An Evolutionary-based Algorithm for Smart-living Applications Placement in Fog Networks

Raheleh Moallemi, Arash Bozorgchenani, Daniele Tarchi
Department of Electrical, Electronic and Information Engineering
University of Bologna, Italy
Email: raheleh.moallemi@studio.unibo.it, {arash.bozorgchenani2,daniele.tarchi}@unibo.it

***Abstract*—Fog computing is an emerging model, complementing the cloud computing platform, introduced to support the Internet of Things (IoT) processing requests at the edge of the network. Smart-living IoT scenarios require the execution of multiple processing tasks at the edge of the network and leveraging on the Fog Computing approach results to be a worthwhile solution. Genetic Algorithms (GA) are a heuristic search and optimization class of techniques inspired by natural evolution. We propose two GA-based approaches for optimizing the processing task placement in a fog computing edge infrastructure aiming to support the Smart-living IoT nodes requests. The numerical results obtained in Matlab show that both GA-based approaches allow to maximize the covered areas while minimizing the resource wastage through the minimization of the overlapping areas.**

I. Introduction

Currently, smart-living scenarios are characterized by a massive amount of Internet of Things (IoT) data emitted by the distributed devices and sent to the cloud for centralized processing and then are sent back from the cloud to data consumers who are often located very close to the generating data sources. This leads to high delays and considerable costs for the usage of cloud-based computational resources. In contrast, the decentralized processing of IoT data has been identified as a suitable approach [1].

Edge IoT devices (e.g., gateways, access points) offer computational, storage, and networking resources, even if in a limited way. Therefore, the presence of these resources allows moving the execution of Smart-living IoT applications to the edge of the network [2]. The paradigm is known as Fog Computing (FC) [3] aiming at extending services provided by the cloud down to the network edge. It also provides a hierarchical, dense and geographically distributed architecture to store and process data in the network devices located between end-users smart objects and Cloud data-centers. Unlike the Cloud, the Fog can support latency-critical IoT applications requiring short response times. Fog computing permits a severe reduction in the overall network latency [4].

Fog computing, firstly introduced by Cisco [5], emerges as a new approach aiming at solving the latency sensitive computing problems. FC utilizes the local computing resources instead of a remote cloud for data processing so that the geographic vicinity between the data source and processors allows decreasing the transmission latency [6].

A huge number of geo-distributed devices (including user devices, routers, switches, and access points) create *small-sized clouds* at the edge of the network, managed in a distributed way by the devices themselves. The basic idea in FC is that instead of always upload/download data to/from core network as in traditional cloud computing, the edge devices which are in proximity, can obtain data from other users through the direct link such as Device-to-Device (D2D) communication and adjacent Small Cell (SC) networks. Moreover, the edge devices in fog network (FogNet) release some of their resources like computing and storage capacity to support the demands of their neighbors. Only those tasks not well handled by the edge devices are sent to the core cloud part for further processing. As a result, fog computing significantly reduces the computing and routing burden of the cloud [7].

On one hand, a FogNet should be able to serve as many users as possible, but on the other hand due to the reduced resources of each node, it is not possible to deploy a network composed of fully-functional nodes. Thus, it requires optimizing the placement of the applications in each node depending on their usage, the nodes requests, as well as the nodes coverage, storage and processing capabilities [4]. The goal of this paper is an application placement technique having the goal of minimizing the outage probability of the requests of the different nodes constrained to the resource limited characteristics of the Fog Nodes (FNs). In particular a Genetic Algorithm (GA)-based approach is foreseen to be used.

