2019 IEEE International Conference on Edge Computing (EDGE)

Meet Genetic Algorithms in Monte Carlo: Optimised Placement of Multi-Service Applications in the Fog*

Antonio Brogi, Stefano Forti
Department of Computer Science, University of Pisa, Italy
Emails: {antonio.brogi, stefano.forti}@di.unipi.it

Carlos Guerrero, Isaac Lera
Department of Computer Science, University of Balearic Islands, Spain
Emails: {carlos.guerrero,isaac.lera}@uib.es

***Abstract*—Managing multi-service applications on top of dynamic and heterogeneous Fog infrastructures is intrinsically challenging and requires suitable tooling to support decision-making.**

Indeed, bad service deployment decisions can lead to unsatisfactory application QoS, to waste of computing resources or money, and to application downtime.

In this paper, we illustrate how combining Genetic Algorithms with Monte Carlo simulations can considerably improve the efficiency of exhaustively searching for QoS-aware application deployments.

***Keywords*-Fog computing, service placement, genetic algorithms, Monte Carlo simulations.**

I. Introduction

Fog computing is expected to support new Internet of Things (IoT) applications, made from many distributed services, which interact with each other to achieve common goals. The Fog will have to suitably support such large, highly distributed software systems by meeting all their IoT and hardware requirements (e.g., RAM, CPU, HDD), as well as their possibly stringent QoS requirements in terms of latency and bandwidth availability. This will be crucial for those applications that are business-, mission- or life-critical.

Much literature has focussed on studying how to support the decision-making process when deploying multi-service applications to Fog infrastructures so to guarantee all their hardware, software and QoS requirements [1]. However, few efforts have been devoted to investigating how to optimally place multiple applications to Fog infrastructures at a time. Similarly, very few works have considered the interactions of application services with IoT devices as a parameter to lead the search of eligible deployments. Since the problem of placing one multi-service application to Fog infrastructures is provably NP-hard ([2], [3]), the same holds for placing multiple multi-service applications on Fog infrastructures.

Brogi et al. [4] proposed a combination of backtracking search with the Monte Carlo method to solve the application placement problem so to evaluate the QoS-assurance of eligible deployments of an application to a Fog infrastructure, also considering varying infrastructure conditions. The proposal was prototyped in a open-source tool FogTorchΠ, and enhanced with a cost model to estimate monthly operational costs of different eligible deployments [5]. FogTorchΠ determines eligible deployments based on hardware, QoS and IoT requirements of application services. However, as it relies on exhaustive search, it shows scalability limitations when dealing with large problem instances.

To tame the complexity of the application placement problem, Guerrero et al. [6] proposed the possibility to rely on genetic algorithms (GAs), by introducing a meta-heuristic approach to determine (sub-)optimal multi-service application deployments. Three different evolutionary algorithms (single objective GA, NSGA-II and MOEA/D) were compared, and the benefits of their adoption in tackling the Fog application placement problem were analysed. Other authors (e.g., Wen et al. [7], Mennes et al. [8]) proposed parallel versions of GAs to solve the problem of Fog service placement and reduce further the execution times. Whilst GAs considerably reduce the effects of exploring large search spaces to solve the considered problem, they are not able to capture the intrinsic uncertainty of Fog networks when deciding which are the best deployments. For these reasons, the combined use of GA and Monte Carlo simulations was recently studied by De Maio and Brandic [9] in Mobile Edge Computing scenarios, focussing on the problem of task offloading from deployed applications so to maximise user satisfaction and provider profit.

Hereinafter, we propose a novel combination of GAs with Monte Carlo simulations and we target the multi-service application placement in the Fog aiming at maximising QoS-assurance, and at minimising Fog resource consumption and operational costs. After extending FogTorchΠ accordingly, we assess the scalability of the new prototype over use cases that require deploying more than one application at a time. The rest of this paper is organised as follows. After formulating an optimisation model for our GA (Sect. II), we discuss its integration in the Monte Carlo framework of FogTorchΠ (Sect. III). We then illustrate the extended prototype at work over two lifelike examples (Sect. IV), and

* This work was partly supported by the project "*DECLWARE*" (PRA_2018_66) funded by the University of Pisa, and by the Spanish Government ("*Agencia Estatal de Investigacion*") and the European Commission ("*Fondo Europeo de Desarrollo Regional*") through grant number TIN2017-88547-P (MINECO/AEI/FEDER, UE).

