© 2021 Elsevier. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

A Two-stage Multi-Criteria Optimization Method for Service Placement in Decentralized Edge Micro-Clouds

Javier Panadero[a], Mennan Selimi[b], Laura Calvet[a,c], Joan Manuel Marquès[a], Felix Freitag[d]

[a] *IN3 - Computer Science Dept., Open University of Catalonia, Barcelona, Spain*
{jpanaderom, lcalvetl, jmarquesp}@uoc.edu
[b] *Max van der Stoel Institute, South East European University, North Macedonia*
m.selimi@seeu.edu.mk
[c] *Valencian International University, Valencia, Spain*
[d] *Department of Computer Architecture, Universitat Politècnica de Catalunya, Barcelona, Spain*
felix@ac.upc.edu

Abstract

Community networks are becoming increasingly popular due to the growing demand for network connectivity in both rural and urban areas. Community networks are owned and managed at the edge by volunteers. Their irregular topology, the heterogeneity of resources and their unreliable behavior claim for advanced optimization methods to place services in the network. In particular, an efficient service placement method is key for the performance of these systems. This work presents the Multi-Criteria Optimal Placement method, a novel and fast two-stage multi-objective method to place services in decentralized community network edge micro-clouds. A comprehensive set of computational experiments is carried out using real traces of *Guifi.net*, which is the largest production community network worldwide. According to the results, the proposed method outperforms both the random placement method used currently in *Guifi.net* and the Bandwidth-aware Service Placement method, which provides the best known solutions in the literature, by a mean gap in bandwidth gain of about 53% and 10%, respectively, while it also reduces the number of resources used.

Keywords: Service Placement, Distributed Systems, Community Networks, Micro-Clouds, Multi-objective Optimization Algorithms

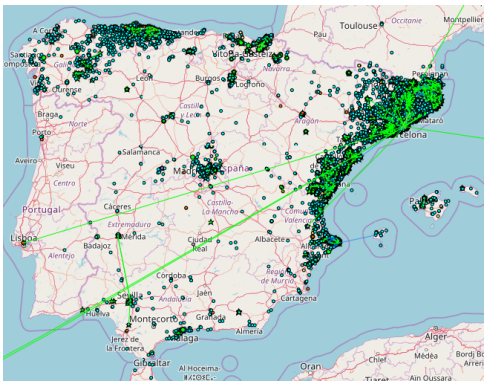
1. Introduction

Community networks (CNs) [1] are networks built and managed in a selflessly way by citizens and non-profit organizations. Public participants combine their efforts to share their resources in order to instantiate network infrastructures. These kinds of networks began to arise in the early 2000s and gained attention quickly due to the need of connectivity in rural areas. Today, CNs are widely extended around the world. One of them is *Guifi.net* [2] located in Spain, which is currently the largest and fast growing CN worldwide, with more than 35.000 operational nodes. Fig. 1 depicts the nodes and links of *Guifi.net* CN in Spain.

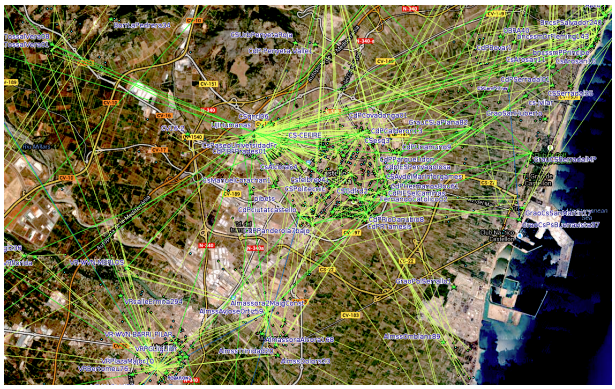
The consolidation of today's cloud technologies offer CNs the possibility to collectively build CN *micro-clouds* [3], building upon user-provided networks and extending towards an ecosystem of cloud services. In CN micro-clouds, services are hosted at edge nodes with communication, computation, and storage capabilities. The micro-clouds can include micro-servers, routers, IoT gateways, mobile devices, etc., and they are connected to the wireless mesh network (WMN) and provide low-latency service to users that are within a suitable communication distance. Examples for services in micro-clouds are distributed storage, monitoring service, live-video streaming or other IoT services.

The provisioning of performing services in micro-clouds is not a simple problem for several reasons. One important difficulty is the dynamics in the network: Given the fact that volunteers in CNs are free to connect and disconnect their resources whenever they want, CNs may suffer from a lack of reliability, which can cause significant delays or a loss of data. Further, given the characteristics of a communication over a wireless channel, unreliable network and user devices at non-optimal locations, the physical topology of the CN where the *micro-clouds* are deployed is in a constant state of flux. Another difficulty is the lack of centralized design. Specifically, the service deployment in *Guifi.net* micro-clouds is not centrally planned, but initiated individually by the CN members. As a consequence, public, user and community-oriented services are placed randomly on super-nodes and users' premises, respectively. Intuitively a network- and service-aware placement by the service providers could help to the exploit specific characteristics of the topology to improve service performance.

The question arises whether the performance of services in micro-clouds within CNs can be improved by carefully choosing the micro-cloud nodes that are going to host a particular service. A second question is how to identify the set of nodes which allow providing better characteristics for a service to be deployed. Key factors to take into account when answering these questions are the bandwidth (*i.e.*, maximize the overall



(a) *Guifi.net* nodes in Spain



(b) *Guifi.net* nodes in Barcelona city

Figure 1: The map of *Guifi.net* nodes in Spain (left) and Barcelona city (right).

available bandwidth between nodes), availability of the nodes, resource usage (*e.g.*, CPU, memory), etc. Specific challenges are the heterogeneity of links and nodes, computational constraints in the micro-clouds, and the dynamics of the network and resources. Since cloud services have started moving to the edge of the network, optimization approaches for these environments will become relevant for important use cases. Many of the current research works, however, have focused on the more homogeneous data center clouds infrastructures.

In this context, as a first contribution we propose a two-stage multi-objective optimization method, named *Multi-Criteria Optimal Placement (MCOP)*, which is a novel and fast method to place services in large-scale CNs. The MCOP method aims to maximize the overall bandwidth among the service instances. At the same time, it carries out an awareness use of the resources, minimizing the number of replicas to place. The first stage of the method relies on a Lexicographic Ordering (LO) multicriteria optimization strategy [4]. This stage consists of a hierarchical method in which the intrinsic properties of both the nodes and the network (*e.g.*, bandwidth, number of hops, and node quality) are categorized according to different priority levels. Then, a sequence of decisions is made following the previously established priority order. This procedure aims to efficiently select a set of nodes (candidate nodes) where a service has to be replicated to maximize the overall Quality of Service (QoS) – in terms of available bandwidth – of the system, which reverts in a higher Quality of Experience (QoE) for the users of the network. Once the candidate nodes have been selected, the second phase is applied. In this phase a multi-objective heuristic, named $Sort\&Merge$, is executed to reduce the number of replicas used to place the service, while considering a minimum loss of QoS. By doing this, the resources' load is alleviated. The MCOP method provides high-quality solutions in a very fast way, since it does not require costly computations in run-time, ensuring a minimum QoS to the users. Moreover, due to its flexibility to prioritize the properties of the system (nodes and network), this method may be also useful in other kinds of networks, such as in data centers (DCs) to place virtual machines (VM) or fog computing environments.

In order to diversify the search of high-quality service placement solutions a second contribution is proposed. The *Biased-Randomized Multi-Criteria Optimal Placement (BR-MCOP)* method is an extension of the MCOP method integrating biased randomization techniques [5]. It refers to the use of skewed probability distributions to induce an oriented (biased) random behaviour of the MCOP method, transforming the deterministic method into a probabilistic one, preserving the logic behind the multi-objective method. This way, the available bandwidth among services is increased at the expense of investing more computational time.

An experimental validation of the proposed service placement algorithms is carried out with real traces of *Guifi.net* CN [6]. According to the results, the proposed methods outperform both the random placement, which is used by default at *Guifi.net*, and the Bandwidth-aware Service Placement algorithm (BASP) placement method [3], which provides the best known solutions of the literature, obtaining an average gap for all the scenarios tested of about 53% and 10%, respectively in the MCOP method, and 58% and 20% in the BR-MCOP method.

The remaining of the paper is structured as follows. Section 2 presents the current state of service deployment in community networks through a study of *Guifi.net* as well as related research on service placement in data centers, distributed clouds and in wireless environment. Section 3 introduces a formal description of the service placement problem. In Section 4 the proposed optimization methodologies (MCOP and BR-MCOP) are explained. The data used, the computational experiments, and the results are described in Section 5. Finally, Section 6 draws the most relevant conclusions of this work and identifies lines of future research.

