Latency Preserving Self-optimizing Placement at the Edge

Luca Ferrucci
ISTI-CNR, National Research Council
Pisa, Italy
luca.ferrucci@isti.cnr.it

Matteo Mordacchini
IIT-CNR, National Research Council
Pisa, Italy
matteo.mordacchini@iit.cnr.it

Massimo Coppola
ISTI-CNR, National Research Council
Pisa, Italy
massimo.coppola@isti.cnr.it

Emanuele Carlini
ISTI-CNR, National Research Council
Pisa, Italy
emanuele.carlini@isti.cnr.it

Hanna Kavalionak
ISTI-CNR, National Research Council
Pisa, Italy
hanna.kavalionak@isti.cnr.it

Patrizio Dazzi
ISTI-CNR, National Research Council
Pisa, Italy
patrizio.dazzi@isti.cnr.it

ABSTRACT

The Internet is experiencing a fast expansion at its edges. The wide availability of heterogeneous resources at the Edge is pivotal in the definition and extension of traditional Cloud solutions toward supporting the development of new applications. However, the dynamic and distributed nature of these resources poses new challenges for the optimization of the behaviour of the system. New decentralized and self-organizing methods are needed to face the needs of the Edge/Cloud scenario and to optimize the exploitation of Edge resources. In this paper we propose a distributed and adaptive solution that reduces the number of replicas of application services that are executed throughout the system, all the while ensuring that the latency constraints of applications are met, thus allowing to also meet the end users' QoS requirements. Experimental evaluations through simulation show the effectiveness of the proposed approach.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Information systems** → *Computing platforms.*

KEYWORDS

Self-optimizing application placement, Application model, Optimization Techniques for Resource Management, Edge computing

ACM Reference Format:
Luca Ferrucci, Matteo Mordacchini, Massimo Coppola, Emanuele Carlini, Hanna Kavalionak, and Patrizio Dazzi. 2021. Latency Preserving Self-optimizing Placement at the Edge. In *Proceedings of the 1st Workshop on Flexible Resource and Application Management on the Edge (FRAME '21), June 25, 2021, Virtual Event, Sweden.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3452369.3463815

FRAME '21, June 25, 2021, Virtual Event, Sweden

ACM ISBN 978-1-4503-8384-4/21/06…$15.00
https://doi.org/10.1145/3452369.3463815

1 INTRODUCTION

The adoption of Cloud computing is spreading at an incredible pace [1]. The transition is not only cost-effective for application providers, which avoid the set-up and most maintenance costs of the server-side infrastructure of services, but is also typically convenient for the end-users, who exploit always up-to-date applications via lightweight clients. However, not all modern applications are easily migrated onto Cloud computing infrastructures, as some application features and constraints, like latency-sensitiveness, can prevent a straightforward Cloud-based deployment and execution of services. Virtual reality applications can hardly be removed from their end-users without impairing the perceived QoE. Other examples of latency and bandwidth constraints are found among IoT-based applications, where a flow of raw data from a huge number of usually low-cost, low computational power sensors must be classified and processed in real time to react to changes in environment, such as in domotics [2]. Edge Computing is an attempt to overcome these limitations, it is a service computing paradigm that aims at bringing the computation as close as possible to the data producers and/or consumers (e.g., end-users). The direct bearing of this approach is to strongly limit the latency between the data producers/consumers and the computing resources. In the Edge Computing perspective the infrastructure is envisioned as revolving around a vertical computing continuum that goes from one (or more) Cloud(s) to a potentially large number of edge resources, geographically distributed over a usually broad area. The management of such a complex set of distributed, typically heterogeneous resources can be very burdensome and poses practical problems, as well as some interesting research challenges. Among those there is the non-trivial task of selecting the most adequate resources for a given application service, conditioned by the set of its users and their dynamically varying locations. Many approaches have been proposed so far to tackle the problem, some based on traditional optimization techniques whereas others exploiting data-driven approaches (e.g. machine-learning). There are solutions based on centralized solvers and others with a distributed nature. This paper proposes a decentralized, latency-aware solution for application placement at the edge. Application instances on the edge interact with one another in order to perform workload exchanges, lowering the overall resource usage while keeping the latency between the users and the edge under an application-dependant preset threshold. The remaining part of this paper is organized as follows. Section 2 contextualizes this work in the related scientific literature. Section 3

presents our formal definition of the problem and illustrates the approach we propose. Section 4 describes the experimental evaluation of the proposed solution. Finally, Section 5 draws concluding remarks and highlights future work directions.