978-1-7281-2708-8/19/$31.00 ©2019 IEEE
DOI 10.1109/EDGE.2019.00016

we finally conclude by highlighting some lines for future work (Sect. V).

II. Optimisation Model

A. Model Overview

We employ the model of [5] to describe the capabilities (and operational costs) of a Fog infrastructure I and the requirements of a multi-service IoT application A.

Fog infrastructures consist of the available Cloud nodes C and Fog nodes F, and IoT devices (or *Things*). Whilst Cloud nodes are assumed to feature unbounded hardware capabilities, Fog nodes are denoted by their hardware capabilities ($f.\mathcal{H}$) —viz., RAM, CPU, HDD— and connected Things. Furthermore, the description L of the infrastructure includes the latency and upload and download bandwidth (i.e., $\langle \ell_{n,n'}, \langle b_{n,n'}, b_{n',n} \rangle \rangle$) of end-to-end communication links among nodes and Things. Latency and bandwidth profiles of each end-to-end link are expressed as probability distributions.

Each application A consists of a set Γ_A of interacting services – each with its hardware ($\gamma.\overline{\mathcal{H}}$) and IoT requirements. Analogously, service-service and service-Thing interactions[1] are characterised by the QoS (i.e., maximum tolerated latency and minimum required bandwidth) needed to suitably support them, as specified in Λ_A. IoT requirements of each service are mapped to infrastructure Things, by specifying a binding ϑ_A.

Overall, a *deployment* $\Delta_A : \Gamma_A \rightarrow F \cup C$ of an application A to a Fog infrastructure I is eligible if it maps all components in Γ_A to nodes in F that can satisfy their hardware requirements or to nodes in C, and in such a way that the end-to-end links in L connecting deployment nodes can support the (service-service and service-thing) QoS requirements specified in Λ_A. For each eligible deployment, the prototype FogTorchΠ outputs the QoS-assurance (how likely is a deployment to meet the application QoS requirements), the resource consumption over Fog nodes, and an estimate of its monthly operational cost.

We extend the described model so to support the simultaneous deployment of multiple multi-service applications. To this end, we define the set $Apps$ and the set D containing the applications to be deployed and their eligible deployments, respectively. FogTorchΠ considers as a constraint the fact that application services deployed to the very same node consume its resources, and therefore it must hold that:

$$f.\mathcal{H} \preceq \sum_{\gamma \in S(f)} \gamma.\overline{\mathcal{H}} \quad \forall f \in F \tag{1}$$

where

$$S(f) = \{\gamma \mid \gamma \in \Gamma_A \wedge \Delta_A(\gamma) = f \wedge A \in Apps \wedge \Delta_A \in D\}.$$

[1]We assume that Fog and Cloud nodes can access directly connected Things as well as Things connected to neighbouring Fog nodes.

Namely, equation (1) states that services deployed to the same node must not exceed its overall hardware capacity. Similarly, also bandwidth represents a constraint and is considered consumable. Hence service-service and service-thing interactions mapped to a same link must not exceed its overall bandwidth capability.

By relying on these ingredients, we can define a fitness function to be maximised by the proposed GA:

$$Fit(D, I') = \frac{1}{3}\, QoSFit(D, I') + \frac{1}{3}\, CostFit(D, I') + \frac{1}{3}\, ResourceFit(D, I') \in [0, 1] \tag{2}$$

Such function can be evaluated for any given set of eligible deployments D in a given state I' of infrastructure I, sampled as per the probability distributions in L. In the rest of this section, we will detail the $QoSFit$, $CostFit$ and $ResourceFit$ objective functions[2] that consider network QoS, deployment operational costs and exploitation of Fog hardware resources, respectively.