While conventional gradient-based optimization algorithms require a smooth and uni-modal space, GAs do not have these limitations and are useful for solving combinatorial problems; therefore, GA has the massive potential use for computer network and telecommunication applications [8]. GAs are a class of optimization algorithms used to determine the solution(s) to an assigned computational problem that maximizes or minimizes a special function [9]. GAs are one of the most efficient meta-heuristic approach used to solve different problems. A GA starts its search with a random set of solutions where every solution has assigned a fitness that is directly related to the objective function of the search and

This work has been partially supported by the project "GAUChO - A Green Adaptive Fog Computing and Networking Architecture" funded by the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015 - grant 2015YPXH4W_004.

problem. After that, the population of solutions is modified to a new population by applying three operators like natural genetic operators as selection, crossover, and mutation which works iteratively by successively applying these three operators in each generation until a terminations criterion is satisfied [10].

This paper introduces a GA-based approach appropriate for problems which are large, non-deterministic, non-linear and discrete in nature similar to our work; in particular, we resort to the meta-heuristic based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [11] to discuss the optimal application placement in fog networks. In this paper, we model the placement problem as a coverage overlapping problem and, hence, we apply the GA for maximizing the covered area by each application and jointly minimizing the overlapping areas of applications so as to maximize the served IoT devices in the system area. We formulated the application placement problem for achieving an optimal placement of the applications on the FNs. Through MATLAB-based simulations, empirical results illustrate that the proposed approach significantly increases the number of served IoT devices while minimizing the resource wastage.

II. System Model

The envisaged environment is composed by a set of sensors, a set of FNs, and a set of IoT services (i.e., application instances) that can be executed in FNs in a large service area.

Sensors collect data from the humans, in-house and medical appliances, as well as from the outer environment, to be sent to the associated application instances for analysis and/or processing sake. Application instances consume processing, storage, energy and communication resources of the hosting node. In general, IoT data can be processed and stored in different locations in the considered Fog architecture [4], allowing a flexible deployment of the applications. In our proposal, we expect that the system has knowledge about: (i) the IoT sensors position and communication range, (ii) FNs position and communication range, (iii) deployable applications on the FNs, (iv) the FNs capacity expressed in terms of maximum number of applications that can be stored.

In such a scheme, the goal is to maximize the number of IoT sensors that can be served by a given application that can be deployed on a certain FN, stated that the IoT sensors and the selected FN are within the communication range.

We can model the scenario as a two-layer architecture where the first layer is composed by a set of fixed sensors denoted as $\mathcal{U} = \{U_1, \ldots, U_n, \ldots, U_N\}$, and the second layer is composed by a set of FNs denoted as $\mathcal{F} = \{F_1, \ldots, F_m, \ldots, F_M\}$. The set of applications to be deployed on the FNs is denoted as $\mathcal{K} = \{K_1, \ldots, K_p, \ldots, K_P\}$. We define the placement binary variable:

$$c_{m,p} = \begin{cases} 1 & \text{if } K_p \text{ is placed in } F_m \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

defining the placement of the generic K_p on the FN F_m. Hence, the overall placement of the P applications on the M FNs can be written as:

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,p} & \cdots & c_{1,P} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,p} & \cdots & c_{m,P} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{M,1} & \cdots & c_{M,p} & \cdots & c_{M,P} \end{pmatrix}, c_{m,p} \in \{0,1\} \tag{2}$$

Each FN has a certain capacity, C_m, corresponding to the maximum number of applications that can be deployed on the generic mth FN. Therefore, the application placement of the mth FN is constrained to:

$$\sum\nolimits_p c_{m,p} \leq C_m \leq P, \quad \forall F_m \in \mathcal{F} \tag{3}$$

Now, we define the set of available FNs that are able to serve the nth IoT sensor as:

$$\mathcal{F}_{FN}^n = \{F_m | d_{nm} \leq R_m\}, \quad \text{for } m = 1, \ldots, M \tag{4}$$

where R_m is the coverage range of the mth FN, and d_{nm} is the Euclidean distance between the nth IoT sensor and the mth FN.

In our problem, each IoT sensor can request multiple services to the FNs. If the application implementing the requested service is not stored in the cache of the FN, the sensor cannot use that service. Hence, it is of great importance to place the applications in the FNs such that the sensors' requests can be responded by the FNs in their coverage area. Each time an application K_p is requested, the nth sensor looks up for the applications in the cache of the FNs listed in $\mathcal{F}_{FN}^n$.