2 RELATED WORK

In recent years, many solutions have been proposed for optimizing the behaviour of Edge-based systems [3, 4]. The common feature of these approaches is to limit the communications to centralized Cloud servers by moving services and/or data closer to users, thus enhancing the quality of the services offered to them. To achieve this result, decentralized and/or self-organizing methods are exploited.

Some solutions are focused on data-driven mechanisms. In this case, the data is moved in the system to make it easy for the users to access it [5–29]. In particular, a common strategy is to shorten the distance between data producers or storage and consumers to reduce the latency needed to respond to the users' requests. Suitable distributed strategies for data placement and replication are proposed in works like Aral and Ovatman [30] and Li et al. [31]. Differently from these approaches, we deal more with computing requests and find an optimal placement/replication of computing services.

Other solutions are more concerned with the placement of applications closer to the data to use and/or their respective users. For instance, the work described in [32] exploits a self-organizing clustering of the Edge data producer devices. Clusters are used to identify the characteristics of the data produced by the various groups of devices and, as a consequence, to optimally place the users' data analysis applications. Maia et al. [33] propose two methods, one based on genetic algorithms and one based on a Mixed-Integer Linear Programming heuristic. However, these two solutions are designed to work offline and with complete knowledge of the set of potential applications and computing nodes, i.e., a simpler scenario than the one we consider in this work. Ning et al. [34] propose a distributed, online solution for service placement at the Edge. Their solution is based on a probabilistic optimization method for computing the utility (in terms of cost, storage capability, and latency) of service migration and placement to determine the service placement configurations. De Lira et al. [35] implement a genetic algorithm that exploits the concept of user coverage to place latency-sensitive services according to the supposed location of end-users to maximize their quality of experience.

Different from the previous set of works, the method we propose in this paper is related to the optimization of the Edge resource consumption levels. One of the solutions available in this field is proposed by Kavalionak et al. [36, 37]. The authors design a distributed solution, where geographically dispersed Edge devices are able to coordinate, both among themselves and with a remote server. In this way, the devices (surveillance cameras, in their use case scenario) are able to fulfill their activities more efficiently by sharing and balancing the required computational costs. Beraldi et al. propose CooLoad [38], a distributed, cooperative load-balancing scheme. In this model, an Edge data center stops processing incoming requests when they exceed a given threshold. To balance the load, new requests are to re-directed to another adjacent data center. Carlini et al. [39] have designed a completely decentralized system, where autonomous entities in a Cloud Federation are able to communicate to exchange services. The objective of the proposed approach is to maximize the profit of the whole Federation by mixing collaborative and selfish behaviours of the entities of the system. Indeed, the devices collaborate by mutually exchanging data and services. However, they make only the exchanges that increase their own subjective revenues, eventually leading the system to optimize its overall profits. Rather than balancing the load or maximizing revenues, in this paper, we propose a solution that optimizes the resource usage on the Edge by limiting the number of redundant replicas of the applications that are executed in the system. As we explain in detail in the following sections, to achieve this result, point-to-point interactions between edge mini clouds are used to move the users' requests from one minicloud to another, hence allowing to shut down some of themini clouds of the same application. User requests are moved only if the users' latency QoS requirements are guaranteed. In this way, we avoid wasting resources on useless replicas without violating the QoS constraints.

3 PROPOSED SOLUTION

In this section, we provide a detailed description of our proposed approach. We first give a formal definition of the problem we face in this paper. Then, we describe how we have tackled the problem, using autonomous interactions between entities at the Edge.

3.1 Problem Definition and Model

In the context of this paper, we consider that the system at its edges is made of entities that are termed as *edge mini-clouds* (EMC). An EMC represents an entity located at the Edge. This entity is in charge of supervising a set of other smaller, simple devices at the Edge. In this paper, the resources handled by an EMC are the sum of all the resources available within it. The only direct interactions that could happen in the system are the ones between different EMCs. Users of the system request the services provided by a set of application. Each application offers a different type of service. Several instances (or replicas) of an application can be deployed on several EMCs, on the basis of users' requests and QoS constraints. Each EMC is in charge of supervising the execution (and all the related optimization aspects) of the application replicas it receives.