2. Background and Related Work

2.1. Community networks background

CNs [7] refer to decentralized, open, neutral, and self-organized communication networks built and operated by citizens for citizens. They are based on the principle of reciprocal sharing of network bandwidth. Their purpose is to both satisfy community's demand for Internet access and provide services of local

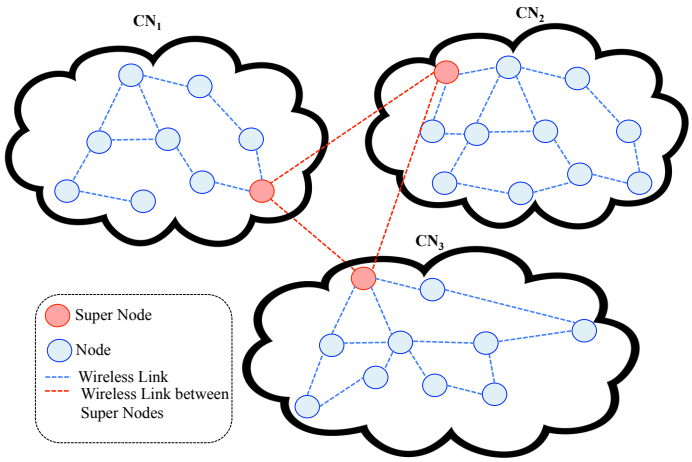


Figure 2: Overview of *Guifi.net* topology.

interest [8]. The main characteristics of CNs are that participants contribute with their resources to help establish the network, which is managed as a common resource [9]. Because resources are contributed in a voluntary manner, the nodes that compose the CN (*i.e.*, routers, laptops, computer machines, etc.) vary widely in terms of capacity, function and capability. Nowadays, with the advent of the telecommunications technology, most CNs are based on Wi-Fi technology, such as ad-hoc networks and IEEE 802.11a/b/g/n access points.

Guifi.net: A real example of a CN is *Guifi.net*, which is described as open, free and neutral. It started in 2004 in Catalonia (Spain) and today it has more than 35,000 operational nodes, which makes it the largest CN worldwide. This network offers a wide range of services, provided by individuals, social groups, small nonprofit or commercial service providers [8]. *Guifi.net* consists of a set of nodes interconnected through mostly wireless equipment that participants voluntary install and maintain, as well as its links. Since *Guifi.net* relies on donated resources, its infrastructure is heterogeneous and highly unreliable, compared to DCs, where they tend to be very homogeneous. This strong heterogeneity is due to the diverse capacity of nodes and links, as well as to the asymmetric quality of wireless links. Moreover, diverse technologies are employed in the network, ranging from very low-cost, off-the-shelf wireless (Wi-Fi) routers, home gateways, and laptops, to expensive optical fiber equipment.

Topology: The network topology of a wireless CN such as *Guifi.net* is different with respect to conventional Internet Service Provider (ISP) networks [10]. Indeed, as shown in Fig. 2, *Guifi.net* is composed of numerous distributed CNs and they represent different types of network topologies. Unlike in a DC environment, its overall topology is constantly changing (mostly mesh topology in urban areas and star topology in rural areas). To connect different CNs among them, all the CNs have a node called Super Node (SN) with multiple wireless links, which connect with the SNs of the other CNs to form the backbone of *Guifi.net*. A node can be a SN only if it meets a set of premises imposed by the board committee. This is done with the goal of maintaining stable and permanent connectivity among CNs.

Non-uniform resource distribution: Another feature adding complexity to this network is that its resources are not uniformly distributed. Indeed, the distribution is affected by geographic singularities. The set of nodes and links are organized under a set of mutually exclusive and abstract structures called administrative zones, which represent the geographic areas where nodes are deployed. A zone can represent nodes from a neighborhood or a city. Each zone can be further divided into child zones that cover smaller geographical areas where nodes are close to each other.

All these intrinsic network properties result in a highly complex and dynamic system. As a consequence, a service placement method that works well in a certain network might have a poor performance in another one. Or even worse, a service placement method working well in a given instance of time, may not fit properly in a different instance of time. For these reasons, it is important to propose placement methods able to consider the network variations and adapt to them.

2.1.1. Motivating use case: Placement of storage servers

The recently developed distributed monitoring system for *Guifi.net* consists of storage servers and monitoring clients [11]. The storage servers are replicated in order to dispose of monitoring data in case of network partitions or server failures.

For the deployment of the storage servers (*i.e.*, instances of the distributed database), computing nodes available in *Guifi.net* need to be chosen. This service placement challenge can be stated as: *given a large set of available computing nodes to become storage servers, on which subset of the nodes the storage servers should be placed?*

If nodes with low-capacity connectivity or hosting already many applications are chosen for the storage server placement, then the data storage at these nodes might suffer from failures and link saturation. In addition, end user experience will be decreased by consuming too much of the nodes' remaining computational resources and link's bandwidth. To avoid this, the service placement algorithm should select nodes with sufficient link bandwidth and remaining computing capacity to host the storage servers of the distributed monitoring system.

2.1.2. Baseline: Bandwidth-aware Service Placement algorithm

Our baseline for the storage server placement is the BASP algorithm [3], which is an algorithm used for placing services in CNs. It allocates services taking into account the bandwidth of the network and the node availability. The BASP is executed every single time a (new) service deployment is about to be made. In every run, the partitions the network topology into k (maximum allowed number of service replicas) and removes the nodes that are under the pre-defined availability threshold (phase 1); estimates and computes the bandwidth of the nodes (phase 2); and finally re-assigns nodes to selected clusters (phase 3). Fig. 3 demonstrates the phases of the BASP.

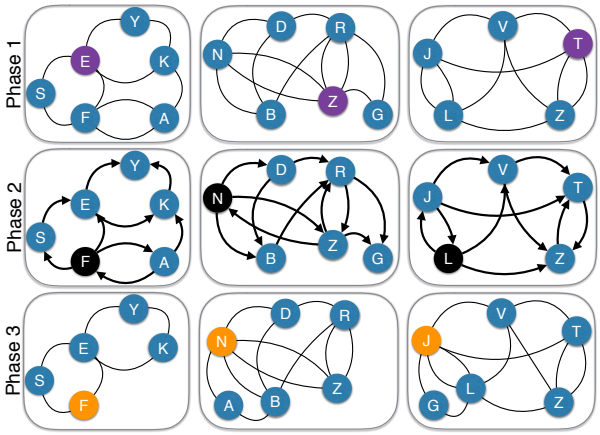


Figure 3: Phases of the BASP algorithm.

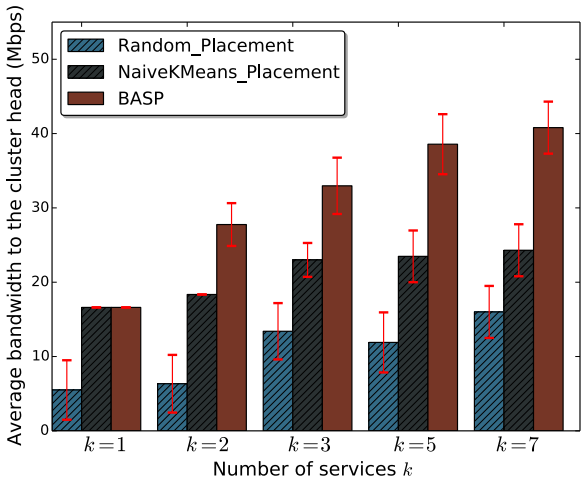


Figure 4: Average bandwidth to the cluster heads.

2.1.3. Bandwidth-aware Service Placement algorithm: Results

Figure 4 depicts the performance of BASP when comparing with other placement strategies (random and K-Means) used in CN micro-clouds. Fig. 4 reveals that for the considered number of services k, BASP outperforms both *Naive K-Means* and *Random* placement. For $k = 2$, the average bandwidth to the cluster heads has increased from 18.3 Mbps (Naive K-Means) to 27.7 Mbps (BASP), which represents a 50% improvement. The highest increase of 67% is achieved when $k = 7$. On average, when having up to 7 services in the network, the gain of BASP over Naive K-Means is of 45%. As shown in Fig. 4, the gap between the two algorithms grows as k increases. It can be concluded that k will increase as the network grows. And hence, BASP will presumably render better results for larger networks than the rest of strategies.

2.2. Related Work

The increasing importance of making an efficient use of resources in large-scale networks composed of heterogeneous devices has boosted the number of works focusing on service placement. This section reviews the main related works classified by application: placement in data centers, placement in distributed clouds, and placement in wireless environment. After describing the works, each subsection discusses the differences between these works and ours.

2.2.1. Placement in data centers

Previous publications (*e.g.*, [12, 13]) claim that the data center network is a limiting factor in the performance of many applications. In data centers hosted in cloud computing infrastructures, service placement plays an important role in the efficient use of the network. LaCurts *et al.* [14] present *Choreo*, a measurement-based method for placing applications in cloud infrastructures, which minimizes the application execution time. *Choreo* makes fast measurements in the cloud network employing packet trains and applies machine learning methods to profile the application network demands. Then, this information is sent to a greedy heuristic that places the applications. Similarly, Duan *et al.* [15] present four different placement schemes to place virtual networks in fat-tree data center networks, attempting to map virtual networks to a physical infrastructure in an efficient way. The authors put forward a model of multicast-capable virtual networks (MVNs) to meet the requirements of instant parallel data transfer between multiple computing units.

Other works have focused on placement decisions based on the evolution of the systems. Agarwal *et al.* [16] present *Volley*, an automated data placement tool for geo-distributed data centers of Microsoft. *Volley* analyzes the system logs by means of an iterative optimization algorithm based on data access patterns and client locations. The objective is to send migration recommendations back to the cloud service. Ghanbari *et al.* [17] develop a simple and fast optimization algorithm for optimal service placement of a set of N-tier software systems subject to changes in the workload, service level agreements (SLA), and administrators preferences.