The application placement problem in our work is defined as placing the applications in the FNs such that the number of covered sensors is maximized. We define with $K_p^n(\tau)$ a generic application requested by the nth sensor at time instant τ. We also define $\Omega(K_p^n(\tau))$, an operator indicating if the requested application K_p by the nth sensor is deployed on a FN that can be reached, as:

$$\Omega\left(K_p^n(\tau)\right) = \begin{cases} 1, & \text{if } \sum\limits_{F_m \in \mathcal{F}_{FN}^n} c_{m,p} \geq 1 \\ 0, & \text{Otherwise} \end{cases} \tag{5}$$

corresponding to state that the pth application is in the cache of at least one of the FNs in the sensor's coverage area. The goal of the problem is:

$$\mathbf{P1}: \max_n \left\{\Omega\left(K_p^n(\tau)\right)\right\} \ \forall n \tag{6}$$

subject to (3) and (4), corresponding to maximize the number of sensor nodes able to use the available applications, subject to the FNs capacity and coverage constraints.

Maximizing the number of serviced sensors by a certain application correspond to maximize the area covered by that specific application when placed in multiple nodes. On the other side, a deployment where multiple FNs are covering the same area with the same application is a resource wastage.

This leads to the definition of an equivalent problem based on a joint goal corresponding to maximize the coverage area of each application and jointly minimize the overlapping areas of FNs with the same applications in their cache, leading to:

$$\mathbf{P2}: \begin{cases} \max\limits_{p}\{A_p^C\} = \max\limits_{p}\left\{\frac{|\mathcal{U}_p^C|}{|\mathcal{U}|}\right\} \\ \min\limits_{p}\{A_p^O\} = \min\limits_{p}\left\{\frac{|\mathcal{U}_p^O|}{|\mathcal{U}_p^C|}\right\} \end{cases} \tag{7}$$

where A_p^C and A_p^O represent, respectively, the covered area and the overlapping area related to the application K_p, $|\mathcal{U}_p^C|$ is the cardinality of the set of sensors covered by the FNs having the pth application, defined as:

$$|\mathcal{U}_p^C| = |\{U_n | d_{n,m} \leq R_M \wedge c_{m,p} = 1\}|, \forall F_m \in \mathcal{F}_{FN}^n$$

$|\mathcal{U}_p^O|$ is the cardinality of the set of sensors covered by more than one FN having the pth application, defined as:

$$|\mathcal{U}_p^O| = \left|\left\{U_n | d_{n,m} \leq R_m \wedge \sum\nolimits_{F_m \in \mathcal{F}_{FN}^n} c_{m,p} > 1\right\}\right|,$$

and $|\mathcal{U}|$ is the cardinality of the set $\mathcal{U}$, corresponding to all the sensors in the area.

III. NSGA-II BASED APPLICATION PLACEMENT

Due to the conflicting objectives, i.e., maximizing the coverage area and jointly minimizing the overlapping areas, the problem of application placement to the FNs, introduced in previous section, can be characterized by a Pareto front of solutions. In order to find the Pareto front we propose a Multi-objective Evolutionary Algorithm (MOEA). EAs offer an efficient way for finding a high quality approximation of the Pareto fronts.

MOEA exploits the evolutionary principles of crossover, mutation, and selection of Darwinian evolution to find the Pareto front. While by crossover and mutation operations, the solutions are combined probabilistically in order to generate possible better solutions, the selection phase discards the low-quality solutions and high-quality solutions are selected for next generation. Before describing the details of our proposed MOEA, we first describe Pareto-optimality and give some definitions as follows.