On the basis of this description, we can define that the system that we study in this paper is made of the following entities:

- M edge miniclouds $\{EMC_1, .., EMC_M\}$
- N types of applications $\{A_1, .., A_N\}$

In addition, we consider that these entities have the following characteristics:

- Each edge mini-cloud EMC_j has a specific capacity of resources (or maximum admittable execution cost) $C_j, j \in [1, \ldots, M]$.
- For each application A_i, there is a set of instances, or *replicas* $R_i = \{a_{ij} | j \in S \subseteq [1, \ldots, M]\}$, where a replica a_{ij} runs on the edge minicloud EMC_j.
- Each application A_i has an associated set of users $\mathcal{U}_i$, $|\mathcal{U}_i| = U_i \geq 1$
- Each application type A_i is characterized by a fixed cost C_i^{fix} and a variable cost. We assume that A_i pays a cost C_i^{var} for

each of the users it serves, so that the total variable cost is computed as $C_i^{var} \cdot U_i$.
- To respect the QoS agreed with its users, each application A_i has a specific maximum latency requirement L_i.

As highlighted in the Introduction, in this paper we face the problem of minimizing the total number of running replicas in the system. This operation should be done by a proper choice of the replicas to maintain active and the selection of the edge miniclouds where they have to be allocated. The selection and allocation processes should ensure that all the users are served, without violating their latency requirements. Moreover, the edge mini-clouds capacity limits should be respected.

Using the notation introduced in this section, this problem can be formally described as an optimization problem, using the following mathematical formulation:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij} \\ \text{s.t.} \quad & \sum_{i} (C_i^{fix} + C_i^{var} u_{ij}) a_{ij} \leq C_j \ \forall j, \\ & \sum_{j} u_{ij} a_{ij} = U_i \ \forall i, \\ & l_{ij} a_{ij} \leq L_i \ \forall i, j, \\ & a_{ij} \in \{0, 1\}, \\ & u_{ij} \in \mathbb{N} \end{aligned} \tag{1}$$

where a_{ij} is a replica of application i on EMC_j (active = 1, inactive/turned off = 0), U_i is the total number of users of application A_i, C_j is the maximum capacity of EMC_j, u_{ij} is the number of users of application A_i handled by the replica running on EMC_j, l_{ij} is the latency experienced by the replica a_{ij}, while L_i is the maximum allowable latency for any replica of A_i .

The objective function minimizes the total number of replicas of applications that are running in the system. The first constraint ensures that the total cost brought by the applications (and their users) running on each EMC does not exceed the EMC's maximum allowable cost. The second constraint guarantees that all the applications' users are served by the active replicas, while the third constraint imposes that the active replicas should respect the maximum latency assigned to their respective applications. Note that, by assuming $U_i \geq 1$, the second constraint also implies that $\sum_j a_{ij} \geq 1 \ \forall i$, i.e. there is at least an active replica for each application A_i.

3.2 Solution Description and Pseudo-code

In the previous section, the problem we face in this paper has been described as an optimization problem. It is characterized by a high complexity, and a general solution would require a global knowledge. Both these points force to exploit a different approach, that is also capable of taking into account the distributed and dynamic nature of the computing environment at the Edge.

In the rest of this section, we detail our proposed solution. This solution is a self-optimization scheme. Indeed, it is based on the autonomous actions of the entities (the EMCs) that compose the system. The overall collective behaviour of these entities eventually leads the whole system to optimize its behaviour.

The description proposed in this section follows the pseudocode given in Algorithm 1. The algorithm describes the step performed by a generic EMC E_j. Specifically, we consider that E_j has a set $N$of neighboring EMCs. These neighbors are the EMCs that are in the communication range of E_j. We assume that the latency between E_j and any other $E_k \in N$ is $L(E_j, E_k)$. Moreover, let A_j be the set of applications running on E_j. Each application $a_{ij} \in A_j$ is characterized by a maximum agreed latency L_i, a set of users u_{ij}, which have a maximum experienced latency l_{ij}.