Due to the high concern regarding environmental issues in data centers, green computing is becoming increasingly popular. In this context, Wu *et al.* [18] present two placement algorithms based on multi-objective optimization [4] and the enhanced Tabu Search [19] to address the trade-off between brown energy (coal, oil, natural gas, etc.) consumption and cost in cloud networks. The goal is to jointly minimize the data center cost and brown energy consumption in a VM migration-enabled cloud network.

All these approaches assume in their placement models that both the nodes and the links of the network are reliable resources (*i.e.*, they are always available). Thus, their only concern is to maximize the overall performance of the system. Different from these works, the methods proposed in this work consider in the service placement the unreliability of the resources in the network, *i.e.*, of nodes and links, to ensure a minimum QoS to the users.

2.2.2. Placement in decentralized clouds

Efficient resource placements are also key in distributed cloud systems, which are geographically distributed over a large number of locations.

MAPLE [20] is a Machine Learning approach for Efficient Placement and Adjustment of Virtual Network Functions (VNFs).

MAPLE partitions the substrate network into a set of on-demand clusters managed by a set of cluster-heads. The k-medoids clustering technique is employed which is praised for its robustness to noise and outliers. Based on the authors, MAPLE provides an original data-driven methodology to reduce the complexity of NP-hard optimization problems using machine learning and it scales well in large data centres. MAPLE however addressed a large topology of 500 nodes and 5260 links with bandwidth from 100 Mbps to 2000 Mbps, which is different in magnitude to that of *Guifi.net* where there are around 70 nodes, bandwidth ranges from 5 Mbps to 100 Mbps, and the nodes are resource constraint low-power devices.

The authors of [21] propose a trust-aware scheduling solution called *BigTrustScheduling*. The proposed approach derives a trust value for each VM based on its underlying performance and then intelligently maps the tasks to the appropriate VMs (i.e., task placement) in such a way that the makespan and the cost of tasks processing is minimized. The authors claim that their solution significantly reduces the run time compared to the studied solutions. Another work [22] proposes the Deep learning Smart Scheduling (DSS), an automated big data task scheduling approach for cloud computing environments. DSS combines Deep Reinforcement Learning (DRL) and Long Short-Term Memory (LSTM) to automatically predict the Virtual Machines (VMs) to which each incoming big data task should be scheduled.

Pasteris *et al*. [23] propose an approximation algorithm for solving the service placement problem in systems with heterogeneous service/node sizes and rewards. Their algorithm has a constant approximation ratio. Similarly, Farhadi *et al*. [24] propose a two-time-scale solution for joint service placement and request scheduling in edge clouds under communication, computation, and storage constraints. They prove the NP-hardness of the service placement problem in the general case and characterize its complexity in all special cases. Steiner *et al*. [25] develop efficient service placement algorithms to maximize the performance of distributed cloud systems. These algorithms require as inputs the status of the network, and the compute and data resources, which are matched to application requirements. Zhang *et al*. [26] present two algorithms for the distributed cloud selection and VM partition problems. These algorithms are based on a novel clustering method, which minimizes the maximum inter-distance between clouds considering the distance between clouds and network graph information such as topology and density. Alicherry *et al*. [27] put forward an efficient 2-approximation algorithm for the optimal selection of data centers in distributed clouds. Considering as input the resources needed for the computational task, the algorithm selects the data centers by minimizing the distance or latency between them. Then, it selects within each data center the most suitable racks and servers where the requested VM for the task will be located.

Due to the huge popularity of social networks, there are several works on the assignment of resources in online social networks (OSN) that run over large-scale distributed clouds. Jiao *et al*. [28] study the problem of optimizing the monetary cost spent on cloud resources when deploying an OSN service over multiple geo-distributed clouds. They propose *Cosplay*, an algorithm that minimizes the total cost while ensuring the QoS and data availability. Similarly, Xia *et al*. [29] analyze the problem of user data placements of OSN in a distributed cloud minimizing the operational cost of a cloud service provider. This work proposes a fast yet scalable algorithm that builds on the use of the community concept by grouping users of a social network into different communities and placing the master replicas of user data in the same community into a data center. In addition, the algorithm replicates the slave replicas of the user data in nearby data centers. Khalajzadeh *et al*. [30] present a novel genetic algorithm-based approach to optimize social media data placement and replication in distributed clouds. The algorithm attempts to find a near-optimal number of replicas for every user's data and a near-optimal placement of replicas to minimize monetary cost while satisfying latency requirements for all users.

The methods proposed in these works focus on the selection of DCs or VMs in distributed clouds to place services or computing applications, both to maximize the benefits of the system and reduce the cost for the users. These methods need to know previously the requirements about the application or the users. The methods proposed in this work differ from the reviewed ones because they are agnostic. Each node of the network could be a potential candidate to host services and the selection is performed without any kind of information about users and application requirements.

2.2.3. Placement in wireless environment

Wireless networks have been gaining popularity during the last few decades, which has been accompanied by an increment of the number of works on this field. Selimi *et al*. [3] put forward a service placement algorithm called BASP, to place micro-cloud services in CNs. The algorithm uses K-Means for clustering and a lightweight bandwidth computation/estimation heuristic. The authors evaluate its performance over the CN *Guifi.net*. Another study about placement in CN is proposed in Vega *et al*. [31], which optimizes the communication cost in CN by minimizing the service overlay diameter and the coordination cost along the network. Coimbra *et al*. [32] propose a novel service placement approach based on community finding using a scalable graph label propagation technique and decentralized election procedure. The authors are using traces from the *Guifi.net* CN.

Al Arnaut *et al*. [33] propose a content replication scheme for wireless mesh networks. This scheme is divided into two phases including the selection of replica nodes (network setup phase) and content placement, where content is cached in the replicas based on popularity.

There are some works that consider a different type of wireless networks. For example, Herrmann *et al*. [34] propose a fully decentralized, dynamic, and adaptive service placement algorithm for ambient intelligence (AmI) [35] environments like the ad-hoc service grid (ASG) infrastructures [36]. This algorithm achieves a coordinated global placement pattern that minimizes the communication costs without any central controller. It does not even require additional communication among the

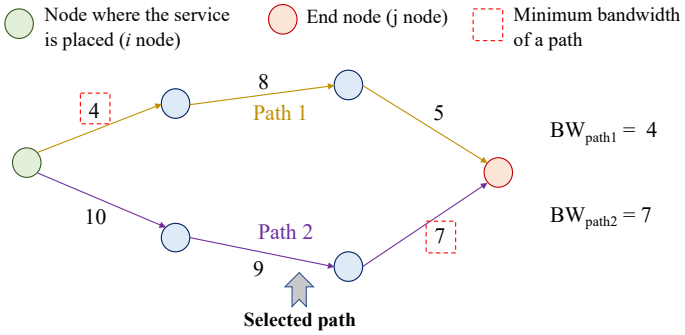


Figure 5: Path selection based on bandwidths.

replicas. Tärneberg *et al*. [37] present an algorithm that holistically returns the globally optimal placement of static and mobile applications in mobile clouds networks (MCN) [38]. The goal of the algorithm is to minimize the global system cost as a means to manage the compute and network resources in a MCN.

Our work differs from these others because the proposed methods enable the prioritization, in an easy and fast way, of the most important parameters of both the network and the nodes to place services, providing flexible and agile methods. Moreover, this flexibility makes it possible to apply them to a wide range of large-scale distributed systems, not just CNs.

3. Service Placement in Decentralized Edge Micro-Clouds

A CN needs to use the available bandwidth resources to reach its full potential. Based on this idea, the objective of our problem is to find a deployment that maximizes the overall available bandwidth. A CN may be modeled as an incomplete directed graph $G = (N, E)$, where N is the set of nodes that compose the CN and E is the set of links (wireless or optic-fiber) that connect pairs of nodes. Nodes in N without links to other nodes (*i.e.*, isolate nodes) are discarded. The links are characterized by a given bandwidth B_{ij}, $\forall(i, j) \in E$, and latency L_{ij}, $\forall(i, j) \in E$, while each node has a particular QoS_i, $\forall i \in N$, derived from the real measurements in the CN. Given a micro-cloud to be deployed, at most $R_{\max}$ replicas can be placed. A micro-cloud can be deployed in a node only if this node has a QoS greater than a minimum threshold (QoS_{min}). Similarly, a given link will be used only if its bandwidth is higher or equal to a given threshold (B_e). Each node not deploying any micro-cloud has to be assigned to at least one node deploying one.

The path from node i to node j is defined as the path with minimum length among all the possible paths. If two paths have the same length, the path with maximum bandwidth is selected. The bandwidth of a given path is described as the minimum bandwidth among its links (see Fig. 5). The bandwidth of a given micro-cloud is the mean bandwidth of the paths associated. The objective function maximizes the mean value of the micro-clouds' bandwidth.

A solution is described by three types of binary variables: (*i*) x_i identifies the nodes where the replicas are placed; (*ii*) y_{ij} reveals whether node i is assigned to node j (note that if y_{ij} is equal to 1, then x_j has to be equal to 1); and (*iii*) z_{ijkl} shows the path from the nodes where i is assigned to itself, thus z_{ijkl} is equal to 1 if arc (k, l) is traversed when serving node i from micro-cloud j. All the notation required is gathered in Table 1.