Since in our problem there are multiple conflicting objectives, the concept of optimality transforms to Pareto-optimality, as any solution point has to be evaluated along multiple dimensions. Thus, the quality of a solution is determined by its Pareto-dominance with respect to previously computed solutions. In particular, let $\boldsymbol{S} = \{s_1, s_2, ..., s_Z\}$ be a set of solutions generated by an MOEA, where s_z corresponds to a possible application placement solution as represented in (2), and Z is the total number of generated solutions. Considering two solutions, say s_1 and s_2, for a given problem with b conflicting objectives, say ω_b (for all $b \in [1, B]$), we adopt the Pareto-dominance definition proposed in [11], i.e.,

***Definition** 1: Let $\omega_b(s)$ be the value of the objective function for the bth objective evaluated at some solution s. Then s_1 is said to Pareto-dominate s_2 (i.e., $s_1 \succ s_2$) if $\omega_b(s_1) \leq \omega_b(s_2)$ for all $b \in [1, B]$, and there exists some $\nu \in [1, B]$ such that $\omega_\nu(s_1) < \omega_\nu(s_2)$.*

The previous Pareto-dominance definition allows to classify solutions based on their quality. Consequently, we can define the set of non-dominated solutions as:

$$S^P = \{s_a \mid \nexists s_b \succ s_a, \text{ for } 1 \leq a, b \leq Z\} \tag{8}$$

Among several MOEA we resort to the Non-dominated Sorting Genetic Algorithm (NSGA-II) to solve the application placement problem introduced in (7). NSGA-II generates Pareto optimal solutions for our Constrained Multi-objective Optimization Problem (CMOP). The proposed NSGA-II based algorithm goes through 4 main steps: (1) initialization, (2) selection, (3) reproduction, and (4) population update. At first, an initial solution population is randomly generated by the proposed algorithm. Then in the selection step, the solutions are ranked based on their quality and the best are selected. In the reproduction step, the NSGA-II based algorithm probabilistically combines high-quality solutions to generate possibly better new solutions by going through crossover and mutation operations. Then the selection and reproduction steps are repeated for certain number of iterations.

1) Initialization: Each solution s_z is in the form of the matrix in (2), representing one possible solution to the allocation problem. In this matrix, FNs and applications are combined, respecting the constraint of FNs capacity on number of applications to cache in (3). In the initialization phase the application placement on each FN is performed randomly to generate the set of initial solutions $\mathbf{S_0} = \{s_1, s_2, \ldots, s_Z\}_{\tau=0}$. More specifically, each $c_{m,p}$ in matrix $\mathbf{C}$ is randomly set to either 0 or 1, respecting the capacity condition. This procedure is performed for Z solutions as defined in $\mathbf{S_0}$.

2) Selection: After generating the initial population, the initial solutions are ranked based on their quality calculating the coverage area and overlapping area in (7) and the Pareto dominance as introduced in the Definition 1. Then, using the Pareto-dominance, all solutions in $\boldsymbol{S_0}$ (i.e., the solutions of the first generation) are compared with each other and given a rank. The solution that is not dominated is given the first rank, the solution that has been dominated once is given the second rank and so on.

To this aim, in order to consider the CMOP, a joint ranking function $\omega(s)$ has been defined operating in two steps. The solutions are ranked following a coverage area descending order at the first step, aiming at selecting as first the solutions able to cover a higher number of IoT sensors. A second step, involving the overlapping areas in introduced, where the top 25% solutions (i.e., the 25% solutions allowing to cover the highest number of sensors) are chosen and successively ordered in terms of increasing overlapping areas, i.e., from that having the lowest amount of overlapped areas to that with the highest value. This corresponds to say that the generated solutions in $\boldsymbol{S_0}$ are ordered by considering a Pareto dominance where the first have a joint higher coverage areas and lower overlapping areas.

3) Reproduction: The generated solutions in $\boldsymbol{S_0}$ are then used in the reproduction step, following the genetic operations of (i) binary tournament, (ii) crossover, and (iii) mutation.

In the binary operation, two of the solutions are randomly selected[1] from $\boldsymbol{S_0}$ and the one having a higher rank is stored in the mating pool. The mating pool, $\boldsymbol{S'}$, is a set of winner solutions that have comparatively high rank and are used for the reproduction procedure. If two solutions have the same rank (i.e., corresponding to have the same dominance value), one of them is selected randomly. The binary tournament continues until Z' solutions are stored in $\boldsymbol{S'}$.