Algorithm 1 Actions performed by an EMC E_j at each cycle

N = set of neighbors of E_j
$L(E_j, E_k)$ = latency between E_j and $E_k \in N, \forall k$
A_j = set of apps running on E_j
L_i = max admitted latency for users of $a_i \in A_j, \forall i$
l_{ij} = max latency for users of $a_i \in A_j, \forall i$
u_{ij} = set of users of $a_i \in A_j, \forall i$

Randomly choose $a_i \in A_j$
Let $N_i = \{E_k \in N | L(E_j, E_k) + l_{ij} \leq L_i\}$
Randomly choose $E_k \in N_i$
O_k = resource occupancy of E_k
C_k = max capacity of E_k
Request from E_k its O_k, C_k and A_k
if $\exists a_i \in A_k$ **and** $O_k + C_i^{var} u_{ij} \leq C_k$ **then**
  move all u_{ij} on E_k
  turn off a_i on E_j
end if

In order to contribute to a solution of the overall problem, at regular intervals E_j performs the following actions. It randomly chooses one of its application $a_{ij} \in A_j$. Then, it selects the subset $N_i \subseteq N$ of its neighbors that respect the condition $L(E_j, E_k) + l_{ij} \leq L_i$. This condition allows to determine which are the neighbors eligible for receiving the users of a_{ij} without violating the latency constraint L_i.

At this point, E_j randomly selects and contacts a neighbor $E_k \in N_i$. E_k responds by communicating to E_j the list A_k of the application replicas it is running, its actual resource occupancy O_k, and its maximum capacity C_k. E_j checks whether A_k contains a replica a_{ik} of A_i, and that the cost of adding the users u_{ij} to E_k does not exceed C_k, i.e. $O_k + C_i^{var} u_{ij} \leq C_k$. In case these conditions are satisfied, E_j redirects all the users u_{ij} to use the replica on E_k. Then, it shuts down its application replica a_{ij}.

In our algorithm the entities in the system collaboratively try to detect application replicas that can be considered redundant, then regroup the users of that application in order to use fewer replicas. The applications' latency constraints are never violated by the regrouping, while the progressive elimination of redundant replicas and the limits imposed by the EMCs' capacities eventually lead the system to converge.

In this paper, such convergence is not formally demonstrated, leaving it as a future work. A possible technique is to represent the algorithm with a workflow, using modelling languages such as StateFlow/StateCharts. Then, it is possible to convert automatically such workflows in linear temporal-logic formulae and exploiting

a Model Checker tool, as in [40], formally verify a property of the whole system, that a terminal state is always eventually reachable following every computation path.

4 EXPERIMENTAL EVALUATION

In order to validate our approach, we simulated a target scenario using PureEdgeSim [41], a discrete-event simulator for Edge environments. The simulated scenario consists of a federation of EMCs. Each EMC is composed of an heterogeneous set of resource constrained edge devices and servers, able to host various types of applications on behalf of a set of users. Addressing such complexity and heterogeneity is beyond the scope of this work, so each EMC is simulated as a single aggregated entity with a PureEdgeSim *Datacenter* object with a quantity of resources that is the sum of the resources of the devices and servers that compose it. Each user device is simulated with a PureEdgeSim *EdgeDevice* object (e.g., a tablet or a smartphone). At the beginning of the simulation, each user device offloads a single application instance (represented by a *Task* object in PureEdgeSim) to the closest EMC. If an instance with the same application type already exists on the same EMC, the user is added to the set of users served by the preexisting instance, increasing its footprint. In this work, mobility of user devices, energy consumption of devices, and dynamic churn of users are not simulated. The number of users is fixed at the beginning of each experiment, varying in the set {60, 120, 180}. Each user device is generated at the beginning of each experiment and placed randomly in a simulated bi-dimensional rectangular space of 200x200 metres. In our experiments the number of EMCs is fixed to 4. They are placed at predefined locations inside the simulation space, at the vertices of a square of coordinates {50,50}, {50,150}, {150,50} and {150,150}. An investigation of the behaviour of our proposed solution when varying the number of EMCs is beyond the aim of this paper; similarly, for this preliminary work, a confrontation of our algorithms with the best practice algorithms existent in literature is not performed, since our aim is to show the effectiveness and validity of realization of our algorithm and if it reaches its goals, expressed in the optimization function. Such confrontation will be performed as a future work. A set of assumptions are made, accordingly to our model: (1) any application type can be run on any EMC, (2) there is at most a single instance of each application type on each EMC, and (3) all the EMCs belonging to the federation are able to communicate with each others and with all user devices, to provide full connectivity.

The types of resources simulated for EMC and applications are limited to three: *VCPU*s, *Ram* and *Bandwidth*. They represent, respectively, the number of VCPUs, the amount of Ram and the amount of network bandwidth required by the VMs/ Containers of a certain application type. For an EMC, instead, they represent, respectively, the maximum declared capacity of VCPU, the maximum declared capacity of Ram, in million of bytes, and the maximum amount of aggregated Bandwidth over communication links to/from the EMC to other EMC or user's device.