Table 1: Notation of the optimization problem

Sets	**Description**
G	Graph
N	Set of nodes
E	Set of links
Parameters	**Description**
R_{max}	Maximum number of replicas
B_{ij}	Bandwidth requirement associated with link (i, j)
B_e	Minimum bandwidth
L_{ij}	Latency associated with link (i, j)
P_{ij}	Minimum number of hops from j to i
QoS_i	Quality of node i
QoS_{min}	Minimum quality
Variables	**Description**
x_i	Micro-cloud deployed in node i
y_{ij}	Node i is assigned to node j
z_{ijkl}	Link (k, l) is part of the path from micro-cloud j to i

4. Two-stage Multi-criteria Optimization

4.1. Model for Estimating Nodes' Quality

An important parameter to place services in a CN is the QoS of a node. Basically, a node with a high QoS is expected to be a good candidate to place a service. In this section, a model to estimate the quality of a node is estimated. It relies on three variables: probability of being connected during a given amount of time (t_i), bandwidth of its links (b_i), and CPU load (c_i). This model will be used in the MCOP method to select the most suitable nodes to place micro-clouds.

The parameter t_i – described in the next subsection– depends on the period in which the node was connected in an uninterrupted way in the past, while b_i is the minimum bandwidth of its links. c_i is also considered in the model in order to avoid placing micro-clouds in overloaded nodes, which would increase the time of response. In particular, the QoS of node i is computed as:

$$QoS_i = \frac{1}{3}(t_i + b_i - c_i), \quad (1)$$

The three inputs and the QoS are normalized to 0-1 ranges. Each variable has the same weight considering that they have the same relevance. Since the network is highly dynamic, the nodes' QoS need to be recalculated regularly.

4.2. Estimating a node's probability of being active

Being able to correctly predict whether a node will be connected in the next future is both challenging and essential. If this information is not available or is not accurate, the optimization method will lack important information and may provide solutions with a poor performance.

There are several machine learning techniques in the literature [39] that are widely used to predict behaviors of real expert systems. Among them, the logistic regression is a probabilistic technique, highly accurate, simple, and fast. The model proposed is defined as:

$$z_i = \beta_0 + \beta_1 \cdot node_i + \beta_2 \cdot week_i + \beta_3 \cdot day_i + \beta_4 \cdot timeSlot_i, \quad (2)$$

where:

- $node_i$, $week_i$, and day_i, identify node id, week, and day of the month for the observation i;
- $timeSlot_i$ refers to a time slot and may take the values 1 (from 12 noon to 8 am), 2 (from 8 am to 4 pm), and 3 (4 pm to 12 noon);
- $\beta_0, \beta_1, \beta_2, \beta_3$, and β_4, are parameters to estimate; and
- z_i represents a linear function of the explanatory variables (*i.e.*, $node_i$, $week_i$, day_i and $timeSlot_i$).

Then, the logistic function can be written as:

$$t_i = \frac{1}{1 + e^{-z_i}}, \quad (3)$$

where:

- e denotes the exponential function; and
- t_i is interpreted as the probability of $node_i$ being connected during $week_i$, day_i and $timeSlot_i$.

This model is designed considering data covering a few weeks or months. If we gathered data for years, the variable $year_i$ could be added to explore whether there is a significant trend.

The historical data is split in two: the observations of the first 3 weeks constitute the training set (*i.e.*, observations used to build the model), and the remaining observations represent the test set (*i.e.*, observations employed to assess the model by computing performance measures). If the data is not grouped by the time slots defined, then a transformation is required. For instance, if an observation refers to an hour, then 8 observations constitute a time slot: only if in 6 or more observations the node has being connected, we will say that the node has being connected during that time slot.

The model estimated can be used to compute t_i for new observations. If $t_i > 0.5$, then the model predicts that $node_i$ will be activated during $timeSlot_i$ of day_i and $week_i$. Otherwise, the model will predict there will be no activity.

The classical performance measures considered are:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (4)$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (5)$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

where:

- true positives is the number of observations where the node is activated and the model predicts 'activated' too;
- false positives is the number of observations where the node isn't activated but the model predicts 'activated'; and
- false negatives is the number of observations where the node is activated but the model predicts 'not activated'.

Finally, it is important to consider a classical problem in prediction problems that can be encountered: unbalanced classes. It happens when the training set does not have the same number of observations per category. In our case, it could happen that most observations indicated 'activated' or 'not activated'. It is a problem because a category is miss-represented and the resulting model may have a poor performance when trying to identify new observations belonging to that category. A simple option is to sample the observations in the training set so that there is the same number of observations of both categories. Thus, if there are n_0 observations of the category 'not activated' and n_1 'activated' and $n_0 < n_1$, then the options are: (1) randomly select only n_0 observations where the node is activated; or (2) randomly select n_1 observations where the node is not activated (thus, having repeated observations).

4.3. Optimization Methodology

This section describes the fundamentals of the MCOP method, which is an efficient and fast two-stage multi-objective optimization method to select the most suitable nodes to place services in large-scale CNs. The main goal of the MCOP method is to maximize the overall available bandwidth among the micro-cloud nodes, at the same time it attempts to minimize the number of replicas of a micro-cloud node instance to avoid inefficient resource usage.

In order to improve the performance of the MCOP method, we also introduce the BR-MCOP method, which is an extension integrating biased randomization techniques [5]. These techniques refer to the use of skewed probability distributions to induce an oriented (biased) random behaviour of the MCOP method. In other words, the deterministic method is transformed into a probabilistic one, while preserving the logic behind it.

Before introducing the proposed methods, the Lexicographic Ordering (LO) approach [4] is briefly introduced. It is highly used in multi-objective combinatorial optimization and key in

the proposed methodology. After that, the two stages of the MCOP are explained. Finally, the extension proposed is described.

4.4. Lexicographic Ordering approach

The LO approach avoids having to add up quantitative parameters using weight factors, which may have some disadvantages due to: (*a*) the numerical quantities are typically not based on a uniform scale; (*b*) the number of parameters can be large; and (*c*) the consequences of a given trade-off cannot be quantitatively known prior to the optimization. Instead of using weight factors, in the LO approach the priorities of the individual planning criteria (*i.e.*, the objective functions) are explicitly incorporated in the optimization process. In particular, the objective functions are ranked in terms of importance being minimized (or maximized) sequentially.

This multi-objective approach can be represented as an objective function $F(x) = [f_1(x), f_2(x), ..., f_N(x)]$, which gathers a collection of N individualized functions ($f_i(x)$) ordered by importance, being $f_1(x)$ the most important and $f_N(x)$ the least. The value found for each objective is transformed into a constraint for subsequent optimizations. By doing this, the optimal value of the most important objectives is preserved. Mathematically, this method can be modeled as an ordered sequence of real objective functions with a set of constraints as follows:

$$\text{Min/Max} \quad f_i(x) \tag{7}$$

subject to:

$$\begin{array}{ll} f_j(x) \leq f_j^* & \text{if } j \text{ is minimized} \\ f_j(x) \geq f_j^* & \text{if } j \text{ is maximized} \end{array} \tag{8}$$

where $i = \{1, 2, ..., N\}$ and $j = \{1, 2, ..., i-1\}$.

As the method progresses down from level 1 to level N, the preceding objective functions are converted to new constraints with boundary values f_j^*. Accordingly, the number of constraints increases with each level, reducing the feasible search space gradually.

4.5. First stage of the Multi-Criteria Optimal Placement method: initial solution

The first stage of the method (Fig. 6) relies on a hierarchical placement method to generate an initial solution. The inputs are the complete list of active nodes, a set of criteria parameters, which are ranked by importance, and a list of constraints. The set of criteria parameters are represented as a vector of tuples:

$$\begin{array}{l} C = [\{1, threshold_1, \%elite_1, type_1\}, ..., \\ \{N, threshold_N, \%elite_N, type_N\}] \end{array} \tag{9}$$

Each criterion (represented by a tuple) has associated four parameters: (*a*) its importance (from 1 to N, being 1 the most important); (*b*) a threshold value (*i.e.*, the minimum or maximum value that can be accepted to meet the requirement); (*c*) the maximum percentage of nodes to be selected to obtain the next list of high quality nodes or 'elite nodes'; and (*d*) whether the corresponding function has to be maximized or minimized.

Initially, the list of active nodes is filtered by the criteria parameters. Thus, the nodes that do not meet the minimum requirements imposed by the thresholds are removed. It makes the sorting procedure more efficient, since those nodes without possibility of being selected are discarded at the beginning.

The filtering returns a non-sorted list of viable nodes. Afterwards, an iterative procedure that takes into account the priority of the criteria is applied. First, as shown in Fig. 6, the list is sorted by the first criterion (the most important one): in an ascendant order if the corresponding value has to be maximized or in descendent order otherwise. Then, a percentage $\%elite_i$ of nodes is selected. After the first iteration, a list of elite nodes taking into account the first criterion is obtained. Then, this procedure continues reducing the size of the list by considering a different criterion at each iteration. By default, we set $\%elite_i$, $\forall i \in \{1, ..., N\}$, to 10%. The loop stops when all the criteria have been considered. From the returned sublist of best ranked nodes, the final subset of candidate nodes is selected taking into account a final list of restrictions.