Then the crossover operation is performed, where the solutions in $\boldsymbol{S'}$ are combined to generate possibly better solutions. In the crossover operation, two parent solutions, s_i and s_j, are selected randomly from the mating pool for generating the child solution s'_k by partially combining the parent solutions.

One of the parameters considered during the crossover operation is the crossover probability $\bar{P}_c$. Every time a new crossover has to be performed, a random value P_c is generated in the range $[0, 1]$; if $P_c \leq \bar{P}_c$, the crossover is performed. In case the crossover operation is performed, the algorithm randomly selects a crossover point χ_{cr} in the matrix defined in (2) corresponding to the element number in the matrix that divides the values to be crossed and those to be maintained.

Following this, the children s'_k is formed by alternating rows from s_i and s_j according to the crossover point in a way that the FN capacity constraint is not violated. Hence, the child s'_k is formed such that $c_{m,p}$, $\forall m \leq \chi_{cr}$ are taken from parent s_i and $c_{m,p}$, $\forall m > \chi_{cr}$ are taken from parent s_j. Hence,

$$s'_k = \begin{pmatrix} c^i_{1,1} & \cdots & c^i_{1,p} & \cdots & c^i_{1,P} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c^i_{\chi_{cr},1} & \cdots & c^i_{\chi_{cr},p} & \cdots & c^i_{\chi_{cr},P} \\ c^j_{\chi_{cr}+1,1} & \cdots & c^j_{\chi_{cr}+1,p} & \cdots & c^j_{\chi_{cr}+1,P} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c^j_{M,1} & \cdots & c^j_{M,p} & \cdots & c^j_{M,P} \end{pmatrix}, \quad (9)$$

where $c^i_{m,p}$ is a generic allocation derived for the parent solution s_i, $c^j_{m,p}$ is a generic allocation derived for the parent solution s_j, and χ_{cr} is the crossover point.

The child solutions are then added to the set $\mathbf{Q_0}$, and the crossover operation continues until Z'' children are generated, where $Z'' + Z' = Z$.

Finally, the mutation operation is performed, where each row of each solution is changed with probability $\bar{P}_{mu}$. In this phase, if the $\bar{m}$th row of the kth solution has to be mutated, its components $c^k_{\bar{m},p}$ are generated once again in a random way similarly to the Initialization step, still respecting the FN capacity constraint, i.e.,

$$\sum\nolimits_p c_{\bar{m},p} \leq C_{\bar{m}}$$

In order to take into account two possible approaches, we resorted to two possible mutation operations:

[1]The solutions are selected randomly, because randomness is an inherent property of GA

(a) **Centralized:** In this case we suppose that in the mutation operation the application placement can be randomly changed by respecting only the capacity constraint in (3), similarly to the Initialization phase. We are modeling the presence of a centralized orchestrator able to change the application placement on the FNs to be mutated while respecting the CMOP.

(b) **Distributed:** In this case the mutation operation on the $\bar{m}$th row is restricted to those applications placed in the nearby FNs, defined as those FNs having a distance lower than $2R_m$ with respect to teh FN to be mutated. This corresponds to say that $c_{\bar{m},p}$ can mutate from 0 to 1 only if $c_{\bar{m}',p}$ is equal to 1, and $d_{\bar{m}'\bar{m}} < 2R_m$, for all $F_{\bar{m}'}$. In this case we are modeling a scenario where the application exchange during mutation is limited only between nearby FNs, implementing a distributed approach, where the FNs exchange the applications among them. Two FNs are considered near if they have a distance lower than $2R_m$, that is supposed to be the inter-FN mesh-based network allowing the direct exchange of the applications among the FNs.