B. Network QoS Objective

We first define the $QoSFit$ objective function that permits to measure how *largely well* service-service and service-thing interactions are supported by end-to-end communication links in a sampled state I' of the available Fog infrastructure, when deploying all $A \in Apps$ according to $\Delta_A \in D$. Indeed, we consider the distance between the latency and bandwidth values featured by the links in the infrastructure, and those required by the applications. Naturally, the lower the featured latency (the higher the featured bandwidth) the better. To deal with different scales between sampled infrastructure QoS metrics, we consider percentage differences instead of absolute values.

Overall, the network QoS objective is defined as the average of the percentage differences between all (service-service and service-thing) QoS interaction requirements and the QoS featured by the communication links to which they are mapped by $\Delta_A \in D$ (in state I' of I):

$$QoSFit(D, I') = \frac{1}{2}\, QoS^{\mathcal{B}}(D, I') + \frac{1}{2}\, QoS^{\mathcal{L}}(D, I') \tag{3}$$

The first addend ($QoS^{\mathcal{B}}(D, I')$) is the measure related to the bandwidth difference, while the second addend ($QoS^{\mathcal{L}}(D, I')$) is the measure related to the latency difference. It is worth noting that both values fall in $[0, 1]$, and that we consider higher values better than lower ones. We assume that services mapped to the same node n can interact with each other (and with Things connected to n) experiencing no latency and infinite bandwidth. Clearly, as

[2]By default, the three objectives – $QoSFit$, $CostFit$ and $ResourceFit$ – are weighted equally in the Fit function. FogTorchΠ users can set different weights depending on the objective(s) they aim at optimising.

per formula (2), also $QoSFit(D, I') \in [0, 1]$, and we will aim at maximising it.

C. Cost Objective

The cost model that we exploit is the one embedded[3] in FogTorchΠ. Such cost model relies on the function $cost(\Delta_A, \vartheta_A, A, I')$, which can estimate the monthly operational cost of an eligible application deployment Δ_A by considering both the cost of purchasing the needed (virtualised) hardware at each deployment Fog and Cloud nodes, as well as the cost of exploiting the Things (e.g., due to IoT data plans) bound by ϑ_A to the IoT requirements of A.

Clearly, we aim at minimising the overall deployment cost:

$$Cost(D, I') = \sum_{\Delta_A \in D} cost(\Delta_A, \vartheta_A, A, I') \tag{4}$$

Thus, we have that

$$CostFit(D, I') = \frac{cost_{max} - Cost(D, I')}{cost_{max} - cost_{min}} \tag{5}$$

where $cost_{min}$ and $cost_{max}$ represent the minimum and maximum cost of eligible deployments over the available infrastructure. Again, $CostFit(D, I')$ falls in $[0, 1]$, and higher values are better than lower ones.

D. Resource Usage Objective

For computing $ResourceFit$, we consider the set $\overline{F} \subseteq F$ of the Fog nodes that host at least one application service and we compute the aggregate average percentage of consumed RAM and CPU:

$$Resources(D, I') = \frac{1}{2} \sum_{\Delta_A \in D} \left(\frac{\sum_{\Delta_A(\gamma) \in \overline{F}} RAM(\gamma)}{\sum_{f \in \overline{F}} RAM(f)} + \frac{\sum_{\Delta_A(\gamma) \in \overline{F}} HDD(\gamma)}{\sum_{f \in \overline{F}} HDD(f)} \right) \tag{6}$$

This value is again a value in the range of $[0, 1]$ because the total RAM and storage capacities of the devices are always greater than the consumed resources. As for the cost, we aim at minimising the consumed Fog resources[4], hence, we define:

$$ResourceFit(D, I') = 1 - Resources(D, I') \tag{7}$$

This completes the information needed to understand how the $Fit(D, I')$ function is computed. In the next section, we describe the GA we employed, and how it was embedded in the Monte Carlo framework of FogTorchΠ.