There are 4 different application types, each characterised by a different resource footprint, which is composed by a fixed cost C^{fix} (the amount of resources required to start an application instance), and a variable cost C^{var} (for each additional user). The different application types and their costs are described in Table 1 and Table 2. *Computational Bound*, *Memory Bound* and *I/O Bound* application types simulate a computational intensive, memory intensive and networking intensive application, respectively. In this perspective "intensive" means having double the requirements of VCPUs, Ram or Bandwidth than the basic *Balanced* application type.

Table 1: C^{fix} for each application type

Type	VCPU	Ram (Mbyte)	BW (Mbit/s)
Balanced	1	200	20
Comp Bound	2	200	20
Mem Bound	1	400	20
I/O Bound	1	200	40

Table 2: C^{var} for each application type, for user

Type	VCPU	Ram (Mbyte)	BW (Mbit/s)
Balanced	1 each 10 users	20	2
Comp Bound	1 each 5 users	20	2
Mem Bound	1 each 10 users	40	2
I/O Bound	1 each 10 users	20	4

For each application type, we have also specified 3 different values for the max network latency: such values vary in the set 0.2, 0.3, 0.5 seconds, and are aimed at testing the behaviour of the experiments with applications with different latency sensitivities. Thus, the number of distinct types of applications used in our experiments is 12 (4 types of applications times 3 different latency constraints). Consequently, each EMC has an equivalent amount of resources, allowing it to be able to host 80-100% of the replicas of the different type of applications at the beginning of each experiment; the capacity of each EMC is { VCPU = 24, Ram = 6Gbytes, Bandwidth = 600Mbits/s}. A simulated latency is calculated for each user's device, using the function

$$f_{latency}(d, EMC) = ChanLat_{fix} + dist(d, EMC) * C_{Lat}$$

This function is composed of two parts:

- a fixed part, $ChanLat_{fix}$, which is dependent from the communication channel type; in our experiments, it is fixed at 0.1 seconds and also includes the part of the latency that depends from the bandwidth of the channel and the dimension of the sent packet; this part is actually negligible for our experiments and thus considered accounted for
- a linear part, proportional to the Euclidean distance $dist(d, EMC)$ between the EMC hosting the instance of the serving application and the user's device d. The C_{Lat} predefined constant represents the latency cost for each unit of distance and is fixed in our experiments at 0.0005.

Our simulation is divided into iterations occurring at discrete time intervals. Each EMC is an agent that, once per iteration, acts as initiator for the algorithm. In our scenario, an iteration is started every 30 seconds, and every experiment has a simulated duration

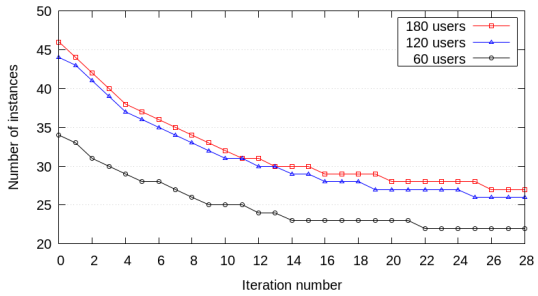

Figure 1: Total number of instances, per iteration

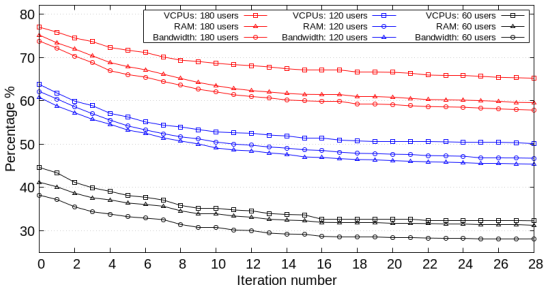

Figure 2: Resources consumption of the federation, per iteration

time of 15 minutes, so the number of iteration is fixed to 29. All the presented values are the average of 10 independent runs, to minimize the effects of randomization introduced by the algorithm.