With the objective of applying the MCOP method for CNs, the following system parameters are employed (in decreasing order of importance):

- Node quality: is predicted by the model described in section 4.1 and has to be maximized.
- Connections: indicates the number of adjacent nodes for a given node and has to be maximized.
- Latency: indicates the largest latency among all the links starting at a given node and is expressed in Mbps and has to be minimized.

Moreover, two constraints are considered: (1) the number of replicas cannot exceed a user parameter, and (2) there is a minimum number of hops between candidate nodes (which is an input parameter). The second restriction aims to select candidate nodes distributed along the network, thus minimizing the distance between the candidate nodes and their assigned nodes.

Having selected the nodes in which services will be placed (so called candidate nodes), the other nodes have to be assigned so that they have access to a service deployed. This assignation relies on the Breadth First Search (BFS) algorithm [40], which is a well-known algorithm for traversing graphs used in graph theory. It starts with a root node and explores all the neighbour nodes at the current depth before moving to the nodes at the next depth level. Sub-graphs are generated which contain a candidate node as a root node and the nodes with a depth lower or equal to the maximum number of hops allowed. Notice that, since the BFS algorithm relies only on distance, a node can be served by more than one candidate node. The average bandwidth of each sub-graph is computed to obtain the system's bandwidth. It is important to highlight that more than one path could be found between a candidate and a final node. To deal with this issue, the methodology relies on the Floyd Warshall algorithm [41], which returns the minimum path between each

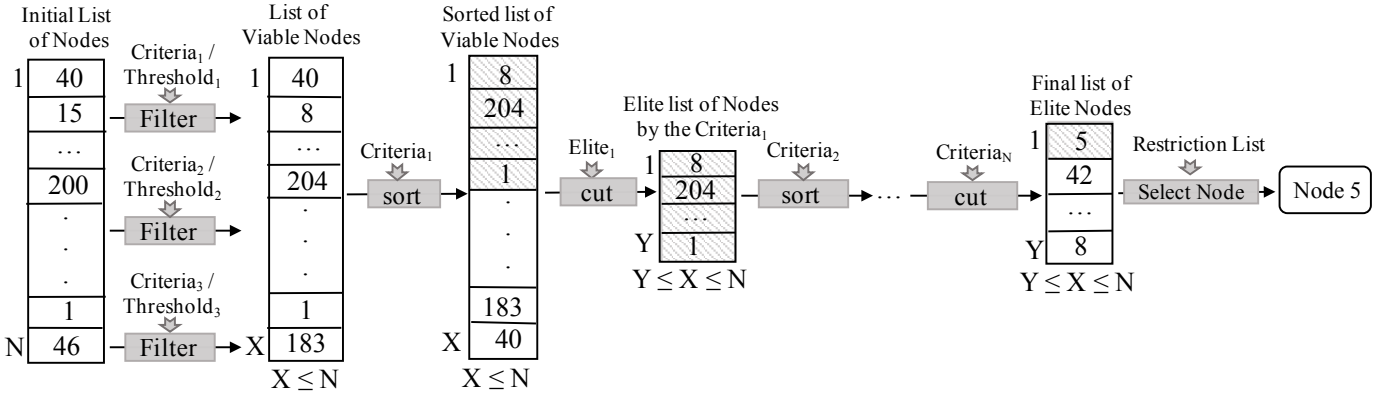


Figure 6: Procedure to generate an initial solution.

pair of nodes. This algorithm is run once at the beginning to create a distance matrix with the number of hops. In the case our algorithm finds more than one path with a minimum distance, it selects the one maximizing the bandwidth.

4.6. Second stage of the Multi-Criteria Optimal Placement method: Reducing the number of replicas

The second stage is a novel and fast multi-objective heuristic named *Sort&Merge* which, given a solution, reduces the number of replicas used while considering a minimum loss of QoS. Thus, this stage aims to reduce the replication of redundant services in the network, helping to alleviate the resources' load.

The main idea behind this heuristic is to merge sub-graphs with a low number of nodes, generating larger sub-graphs and thus avoiding the use of resources to serve few nodes. The heuristic tries to merge sub-graphs with shared nodes (*i.e.*, nodes served by more than one candidate node). Fig. 7(a) shows a solution returned by the first stage. In this example the solution has 3 sub-graphs, and all of them share the node 8. An iterated procedure is applied to merge the sub-graphs, subject to an acceptance criterion. The result is a new sub-graph with the shared node as a candidate node. According to Fig. 7(b), the sub-graph 2 – the smallest– is merged with the sub-graph 1. Then, in a new iteration, the heuristic tries to merge this new sub-graph with the sub-graph 3. The details of the heuristic are shown in Algorithm 1 and described below.

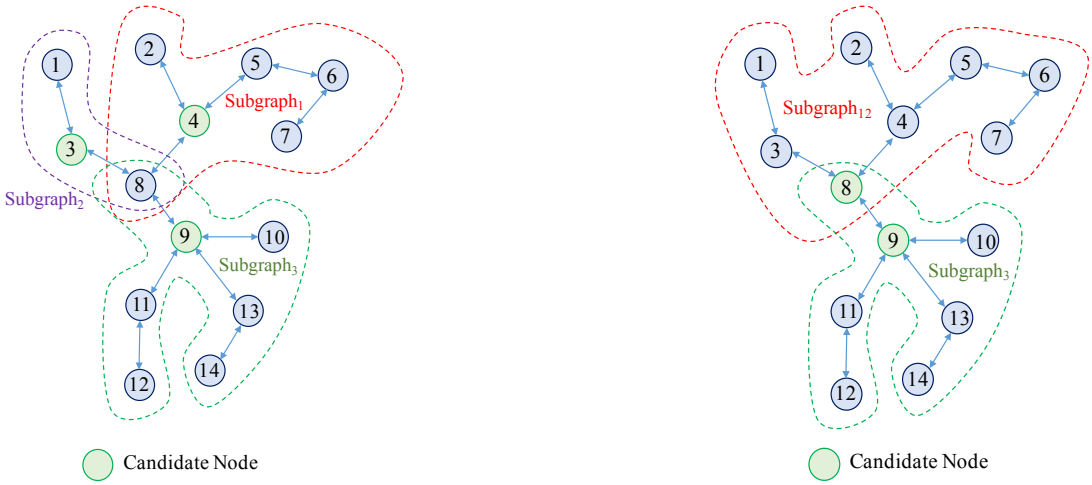
The inputs are the set of candidate nodes, the graph, and an initial solution. First, all the shared nodes are identified and stored in the list *sharedNodes* (line 2). Then, the heuristic starts an iterative procedure (lines 3-31), where each iteration considers a shared node. Initially, the heuristic searches all the sub-graphs containing that node and stores them in the list *subGraphList* (line 4). Afterwards, another iterative procedure (lines 6-30) starts. The first step is sorting the sub-graphs from that with the smallest number of nodes to the one with the biggest number of them. Then, it considers merging the sub-graph with the lowest number of nodes (*baseSubGraph*) with each of the other sub-graphs, establishing the shared node as candidate node for the resulting new sub-graph (*newSubGraph*). For each potential merging, the heuristic checks if the resulting sub-graph is promising (line 19). In particular, a sub-graph is labeled as promising if its bandwidth is at most a 1% lower than the average bandwidth of the two sub-graphs considered. If a merge is promising, the heuristic assesses whether the bandwidth of *newSubGraph* improves the bandwidth of the previous merges (line 20). Once all the potential merges have been analyzed, the heuristic merges *baseSubGraph* with the sub-graph with the highest bandwidth. After performing the merge, the original sub-graphs are deleted from *subGraphList* while the new sub-graph is added. The current solution (*currentSol*) is accordingly updated. Finally, once all the nodes in *sharedNodes* have been considered, the total bandwidth of *currentSol* is computed and the solution is returned.

The parameter to choose whether a merge is accepted has been set to 1%, but this parameter can be configurable by the user depending on the level of granularity needed. If he/she prioritizes minimizing the number of replicas for a service, this parameter should be increased at the expense of reducing the overall QoS.

This method builds on the idea of making sub-graphs that can be seen as clusters. However, while many clustering methods require as a parameter the number of clusters (replicas) to be used, the MCOP method is able to adjust that number on the fly.

4.7. Extending the Multi-Criteria Optimal Placement with biased randomization

The MCOP method is a constructive approach that employs an iterative procedure to generate a solution in an efficient way. At each iteration, the most promising nodes are selected from a list of potential candidates that have been sorted according to a specific criterion. As a result, a reasonably good quality solution is generated. Notice, that this is a somewhat myopic behavior, since the method selects the next movement without considering how the current selection will affect subsequent decisions and may fall in local minima. Even worse, this property



(a) Initial solution

(b) Final solution

Figure 7: Illustration of the *Sort&Merge* heuristic.