4) Population Update: Once the offspring population $\mathbf{Q_0}$ has been generated, the new solution population $\mathbf{S_1}$, should be formed for next generation. Thus, we define $\mathbf{\Psi_0} = \mathbf{S_0} \cup \mathbf{Q_0}$, as the aggregated solution population of the previous solution population and the offspring population. Then, the solutions in $\mathbf{\Psi_0}$ are ranked based on their dominance rank, as explained in reproduction step, and the high-quality solutions are added to $\mathbf{S_1}$ in a descending rank order until it reaches its maximum size to form the new solution population. Similarly, the solution population in the tth iteration, $\mathbf{S_t}$, is based on $\mathbf{\Psi_{t-1}} = \mathbf{S_{t-1}} \cup \mathbf{Q_{t-1}}$. The summary of the proposed NSGA-II based algorithm is shown in Algorithm 1.

Algorithm 1 The evolutionary-based application placement algorithm

Require: $\mathcal{U}$, $\mathcal{F}$
  Execute initialization phase and generate $\boldsymbol{S_0}$ having size Z
  for $t = 0$ to X **do**
    for all $s_z \in \mathbf{S_t}$ **do**
      Calculate A^C_p and A^O_p $\forall p$ using (7)
    end for
    Sort $\mathbf{S_t}$ based on the Pareto dominance relationship
    Use binary tournament to fill the mating pool
    Apply crossover on $\mathbf{S_t}$ and generate $\mathbf{Q_t}$
    Apply mutation on $\mathbf{Q_t}$
    Generate $\mathbf{\Psi_t} = \mathbf{S_t} \cup \mathbf{Q_t}$
    Sort $\mathbf{\Psi_t}$ based on Pareto dominance relationship
    Select the best Z solutions to form next generation
  end for
Ensure: Pareto Fronts of the multi-objective application placement problem

IV. Numerical Results

In order to test the proposed NSGA-II based approaches, in this Section, we introduce the numerical results obtained through computer simulations implemented in Matlab.

We consider a service area having size 200×200 meters, where the FNs, having the same coverage area equal to 25 m,

TABLE I
SCENARIO 1: VARIABLE FNS

Parameter	Value
Service Area	200×200
FN Coverage Area	25 m
Number of Sensors (S_n)	1000
Number of FNs (F_n)	$[10-60]$
FNs Capacity (C_n)	2
Number of Applications (K_p)	4
Number of GA Iterations	100
Number of simulation Trials	10
Crossover probability (P_c)	0.60
Mutation probability (P_{mu})	0.01

are scattered in a random position. Let's think as an example to a neighborhood willing to implement a smart-living infrastructure. It is worth to be noticed that both FNs and Sensors are placed in random positions hence no planning optimization is performed when obtaining the following results. The crossover $\bar{P}_c$ and mutation $\bar{P}_{mu}$ probabilities of the GA have been set to 0.60 and 0.01, respectively, after a careful optimization phase. Such values are similar to those used in similar contexts. For simplicity but without loss of generality, the iteration process of the genetic algorithm is assumed to be repeated 100 times, while we fixed to Z' equal to 10 the number of winner solutions for generating off springs. The crossover point χ_{cr} is randomly selected in the range $[1, P-1]$ at every crossover operation. All the simulations have been carried out for 10 trials, and then the results averaged, in order to ensure a higher reliability in the numerical results.

The numerical results are expressed in terms of average coverage areas and overlapping areas by comparing three approaches: the random approach, the centralized approach and the distributed approach. It is worth to be noticed that the average coverage area corresponds to the percentage of sensors we are able to serve averaged among all the deployed applications, while the average overlapping area correspond to the percentage of sensors having more than one FN that can serve them with a given application, averaged over all the deployed applications. The coverage area should be maximized, while the overlapping area should be minimized for avoiding resource wastage.

The random approach consider the case in which the applications are randomly placed in the network; this is considered as the benchmark solution. The Centralized NSGA-II approach where we optimize the application placement based on the NSGA-II algorithm by considering the presence of a centralized repository able to exchange the applications among every FN. The Distributed NSGA-II where we constraint the applications to be exchanged among nearby FNs.