The graph in Figure 1 shows the speed of convergence and the effectiveness of our algorithm in optimizing the objective function, with a varying number of users. As we expected, with the small federation used in our experiments, even with a limited number of users (120 - 180) the number of instances initially hosted by the federation is close to the maximum the whole platform can bear, that is 48; only with 60 users the initial number of instances is 34. The graph shows a high speed of convergence, also for the scenario with a low number of users, mainly due to the limited number of EMCs and the high number of initial application instances, which increases the probability that the randomly chosen EMC to contact actually hosts an instance of the same randomly chosen application type, and the swap does not violate latency constraints. The convergence is reached between 12-15 iterations, with a reduction in the number of instances of about 45% in average for 120 and 180 users, reduced to 25% for 60 users. This is an almost ideal scenario, as we expect lesser reduction in number of instances and slower speed of convergence for scenarios with (1) a greater number of EMCs, keeping fixed the number of users, or with (2) lower maximum latency. Both conditions decrease the probability of being able to move users to another EMC in order to coalesce app instances.

The graph in Figure 2 shows the reduction in resources consumption for all resource types and the entire federation, varying the number of users. Analyzing the graph, we can see that the reduction is more consistent during the first 10-15 iterations of the algorithm: it is an expected result, due to the corresponding behaviour in the

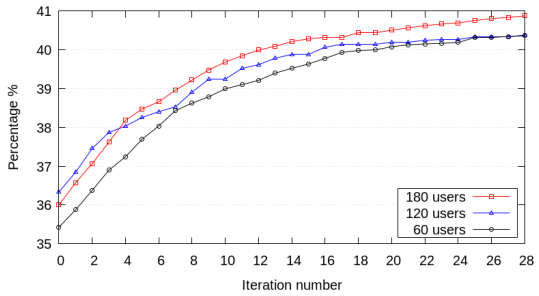

Figure 3: Average latency, per iteration

number of application instances that switched. We shall remind that the asymptotic resource cost is not linear with the number of users, as the fixed costs are proportional to the number of EMCs and their allocation is constrained by the actual set of communication latencies, that will generally prevent full utilization of instances.

Finally, the graph in Figure 3 shows the average measured network link latency, in percentage respect to the average of maximum network link latencies, for all the application types. We know from the simulator that no latency constraint is exceeded, hence it makes sense to look at the aggregation of results with respect to the varying number of users. We can observe analyzing the results that, at the beginning of the simulation, the average measured latency is about the 36% of the average maximum latency, as every user offloads its application to the closest EMC. While choosing the locally minimal latency ensures the minimum average latency overall, it does not lead to a minimization of the overall resources consumption or of the number of application instances. The two metrics of average communication latency and resource usage are actually in conflict, and optimizing the second while subject to the constraint of the first needs to achieve a trade-off. The graph shows this trade-off in the simulation. The average network latency increases as users are moved to other EMCs by the resource optimization algorithm, as opposed to previous two graphs that show a reduction in resource consumption and number of the application instances.

Analyzing the experimental results, we can conclude that our algorithm is suitable to reach a sound trade-off, at least in the most common scenarios. The results document a good speed of convergence (achieving most of the resource savings within the first 15 iterations in our experiments).

5 CONCLUSIONS AND FUTURE WORK

This paper presents a solution for application placement in an edge computing environment based on a fully decentralized approach. It works by performing inter-edge exchanges with the objective of reducing the resource usage, while safeguarding the QoE of applications by keeping the communication latency below application dependant-thresholds. The paper gives a formal definition of the constraints and the objectives, as well as the pseudo code of the proposed approach. An experimental evaluation via simulation shows the viability and validity of the solution in the most common scenarios, where the number of instances and the overall resources consumption of a set of application types in an Edge Federation is minimized without violating the latency constraints.

While the solution is quite a promising one, there is space to improve the results in the near future. It is worth e.g. considering alternative local search criteria and heuristics for the selection criteria of the EMC and application for the swap proposal, which currently is a plain random choice. This may improve the asymptotic cost savings and is likely to improve the achieved savings as well as the convergence speed of our algorithm. We also plan to provide a complexity analysis of the approach, a more detailed experimental evaluation including user mobility and variations in the edge resources, and an evaluation of the impact on energy consumption.

ACKNOWLEDGMENTS

This work has been partially supported by the European Union's Horizon 2020 Research and Innovation program, under the project ACCORDION (Grant agreement ID: 871793) and by the OK-INSAID project funded by the Italian Ministry of Education and Research (MIUR) under grant agreement No. RS01_00917.