Algorithm 1 *Sort&Merge* heuristic

```
1:  procedure SORT&MERGE (candidateNodes, graph, currentSol)
2:      sharedNodes ← selectSharedNodes(candidateNodes, graph)
3:      for each (node in sharedNodes) do
4:          subGraphList ← selectSubGraphs(node)
5:          isMerged ← True
6:          while (isMerged is True) do
7:              isMerged ← False
8:              subGraphList ← sortDecreasingOrder(subGraphList)
9:              bestBandwidth ← 0
10:             baseSubGraph ← getSubGraph(subGraphList, 1)
11:             for (i from 2 to size(subGraphList)) do
12:                 newSubGraph ← initializeEmpty()
13:                 newSubGraph ← setCandidateNote(node)
14:                 newSubGraph ← addNodes(baseSubGraph)
15:                 subGraph ← getSubGraph(subGraphList,i)
16:                 newSubGraph ← addNodes(subGraph)
17:                 newPartialBW ← calBW(newSubGraph)
18:                 currentPartialBW ← (calBW(baseSubGraph)+calBW(subGraph))/2
19:                 if (acceptMerge(newPartialBW, currentPartialBW)) then
20:                     if (newPartialBandwidth > bestBandwidth) then
21:                         isMerged ← True
22:                         bestBandwidth ← newPartialBW
23:                         subGraphToAdd ← newSubGraph
24:                         subGraphToDelete ← subGraph
25:             if (isMerged is True) then
26:                 subGraphList ← delete(baseSubGraph)
27:                 subGraphList ← delete(subGraphToDelete)
28:                 subGraphList ← add(subGraphToAdd)
29:                 currentSol ← delete(baseSubGraph)
30:                 currentSol ← delete(subGraphToDelete)
31:                 currentSol ← add(subGraphToAdd)
32:     calBw(currentSol)
33:     return currentSol
```

results in a deterministic procedure (*i.e.*, if the method is executed several times, the same solution is obtained).

In order to avoid falling in local minima and improve the quality solution of the MCOP method, the search has to be diversified. To do this, the BR-MCOP method is proposed, which is an extension of the MCOP method integrating biased randomization techniques. These techniques enable transforming the deterministic behavior of the MCOP method into a probabilistic one. In particular, they introduce a certain degree of randomness, while still preserving the main logic behind the method. Thus, the first stage of the MCOP method is extended by using biased randomization and encapsulating the whole method inside a multi-start algorithm [42]. Algorithm 2 describes the extended MCOP.

The procedure starts generating an initial solution (*initSol*) using the (deterministic) MCOP method. It is stored as the best solution found so far (*bestSol*). Then, the multi-start process relying on the biased randomized MCOP is initiated. At each iteration, it builds a new solution (*newSol*) selecting the elite nodes by applying a biased-randomized process. This means that the nodes on the top positions of the list will be selected, but not necessarily the best nodes, which diversifies the node selection. A geometric probability distribution, which has a single parameter $\beta(0 < \beta < 1)$, is used to induce this skewed behavior. Thus, different best elite nodes may be selected at each iteration of the multi-start process, while still preserving the behavior of the original heuristic. Once *newSol* has been generated, it replaces *bestSol* if its average bandwidth is higher. This iterative process is repeated until the maximum computational time is reached. Although this method provides higher performance – bandwidth – than the MCOP method, due to this iterative process more computational time is needed to carry out the deployment. Thus, the network administrator should decide which method to use depending on the deployment needs.

Algorithm 2 Multi-Start biased randomization MCOP (Extended MCOP)

```
1: procedure EXTENDED MCOP (graph, maxNReplicas, maxCompTime, β)
2:     initSol ← MCOP(graph, maxNReplicas)
3:     bestSol ← initSol
4:     compTime ← 0
5:     while (compTime < maxCompTime) do
6:         newSol ← MCOP(graph, maxNReplicas, β)
7:         if (calBW(newSol) > calBW(bestSol)) then
8:             bestSol ← newSol
9:         updateTime(compTime)
10:    return bestSol
```

5. Empirical Studies

5.1. Data description

To carry out the experiments based on a real-life CN, data of *GuifiSants*, which is a subset of *Guifi.net*, have been used. This CN is composed of around 80 nodes (with more than 300 active users) and there are two gateways (*i.e.*, proxies) that connect the network to the rest of *Guifi.net* and the Internet [43]. Typically, the users of *GuifiSants* have an outdoor router (OR) with a Wi-Fi interface on the roof, which is connected through Ethernet to an indoor access point (AP).

Measurements were obtained by connecting via SSH to each *GuifiSants* OR and running basic system commands available in the QMP distribution. This method has the advantage that no additional software needs to be installed on the nodes. In order to obtain real-traces of this CN, measurements over a one-month period were captured. The live monitoring page and data is publicly available in the Internet[1].

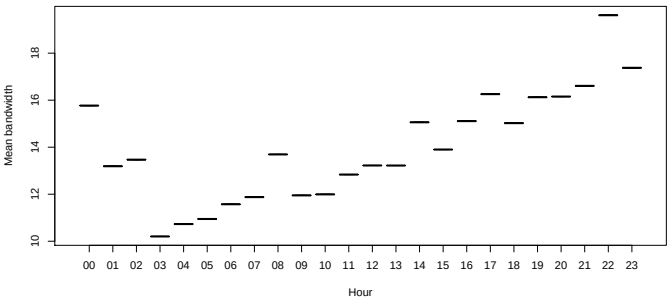


Figure 9: Mean bandwidth (in Mbps) for each hour of the day.

We have logs for 31 successive days (1 month) and 74 nodes. The monitoring system provides information for each hour of the day. There are only 6 non-consecutive hours where the process failed and there is no information. Fig. 8 displays several time series that describe the data. The first plot shows the number of nodes activated, which is relatively stable, ranging from

[1] `http://dsg.ac.upc.edu/qmpsu/`

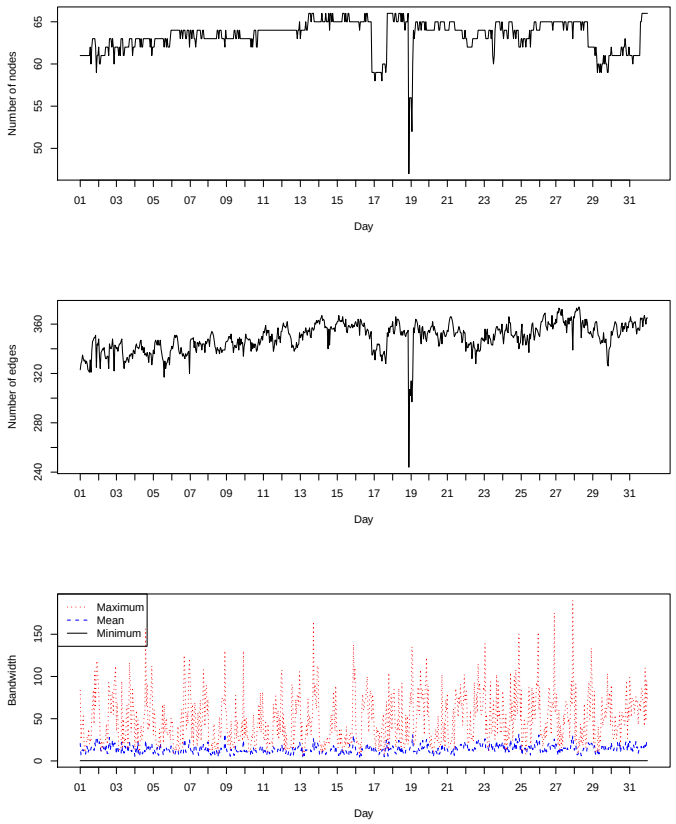


Figure 8: Time series of number of nodes (up), number of edges (middle), and maximum, mean, and minimum bandwidth in Mbps (down).

60 to 65 but with a dramatic drop in day 19th. The second plot shows the behavior of the number of edges, which varies between 320 and 360. There is also a drop in day 19th. A strong positive correlation between both measures is expected. Finally, the third plot displays the minimum, mean and maximum bandwidth (in Mbps) for each hour. It can be concluded that the maximum bandwidth is highly variable, ranging from 10 to 200 Mbps, approximately. The line of means is much closer to the line of minimums. This reveals that there were edges with extremely high levels of bandwidth. Fig. 9 averages the bandwidths for each hour of the day. It ranges from 10 to 20 Mbps, and there is a maximum at 10pm and a minimum at 3am.

5.2. Logistic regression

After transforming the data to have slots of time of 8 hours, the resulting dataset has 6,815 observations (each observation is a combination node-week-day-slot). 6,662 of these observations refer to nodes activated during the slot considered. Three first weeks are used to train the predictive model and the last one to test it. Among the 74 nodes, 66 are activated the three weeks, thus, they present no variability. Since they cannot provide information about changing states, these nodes are not considered for building the model. The predictions for these nodes will be

'activated' (probability = 1). The model built for the remaining 8 nodes is represented in Table 2. The first column displays the parameters in the model while the second column shows the estimated value for the coefficients (which can be positive and negative). Then, the third column shows the p-value obtained from testing the hypothesis "coefficient equals 0". Finally, the fourth column relates p-value to significance: if p-value < 0.001 (represented by '***'), one can state that the associated variable is statistically significant; if p-value is between 0.001 and 0.01 (represented by '**'), then the significance is lower than the previous case; and so on. Thus, the slot is statistically significant. In addition, there are significant differences among nodes. The performance measures are: precision = 0.878, recall = 1, and F_1 = 0.93. We attempted to transform the training set to create balanced classes, but no improvements in terms of F_1 were obtained. Thus, the original training set is used. The performance measures for the 'naive' model (all the predictions equal to 1) are: precision = 0.77, recall = 1, and F_1 = 0.87.