We have performed the simulations for two scenarios. The first scenario is based on the presence of a fixed number of applications, equal to 4, while each FN can cache only 2 applications. We have instead changed the number of deployed FNs in the service area between 10 to 60. In Tab. I, the parameters for the first scenario are listed.

In Figs. 1 and 2 the numerical results are shown, where it is possible to notice that NSGA-II based solutions allow to

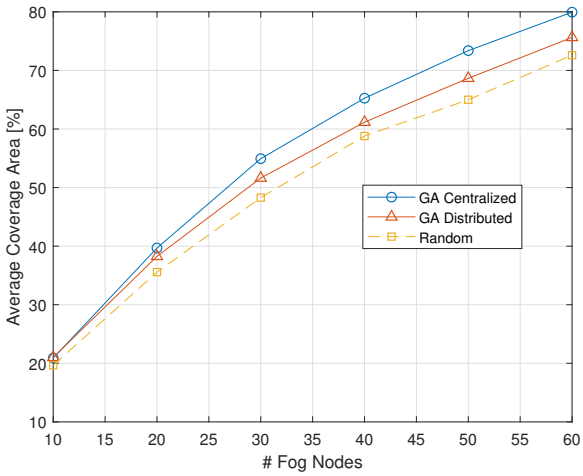

Fig. 1. Coverage Area for variable FNs

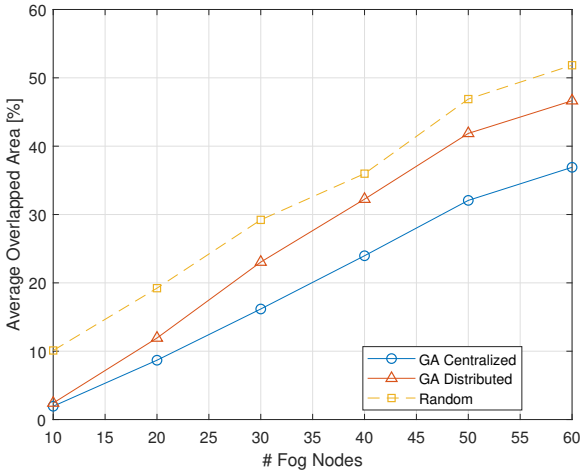

Fig. 2. Overlapping Area for variable FNs

achieve better performance in terms of both coverage areas and overlapping areas. By increasing of the number of FNs, the coverage area increases as expected by the increased number of FNs; however, it is possible to notice that GA-based solutions achieve better performance than the random solution. Moreover, it can be noticed that the Centralized NSGA-II allows to achieve better performance. This is expected since the Centralized approach could gain from a bigger set of possible solutions, while in the Distributed approach the solutions are constrained by the FNs proximity.

Even for the overlapping areas, leading to resource wastage, it is possible to notice that both GA approaches allow to achieve better performance. The increased number of FNs impact in a higher overlapping areas, where the Centralized approach performs better.

The second scenario we have considered is based on a variable number of applications to be deployed, between 4 and 8, while considering the capacity of each FN fixed to 2,

TABLE II
SCENARIO 2: VARIABLE APPLICATIONS

Parameter	Value
Service Area	200×200
FN Coverage Area	25 m
Number of Sensors (S_n)	1000
Number of FNs (F_n)	40
FNs Capacity (C_n)	2
Number of Applications (K_p)	$[4-8]$
Number of GA Iterations	100
Number of Simulation Trials	10
Crossover Probability (P_c)	0.60
Mutation Probability (P_{mu})	0.01