REFERENCES

[1] C-H. Youn, M. Chen, and P. Dazzi. *Cloud Broker and Cloudlet for Workflow Scheduling*. Springer, 2017.

[2] V. Miori, D. Russo, and L. Ferrucci. Interoperability of home automation systems as a critical challenge for iot. In *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, pages 1–7. IEEE, 2019.

[3] F. Salaht, F. Desprez, and A. Lebre. An overview of service placement problem in fog and edge computing. *ACM Comput. Surv.*, 53(3), 2020.

[4] G. Z. Santoso, Y-W. Jung, S-W. Seok, E. Carlini, P. Dazzi, J. Altmann, J. Violos, and J. Marshall. Dynamic resource selection in cloud service broker. In *2017 International Conference on High Performance Computing & Simulation (HPCS)*, pages 233–235. IEEE, 2017.

[5] M. Mordacchini, P. Dazzi, G.Tolomei, R. Baraglia, F. Silvestri, and S. Orlando. Challenges in designing an interest-based distributed aggregation of users in p2p systems. In *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pages 1–8. IEEE, 2009.

[6] M. Mordacchini, A. Passarella, M. Conti, S. M. Allen, M. J. Chorley, G. B. Colombo, V. Tanasescu, and R. M. Whitaker. Crowdsourcing through cognitive opportunistic networks. *ACM Trans. Auton. Adapt. Syst.*, 10(2), June 2015.

[7] R. Bruno, M. Conti, M. Mordacchini, and A. Passarella. An analytical model for content dissemination in opportunistic networks using cognitive heuristics. In *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2012.

[8] M. Mordacchini, M. Conti, A. Passarella, and R. Bruno. Human-centric data dissemination in the iop: Large-scale modeling and evaluation. *ACM Trans. Auton. Adapt. Syst.*, 14(3), February 2020.

[9] J. Patman, P. Lovett, A. Banning, A. Barnert, D. Chemodanov, and P. Calvam. Data-driven edge computing resource scheduling for protest crowds incident management. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8, 2018.

[10] A. Lulli, L. Ricci, E. Carlini, and P. Dazzi. Distributed current flow betweenness centrality. In *IEEE Ninth International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2015.

[11] R. Baraglia, P. Dazzi, B. Guidi, and L. Ricci. Godel: Delaunay overlays in p2p networks via gossip. In *IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 1–12. IEEE, 2012.

[12] C. Gennaro, M. Mordacchini, S. Orlando, and F. Rabitti. Mroute: A peer-to-peer routing index for similarity search in metric spaces. In *5th VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2007)*, 2007.

[13] M. Marzolla, M. Mordacchini, and S. Orlando. A p2p resource discovery system based on a forest of trees. In *17th International Workshop on Database and Expert Systems Applications (DEXA'06)*, pages 261–265. IEEE, 2006.

[14] R. Baraglia, P. Dazzi, M. Mordacchini, L. Ricci, and L. Alessi. Group: A gossip based building community protocol. In *Smart spaces and next generation wired/wireless networking*, pages 496–507. Springer, Berlin, Heidelberg, 2011.

[15] E. Carlini, M. Coppola, P. Dazzi, D. Laforenza, S. Martinelli, and L. Ricci. Service and resource discovery supports over p2p overlays. In *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pages 1–8. IEEE, 2009.

[16] M. Danelutto, P. Dazzi, et al. A java/jini framework supporting stream parallel computations. In *PARCO*, pages 681–688, 2005.

[17] A. Lulli, E. Carlini, P. Dazzi, C. Lucchese, and L. Ricci. Fast connected components computation in large graphs by vertex pruning. *IEEE Transactions on Parallel and Distributed systems*, 28(3):760–773, 2016.

[18] M. Bertolucci, E. Carlini, P. Dazzi, A. Lulli, and L. Ricci. Static and dynamic big data partitioning on apache spark. In *PARCO*, pages 489–498, 2015.

[19] M. Mordacchini, A. Passarella, and M. Conti. A social cognitive heuristic for adaptive data dissemination in mobile opportunistic networks. *Pervasive and Mobile Computing*, 42:371–392, 2017.

[20] L. Ferrucci, L. Ricci, M. Albano, R. Baraglia, and M. Mordacchini. Multidimensional range queries on hierarchical voronoi overlays. *Journal of Computer and System Sciences*, 2016.