Table 2: Logistic model to predict whether a node will be activated.

Coefficients	Estimate	Pr(> |z|)	Significance
Intercept	2.02	0.00	***
TimeSlot	0.21	0.08	.
Day	0.05	0.29	
Week	-0.53	0.13	**
*Node*68	0.00	1	
*Node*69	0.00	1	
*Node*70	0.00	1	
*Node*71	-1.14	0.00	**
*Node*72	-0.31	0.47	
*Node*73	-1.19	0.00	**
*Node*74	-3.42	0.00	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

5.3. Computational experiments

This section describes the computational experiments carried out to assess our methods. They were implemented in Java 8 and the experimentation was executed in a standard personal computer. In particular, an Intel QuadCore i7 CPU at 2.2 GHz with 8 GB RAM and Ubuntu 18.04 as Operating System was employed to execute all the tests. The configuration parameters used are shown in Table 3. The maximum run time for the BR-MCOP method has been set to 100 seconds, which represents a good trade-off between the quality of the solution and execution time. The parameter of the geometric distribution is modeled as a uniform distribution ranging from 0.3 to 0.4. This range has been tuned based on some sensitivity analysis. The maximum number of replicas (*Rmax*) has been set to 10. This value has been selected based on the total number of nodes of the network in order to not overload the network. Finally, the minimum bandwidth (*Be*) and the minimum *QoS* are set to 5MB and 20MB, respectively. 3 weeks (21 days) were used to train the model to predict the quality of the nodes, and the last days of the month were used to validate the approaches. A day is composed of 24 traces (one trace per hour). To compute the results of a day, each trace is executed and the average of the bandwidth, time and number of replicas are calculated.

Table 3: List of parameters and values.

Algorithm's parameters	**Value or distribution**
maxTime (sec.)	100
β	U(0.3,0.4)
User's parameters	**Value**
R_{max}	10
B_e	5 Mbps
QoS_{min}	20 Mbps

5.4. Results

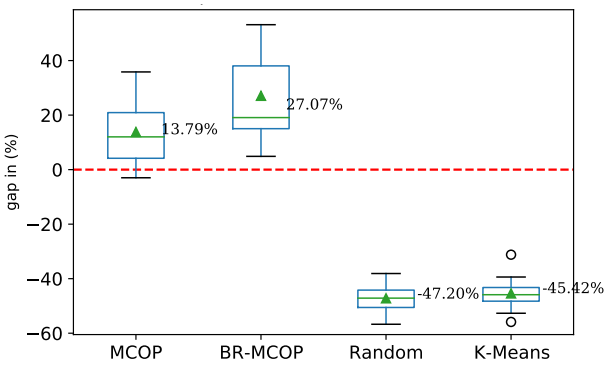
The results obtained with the proposed approaches are summarized in Table 4. The first column 1 indicates the tested day. Columns 2-4 show the average bandwidth of the system (in megabits per second), the computational time (in seconds) for the solution found with the MCOP method, and the number of replicas used. Columns 5-7 detail the same information for the BR-MCOP method. Notice that, although the algorithm is executed during 100 seconds, column 6 reveals the seconds required to find the best solution. Columns 8-10 describe the results obtained using a random approach, which is the current method applied by *Guifi.net* to place services. In order to reach QoS_{min} using this method, a node is randomly selected as candidate node to place a replica of a micro-cloud and then calculate the average bandwidth. If QoS_{min} was not reached, then a new candidate node was randomly selected and added, recalculating the average bandwidth of the system. These steps were repeated until reaching either QoS_{min} or the maximum number of replicas allowed. Columns 11-13 show the results obtained using a K-means approach, which is based on the locality of the nodes in the network to find the centroids (replicas) of the clusters. To define the best number of centroids, different experiments have been carried out varying the number of clusters, and the the best results were obtained using 2 centroids. Finally, columns 14-16 show the solutions provided by Selimi *et al*. [3], which constitute the best known solutions (BKS) in the literature. A row is added at the end of the table to display the average value for each column.

According to the results, the MCOP method provides, on average, a bandwidth of about 47.45 MB using three replicas, investing a computational time of 3.36 seconds. MCOP outperforms in terms of bandwidth and computational cost the previously known BASP placement method. Regarding to the random placement, it has the least computational cost, but it achieves a poor bandwidth between nodes, which impedes the usability of micro-clouds for higher demanding services. Finally, the K-means method achieves a similar bandwidth and computational cost than the random method with the half number of replicas.

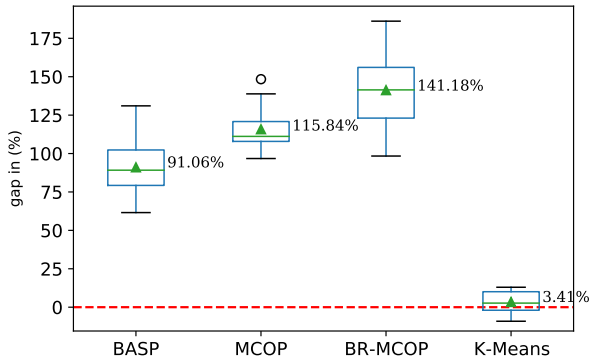
Figures 10a and 10b compare the different methods in terms of bandwidth. They reveal that the MCOP method improves both the current method used in *Guifi.net* (*i.e*., random method)

	MCOP			BR-MCOP			Random			K-Means			BASP		
Day	Bw [Mbps]	Time [s]	R	Bw [Mbps]	Time [s]	R	Bw [Mbps]	Time [s]	R	Bw [Mbps]	Time [s]	R	Bw [Mbps]	Time [s]	R
21	44.08	3.63	3	47.47	27.75	4	20.88	0.47	7	21.24	0.101	2	36.22	15.31	4
22	44.10	3.77	3	50.21	65.64	3	20.80	0.83	7	23.5	0.1	2	43.03	14.84	4
23	45.82	3.15	3	54.20	61.60	3	20.44	0.23	4	22.35	0.1	2	47.23	14.74	4
24	49.54	3.57	4	48.40	43.13	3	22.78	0.22	4	23.33	0.99	2	42.50	14.64	4
25	49.48	3.26	4	58.28	24.25	3	20.72	0.12	5	22.19	0.97	2	41.19	14.22	4
26	47.48	3.14	3	50.13	25.33	3	22.92	0.08	5	23.53	0.1	2	43.48	14.71	4
27	50.89	3.67	4	59.79	22.79	3	24.39	0.52	5	22.16	0.1	2	50.21	11.00	4
28	49.86	3.69	3	56.22	13.60	4	20.08	0.51	5	22.25	0.99	2	36.71	14.70	4
29	45.85	4.00	4	54.09	37.54	3	21.90	0.62	3	24.34	0.98	2	35.38	15.32	4
30	47.71	3.63	4	56.30	25.09	3	24.25	0.51	4	22.84	0.99	2	42.59	14.28	4
31	47.14	3.80	2	47.71	59.58	4	23.55	0.90	7	22.21	0.98	2	44.54	15.37	4
Avg.	47.45	3.57	3	52.98	36.94	3	22.06	0.46	5	22.72	0.58	2	42.10	14.47	4

Table 4: Results for the MCOP, BR-MCOP, random, K-Means and BASP methods.



(a) Comparison with respect to the BKS.



(b) Comparison with respect to the current method used in *Guifi.net*.

Figure 10: Comparison between the methods in terms of average bandwidth.

and the BKS (obtained with the BASP method). In particular, the MCOP method outperforms the current method used in *Guifi.net* obtaining a gain of about 115%, while saving 40% of the resources (nodes). Hence, it can be stated that the current placement method used in *Guifi.net* leads to an inefficient use of resources and, consequently, to a sub-optimal performance. With respect to the BASP method, the MCOP method improves the quality of the solutions, obtaining a gain of about 13%, while saves 25% of the resources. The obtained results confirms that the MCOP method is suitable to be used in this kind of networks, providing good solutions in real time.