[3]We refer the reader to the work by Brogi et al. [5] for all the details.

[4]Naturally, we could also aim at maximising them and this would be a trivial extension of the proposed solution.

III. GA Extension of FogTorchΠ

The FogTorchΠ prototype estimates the QoS-assurance of application deployments by relying on the Monte Carlo method and by repeatedly performing an exhaustive search of eligible deployments against varying (sampled) infrastructure conditions. We extend this approach and the prototype[5] by including a traditional GA search, which relies on the previously described model and fitness function.

The new prototype can therefore handle the simultaneous deployment of multiple multi-service applications to Fog infrastructures, taming the size of the search space according to a two-step approach that combines GA and Monte Carlo. The proposed solution works as follows:

(1) two solution deployments sets D^{best} and D^{worst} for all $A \in Apps$ are generated through the GA by considering two particular states I^{best} and I^{worst} of the Fog infrastructure, built so to be representative of I and to feature the best (lower latency and higher bandwidth values) and worst QoS (higher latency and lower bandwidth values), respectively, and

(2) both deployment sets D^{best} and D^{worst} are assessed against probabilistic variations of the infrastructure by employing the Monte Carlo method only to estimate their QoS-assurance (i.e., how likely they are to be eligible when infrastructure conditions vary).

The first step of the proposed genetic algorithm is performed over each considered representative infrastructure built when the algorithm starts. The algorithm represents solutions as mappings from all application services to their candidate deployment nodes. Initially, the GA generates a random population of such solutions and checks their eligibility. The size of the population is directly affected by the complexity of the solution (size of the individuals).

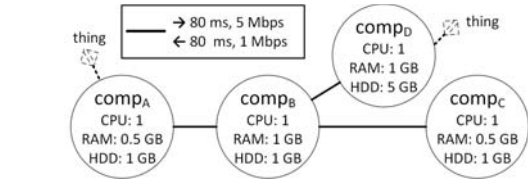
Afterwards, the initial population evolves along a specified number of generations emulating natural selection. Individuals are selected using a binary tournament operation and combined through a single-point crossover operation [10]. Resulting solutions are then mutated with a 0.1 probability (one application service is randomly selected and its deployment is changed to another random device). Finally, for the inclusion of the new solution into the population we have considered an elitism approach, i.e., new solutions with higher fitness replace those with the lowest fitness.

IV. Experimental Evaluation

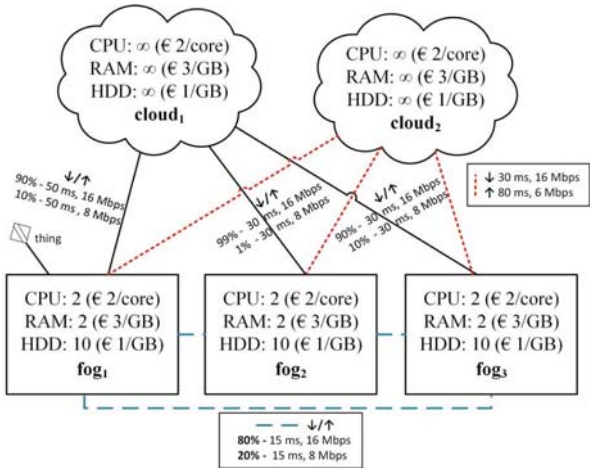
In this section, we compare the exhaustive and the genetic-based FogTorchΠ proposals over two different lifelike example use cases. A first, small-scale, example is inspired by the one described in [5], whilst the second one showcases how the GA solution outperforms the exhaustive search over a large-scale example.

[5]Code available at: https://github.com/di-unipi-socc/FogTorchPI/tree/genetic-algs.

The small-scale experiment considers deploying two instances of an application consisting of four interacting services (Figure 1(a)) to a Fog infrastructure including five nodes (two Cloud and three Fog nodes) with a *thing* IoT device connected to one of the available Fog nodes (Figure 1(b)), and required by services A and D of the application.



