large amount of data. We pointed out that a classical cloud scenario, where all the communications converge on a single cloud data center (or, at most, on few data centers) becomes unmanageable due to the risk of network congestion. As some applications in a smart city scenario are clearly latency-sensitive (e.g. applications related to automated traffic management) or produce a bulk of data that could create congestion at the network level (e.g., widespread sensors for environmental analysis) the most suitable approach is to push a level a pre-processing as close as possible to the sensors to filter and aggregate the data or to perform latency-critical tasks.

The presence of fog computing layer, opens the problem we discussed in our research, that is how to map the data streams from the sensors over the fog nodes. We provided a formal model for the problem of minimizing the overall latency experienced in the system, considering both data transfer and processing times. Furthermore, we proposed an heuristic algorithm, based on genetic programming to solve the problem without the need to rely on an external solver.

Our proposed solution is validated using a smart-city scenario based on a realistic testbed. The experiments demonstrate the viability of the proposed genetic algorithm to solve the problem and provides a sensitivity analysis with respect to the main parameters of the proposed heuristic.

This paper is just a first step in a research line on the application of fog computing to smart cities. We plan to extend the current research taking into account more complex scenarios that involve dynamic changes in the workload (for example to capture sensor mobility or to consider adaptive sampling techniques at the sensor level) providing contributions both at the level of scenario definition and at the level of algorithm and architecture proposals.

REFERENCES

[1] AMPL: Streamlined modeling for real optimization, 2018. – https://ampl.com/.

[2] D. Ardagna, M. Ciavotta, R. Lancellotti, and M. Guerriero. A hierarchical receding horizon algorithm for qos-driven control of multi-iaas applications. *IEEE Transactions on Cloud Computing*, pages 1–1, 2018.

[3] S. Binitha, S. S. Sathya, et al. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.

[4] C. Canali and R. Lancellotti. Scalable and automatic virtual machines placement based on behavioral similarities. *Computing*, 99(6):575–595, June 2017.

[5] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli. Optimal operator placement for distributed stream processing applications. In *Proceedings of the 10th ACM International Conference on Distributed and*

Event-based Systems, DEBS '16, pages 69–80, New York, NY, USA, 2016. ACM.

[6] DEAP: Distributed Evolutionary Algorithms in Pyton, 2018. – https://deap.readthedocs.io.

[7] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181, Dec 2016.

[8] H. Duan, C. Chen, G. Min, and Y. Wu. Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Future Generation Computer Systems*, 74:142 – 150, 2017.

[9] M. B. Karimi, A. Isazadeh, and A. M. Rahmani. Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm. *J. Supercomput.*, 73(4):1387–1415, Apr. 2017.

[10] J. Liu, J. Li, L. Zhang, F. Dai, Y. Zhang, X. Meng, and J. Shen. Secure intelligent traffic light control using fog computing. *Future Generation Computer Systems*, 78:817 – 824, 2018.

[11] M. Noshy, A. Ibrahim, and H. Ali. Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications*, 110:1–10, 2018. cited By 1.

[12] T. project staff. Forecast of the impact by local emissions at an urban micro scale by the combination of lagrangian modelling and low cost sensing technology: the trafair project. In *Proc. of 19th International conference on Harmionisation within Atmospheric Dispersion Modelling for Regulatory Purposes*, Bruges, Belgium, June 2019.

[13] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao. Vehicle control system coordinated between cloud and mobile edge computing. In *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 1122–1127, Sept 2016.

[14] M. Shojafar, C. Canali, and R. Lancellotti. A Computation- and Network-Aware Energy Optimization Model for Virtual Machines Allocation. In *Proc. of International Conference on Cloud Computing and Services Science (CLOSER 2017)*, Porto, Portugal, Apr. 2017.

[15] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar. Towards qos-aware fog service placement. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 89–96, May 2017.

[16] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData & SocialInformatics 2015*, ASE BD&SI '15, pages 28:1–28:6, New York, NY, USA, 2015. ACM.

[17] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos. Fog orchestration for internet of things services. *IEEE Internet Computing*, 21(2):16–24, Mar 2017.

[18] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42, New York, NY, USA, 2015. ACM.

[19] A. Yousefpour, G. Ishigaki, and J. P. Jue. Fog computing: Towards minimizing delay in the internet of things. In *2017 IEEE International Conference on Edge Computing (EDGE)*, pages 17–24, June 2017.

[20] Z. I. M. Yusoh and M. Tang. A penalty-based genetic algorithm for the composite saas placement problem in the cloud. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.