[21] F. Baiardi, A. Bonotti, L. Ferrucci, L. Ricci, and P. Mori. Load balancing by domain decomposition: the bounded neighbour approach. In *Proc. of 17th European Simulation Multiconference*, pages 9–11, 2003.

[22] A. H. Payberah, H. Kavalionak, A. Montresor, J. Dowling, and S. Haridi. Lightweight gossip-based distribution estimation. In *2013 IEEE International Conference on Communications (ICC)*, pages 3439–3443. IEEE, 2013.

[23] R. Baraglia, G. Capannini, P. Dazzi, and G. Pagano. A multi-criteria job scheduling framework for large computing farms. *Journal of Computer and System Sciences*, 79(2):230–244, 2013.

[24] G. F. Anastasi, E. Carlini, and P. Dazzi. Smart cloud federation simulations with cloudsim. In *Proceedings of the first ACM workshop on Optimization techniques for resources management in clouds*, pages 9–16, 2013.

[25] M. Coppola, P. Dazzi, A. Lazouski, F. Martinelli, P. Mori, J. Jensen, I. Johnson, and P. Kershaw. The Contrail approach to cloud federations. In *Proceedings of the International Symposium on Grids and Clouds (ISGC'12)*, volume 2, page 1, 2012.

[26] R.G. Cascella, L. Blasi, Y. Jegou, M. Coppola, and C. Morin. Contrail: Distributed Application Deployment under SLA in Federated Heterogeneous Clouds. In A. Galis and A. Gavras, editors, *The Future Internet*, pages 91–103, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[27] E. Carlini, L. Ricci, and M. Coppola. Integrating centralized and peer-to-peer architectures to support interest management in massively multiplayer on-line games. *Concurrency and Computation: Practice and Experience*, 27(13):3362–3382, 2015.

[28] L. Ricci, L. Genovali, E. Carlini, and M. Coppola. Aoi-cast in distributed virtual environments: an approach based on delay tolerant reverse compass routing. *Concurrency and Computation: Practice and Experience*, 27(9):2329–2350, 2015.

[29] E. Carlini, L. Ricci, and M. Coppola. Flexible load distribution for hybrid distributed virtual environments. *Future Generation Computer Systems*, 29(6):1561–1572, 2013.

[30] A. Aral and T. Ovatman. A decentralized replica placement algorithm for edge computing. *IEEE Trans. on Network and Service Management*, 15(2):516–529, 2018.

[31] Chunlin Li, YaPing Wang, Hengliang Tang, Yujiao Zhang, Yan Xin, and Youlong Luo. Flexible replica placement for enhancing the availability in edge computing environment. *Computer Communications*, 146:1–14, 2019.

[32] P. Dazzi and M. Mordacchini. Scalable decentralized indexing and querying of multi-streams in the fog. *Journal of Grid Computing*, 18(3):395–418, 2020.

[33] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro. Optimized placement of scalable iot services in edge computing. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 189–197, 2019.

[34] Zhaolong Ning, Peiran Dong, Xiaojie Wang, Shupeng Wang, Xiping Hu, Song Guo, Tie Qiu, Bin Hu, and Ricky Kwok. Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[35] V. M. de Lira, E. Carlini, and P. Dazzi. Polar: Geographic placement optimization for latency sensitive applications. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 361–362. IEEE, 2019.

[36] H. Kavalionak, C. Gennaro, G. Amato, C. Vairo, C. Perciante, C. Meghini, and F. Falchi. Distributed video surveillance using smart cameras. *Journal of Grid Computing*, 17(1):59–77, 2019.

[37] H. Kavalionak, C. Gennaro, G. Amato, and C. Meghini. Dice: A distributed protocol for camera-aided video surveillance. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 477–484. IEEE, 2015.

[38] R. Beraldi, A. Mtibaa, and H. Alnuweiri. Cooperative load balancing scheme for edge computing resources. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 94–100. IEEE, 2017.

[39] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini, and A. Passarella. Self-optimising decentralised service placement in heterogeneous cloud federation. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 110–119, 2016.

[40] L. Ferrucci, D. Mandrioli, A. Morzenti, and M. Rossi. A metric temporal logic for dealing with zero-time transitions. In *2012 19th International Symposium on Temporal Representation and Reasoning*, pages 81–88. IEEE, 2012.

[41] C. Mechalikh, H. Takta, and F. Moussa. Pureedgesim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments. In *2019 International Conference on High Performance Computing Simulation (HPCS)*, pages 700–707, 2019.