(a) Application



(b) Small-scale infrastructure

Figure 1. Multi-service application and Fog infrastructure.

The large-scale experiment differs from the first one in the size of the infrastructure[6], which consists of 200 Cloud and Fog nodes, organised according to a Barabasi-Albert topology [11]. It also differs because ten instances of the application of Figure 1(a) (viz., 40 services overall) are deployed. As a consequence, the solution space to be explored is much larger with respect to the first example, and the population size and number of generations increases accordingly (see Table I). Fog nodes feature hardware resources according to probability distributions of CPU, RAM and HDD. CPU cores are in the (ordered) set $\{2^i \mid i \in \mathbb{N}_4\}$ with associated element probabilities of $0.3, 0.3, 0.2, 0.1, 0.05$ and 0.05 respectively. RAM values are in the (ordered) set $\{2^i \mid i \in \mathbb{N}_5\}$ with associated element probabilities of $0.2, 0.4, 0.2, 0.1, 0.05$ and 0.05 respectively. Last, HDD values stay in $\{10, 20, 30\}$ with associated probabilities of $0.3, 0.4$ and 0.3 respectively.

[6]Configuration details of the large-scale experiment are available at https://github.com/di-unipi-socc/FogTorchPI/tree/genetic-algs/src/main/java/di/unipi/socc/fogtorchpi/experiments/EDGE2019/large

Table I
EXPERIMENTS CONFIGURATION

Parameter	Small-scale	Large-scale
Population size	40	50
Number of generations	20	100
Number of Monte Carlo runs	1000	100

A. Results

The output of our algorithms on the considered examples is a set of eligible solutions deployments. We leave to system administrators or to a multi-objective optimisation module the possibility to select one among those sets as the best candidate, also depending on the deployers' desiderata (i.e., target QoS-assurance, cost and resource consumption). Thus, the comparison between the algorithms is based on the analysis of their output solution sets (when it is possible to compute them), after running the Monte Carlo simulation to obtain the QoS-assurance of eligible deployments against network QoS variations. It is worth recalling that the exhaustive search version evaluates all the solutions for a number of infrastructure instances equal to the Monte Carlo runs. Naturally, the size of the solution set for the exhaustive search is always larger than the one for the genetic algorithm. Also, different solution sets can have the same values estimated by the Monte Carlo assessment.

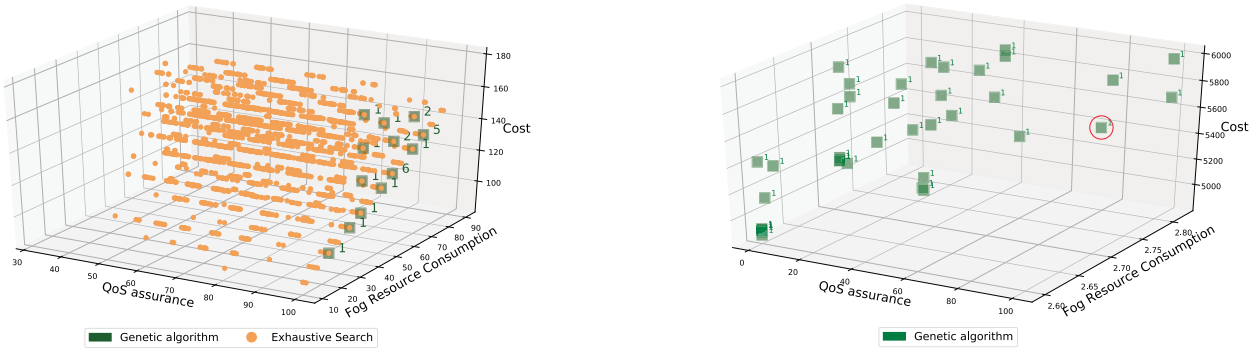
Figure 2(a) shows the results of running both the exhaustive search and the GA over the first example. Each point represents one of the output solutions, plot with respect to its QoS-assurance, cost and resource consumption. We have also included the number of GA solutions that have the same objective values, represented by the same green box in the plot along with a number indicating their frequency. Figure 2(b) shows the results of running the GA search over the second, large-scale example.