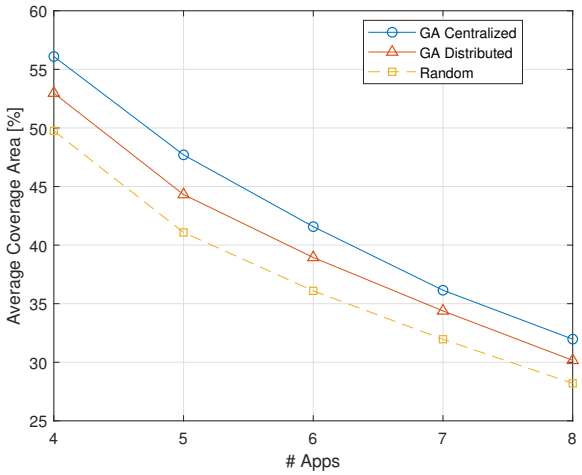

Fig. 3. Coverage Area for variable Applications

and the number of FNs to 40. The parameters are listed in the Table II.

Figs. 3 and 4, show the results in terms of coverage and overlapping areas for the Scenario 2. It is possible to notice that both NSGA-II approaches outperform the random approach by increasing the coverage area and minimizing the overlapping areas. In general, as expected, we can notice that a higher number of possible applications impact on a lower coverage areas; this is expected since we are constraining the number of applications per node to the same value, independently from the possible deployable applications. However we can notice that the overlapping areas are drastically decreased, leading to the fact that we are able to optimize at the best the system. In general, we are noticing that the Centralized approach allows to achieve better performance due to the higher flexibility in deploying the applications.

V. CONCLUSION

In this paper, an application placement approach for fog computing is introduced. We present the impact of an evolutionary algorithm over a distributed architecture, like the Fog Computing, for solving the problem of maximizing the number of serviced sensors in a given area. We firstly have formulated the application placement problem in a Fog Network, and then we proposed a meta-heuristic solution based on NSGA-II. Finally, we provided a framework for testing our policies. The numerical results show promising results by minimizing the overlapping area when changing number of applications and FNs for two different approaches.

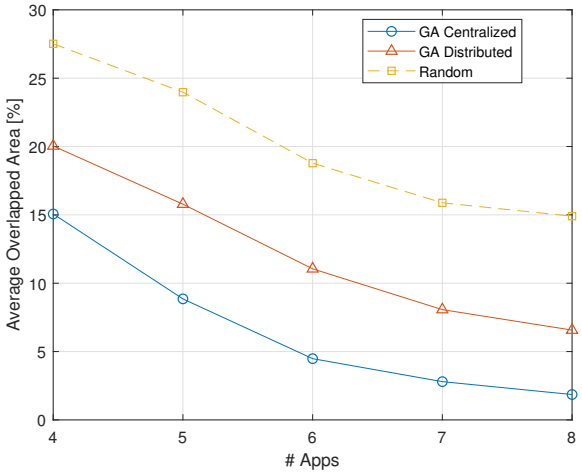

Fig. 4. Overlapping Area for variable Applications

REFERENCES

[1] S. M. A. Oteafy and H. S. Hassanein, "IoT in the fog: A roadmap for data-centric IoT development," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 157–163, Mar. 2018.

[2] B. McMillin and T. Zhang, "Fog computing for smart living," *Computer*, vol. 50, no. 2, pp. 5–5, Feb. 2017.

[3] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, pp. 25 445–25 454, 2017.

[4] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "iFogStor: An IoT data placement strategy for fog infrastructure," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, May 2017, pp. 97–104.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, Aug. 2012, pp. 13–16.

[6] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in iot fog computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7475–7484, Aug 2018.

[7] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1826–1857, Third Quarter 2018.

[8] A. Kumar, R. M. Pathak, and Y. P. Gupta, "Genetic-algorithm-based reliability optimization for computer network expansion," *IEEE Transactions on Reliability*, vol. 44, no. 1, pp. 63–72, March 1995.

[9] M. Mitchell, *An Introduction to Genetic Algorithms*, ser. A Bradford book. Cambridge, MA, USA: The MIT Press, 1998.

[10] B. M. Varghese and R. J. S. Raj, "A survey on variants of genetic algorithm for scheduling workflow of tasks," in *2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, March 2016, pp. 489–492.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.