As discussed in Section II, our objective is to maximise QoS-assurance, minimise cost and minimise Fog resource consumption. Consequently, the best solutions are the ones placed closer to the bottom right front vertex of the plot. The execution time[7] for the GA version was of 1.5 and 40 seconds for the small- and large-scale examples, respectively. On the contrary, the Monte Carlo version lasted 67 seconds for the small-scale example and it did not successfully terminate on the large-scale one.

B. Analysis

The first conclusion from the experiment is that the solution space (diversity of solutions) is smaller in the case of the genetic search. In particular, the genetic search found 24 and 37 solutions for the small-scale and for the large-scale example respectively, while the exhaustive search found 26,623 solutions for the small-scale example.

[7]Experiments were run over a MacBook Pro i7 3Ghz 16GB RAM.



(a) Small-scale experiment

(b) Large-scale experiment

Figure 2. Solution sets for the small-scale and large scale examples.

In multi-objective optimisations, the most interesting solutions are those placed in the Pareto front. The Pareto front is the set of solutions in which one objective cannot be improved by any solutions without sacrificing at least one of the other objectives, i.e., the solutions that are not dominated by any other solution. Although the solution space of the genetic search is smaller, we can observe in Figure 2(a) that its solutions are concentrated in the Pareto front region, i.e., they are solutions with best objective values. On the contrary, the exhaustive solutions cover many cases with worse objective values.

For Figure 2(b), we can highlight that the solution search space is quite wide to offer the system administrator a flexible selection of solutions. For example, if the main concern is the QoS, the best solutions in the final population are the four in the top right corner of the plot, as they can guarantee 100% QoS-assurance. Thus, system administrators will probably select their best candidate in this set. Among those solutions, the circled solution set shows the lower cost at a lower Fog resource consumption.

Finally, we can observe that the GA solution its 43.89 time faster over the small-scale experiment and can handle the large-scale experiment, which the exhaustive search cannot.

V. Conclusions

The combination of the Monte Carlo method of FogTorchΠ with genetic algorithms has shown a substantial performance improvement with respect to the algorithm proposed in [5]. Indeed, our new methodology finds (a subset of) the best candidate deployment solutions in a significantly lower execution time ($40\times$ on a small-scale experiment).

Directions for future work include the possibility of further exploring the combination of Monte Carlo with GAs by applying newer and more sophisticate algorithms as in [6] and comparing their performance within the proposed methodology. Last but not least, it would be interesting to assess the quality of our predictions against actual or laboratory application deployments.

References

[1] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to Place Your Apps in the Fog – State of the Art and Open Challenges," *arXiv preprint arXiv:1901.05717*, 2019.

[2] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

[3] M. Nardelli, V. Cardellini, V. Grassi, and F. Lo Presti, "Efficient operator placement for distributed data stream processing applications," *IEEE TPDS*, 2019.

[4] A. Brogi, S. Forti, and A. Ibrahim, "Predictive analysis to support Fog application deployment," *Fog and edge computing: principles and paradigms*, pp. 191–222, 2019.

[5] ——, "Deploying Fog applications: How much does it cost, by the way?" in *CLOSER*, 2018, pp. 68–77.

[6] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures," *FGCS*, 2019.

[7] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, Mar 2017.

[8] R. Mennes, B. Spinnewyn, S. Latr, and J. F. Botero, "Greco: A distributed genetic algorithm for reliable application placement in hybrid clouds," in *Proc of the 5th Cloudnet*, 2016.

[9] V. D. Maio and I. Brandic, "Multi-objective mobile edge provisioning in small cell clouds," in *Proc. of the 10th ACM/SPEC ICPE 2019.*, 2019, pp. 1–12.

[10] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University computing*, vol. 15, no. 2, pp. 56–69, 1993.

[11] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *MASCOTS 2001*, pp. 346–353.