In order to help the designer in the selection of the most appropriate method for a specific service placement problem, Fig. 11 compares the five methods from different perspectives: average bandwidth, number of replicas (R), efficiency, and computational time. The efficiency is measured as the average bandwidth divided by the number of replicas. It provides insights into how efficiently the methods are using the resources of the system. The BR-MCOP method outperforms the other methods in terms of bandwidth, reaching an average of 52.98 Mbps, at the expense of requiring more computational time. The optimal placement for a specific day and and the case of three replicas is computed by exploring all possible solutions. With this brute force approach the best placement obtained an average bandwidth of 62.7 Mbps and in comparison with this optimum the BR-MCOP method is around 15% lower. On average, the

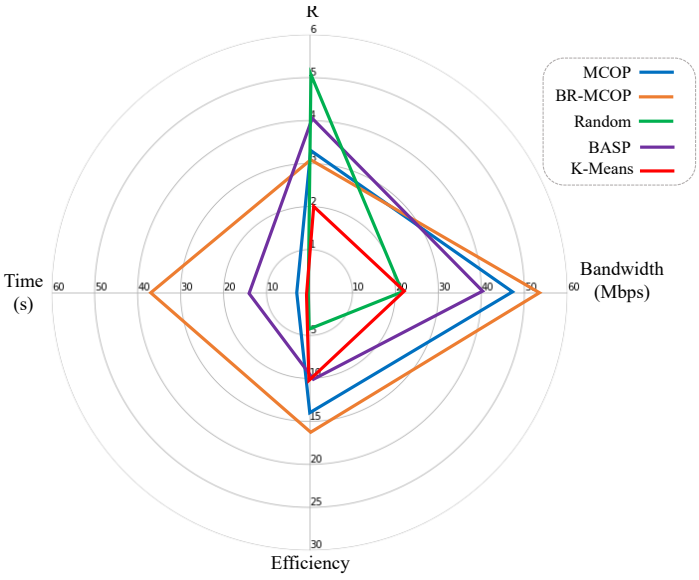


Figure 11: Comparison of the different approaches from different perspectives: bandwidth, number of resources, time, and efficiency

MCOP method needs 3.60 seconds to find the best solution, while the BR-MCOP method needs about 37 seconds. Concerning the number of resources used (replicas), the BR-MCOP method maintains the number or resources with respect to the MCOP method, being the random method (which has no intelligent behaviour) the approach that used a higher number of replicas. Regarding to the K-Means method, it can be observed that it achieves a similar behavior to the random method, both in average bandwidth and computational time, saving a 60% of resources.

Comparing the efficiency of the two proposed methods, the BR-MCOP method offers the highest efficiency. Finally, the Wilcoxon signed-rank test has been applied. It is a non-parametric statistical hypothesis test used to assess if the population mean ranks of two related samples, matched samples, or repeated measurements differ. The population mean ranks of the efficiency values of the BR-MCOP and MCOP methods are not significantly different ($p > 0.2$). In contrast, the population mean ranks of the efficiency values of the BR-MCOP and BASP methods are indeed significantly different ($p < 0.001$).

In conclusion, the proposed methods provide good quality solutions in terms of bandwidth and replicas used in a reasonable time in comparison with both the current method used in *Guifi.net* (*i.e.*, random) and the BASP method. Although the MCOP method returns good quality solutions in a few seconds, in scenarios where the micro-cloud is a key service and the performance is a critical point, the BR-MCOP method should be used instead, since in these scenarios is justified to invest more computational time to place services.

5.5. Experiments in the GuifiSants Network

5.5.1. Testbed

In order to conduct the performance evaluation of the MCOP and BR-MCOP methods in a real production mesh network, several x86 mini-PCs and Raspberry Pi boards have been installed in a wireless mesh network part of *Guifi.net* (at users' homes) to form a testbed. The wireless mesh network is GuifiSants located in the city of Barcelona, Spain. Nodes and network topology can be found at [2].

Fig. 12 illustrates the deployed testbed and provides some information about the network topology. Ten red nodes correspond to Raspberry Pi (RPi) 3B+ devices (Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 1GB RAM) running Raspian OS. Most of the RPi devices are geographically far from each other with a few hops of wireless links between them. They are located at users home.

Node Selection: The location of three RPi devices highlighted with red circle in Figure 12 is chosen based on the output of the MCOP and BR-MCOP methods. This corresponds to the top-ranked nodes (*i.e.*, cluster heads) selected from the MCOP and BR-MCOP; with higher bandwidth, availability and CPU resources. The other RPi nodes are used for comparison purposes (Nodes selected from random and K-Means heuristic). The MCOP and BR-MCOP are compared with the BASP, the K-Means and the random heuristic (*i.e.*, the existing in-place and naturally fast strategy in the *GuifiSants* network).

Network performance: MCOP vs. Random Placement

Before running real user services (*e.g.*, Web Server) in the candidate nodes selected by MCOP and Random method, their network performance (wireless links) are characterized by studying their RTT (round-trip-time) and bandwidth. Fig. 13a and Fig. 13b show the average RTT distribution of the nodes selected with the random and MCOP methods. Fig. 13b depicts that the gain brought by the MCOP method is 70% higher compared to the random method in terms of round trip time distribution. The average RTT of the node selected with MCOP is 9.3 ms compared to random with 31.8 ms. This improvement can be critical for delay-sensitive application in the network (*e.g.*, live video streaming, VoIP, games etc).

Fig. 14a and Fig. 14b depict the average bandwidth distribution of the nodes selected with the MCOP and random methods, respectively. Fig. 14a reveals that the average bandwidth of the node selected by the random method is 9.3 Mbps and for the node selected with the MCOP method is 104 Mbps (91% gain). This is due to the fact that MCOP aims to maximize the overall bandwidth among the service instances. This is critical when placing bandwidth-intensive services (*e.g.*, distributed storage, video-on-demand etc).

5.6. Evaluation of End-User Services

The service used to quantify the gain of the MCOP and BR-MCOP methods is the Web 2.0 service which mimics a social networking application (*e.g.*, Facebook). The content of the Web 2.0 website is dynamically generated from the actions of multiple users. For the evaluation, the dockerized version of the CloudSuite[3] Web Serving benchmark is employed. Cloudsuite benchmark has four tiers: the web server, the database

[2] `http://dsg.ac.upc.edu/qmpsu/index.php`
[3] https://www.cloudsuite.ch/

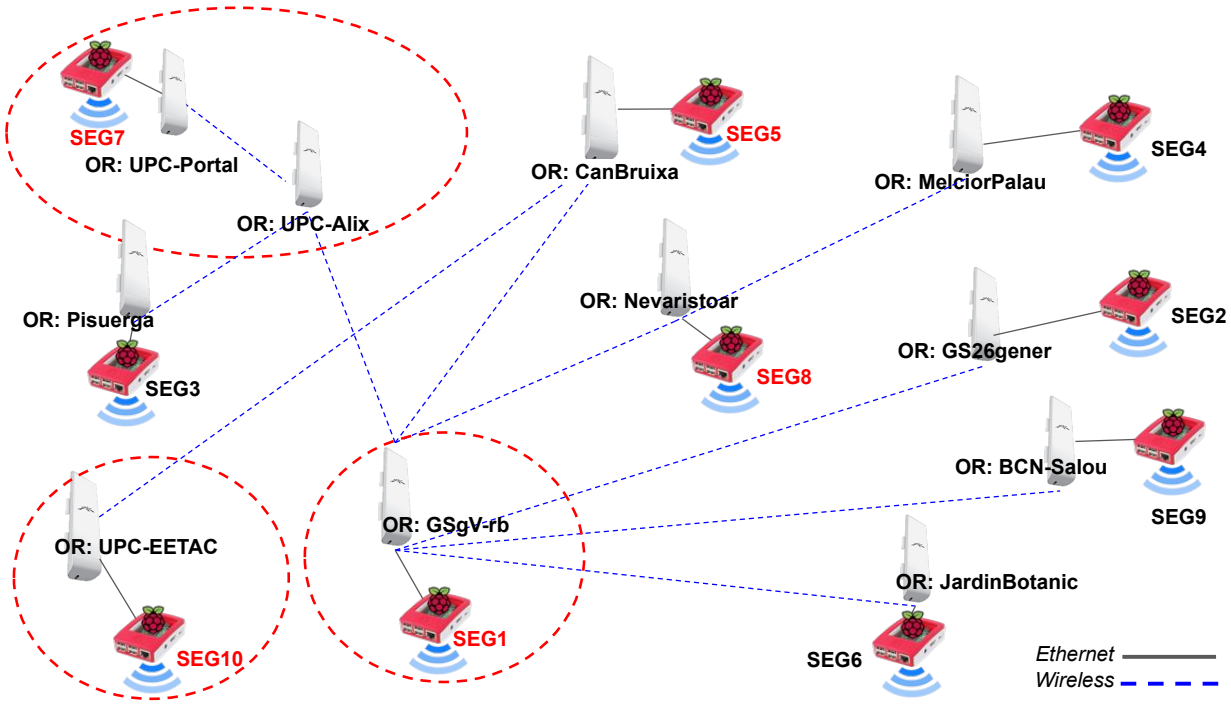


Figure 12: The testbed is deployed in the *GuifiSants* wireless mesh network in Barcelona, Spain.

server, the memcached server, and the clients. Each tier has its own Docker image. The web server runs Elgg[4] social networking engine and it connects to the memcached server and the database server. The clients (implemented using the Faban workload generator) send requests to login to the social network and perform different operations. In this type of service, the placement of the web server (together with the database server) is decisive for the user QoS. To quantify this, on the client side, we measure the response time when different number of clients are performing some operations such as login, live feed update (updating Facebook wall) etc. The web server is placed on a random node (Random method), on K-Means cluster head (K-Means) and on the node that MCOP method suggests.

Fig. 15 depicts the response time observed when up to 80 clients are performing update operations in the Elgg application. The average response time that clients perceive when web server is placed with K-Means is 3.7 seconds and with the random method is 4.1 seconds. Further, Fig. 15 reveals that, the client response times for higher workloads decreases an order of magnitude when using our MCOP method compared to Random and K-Means approach (average response time for MCOP is 0.35 seconds). It can be noticed that the gain brought by the MCOP method is higher for more intensive workloads (*i.e.*, higher number of clients).

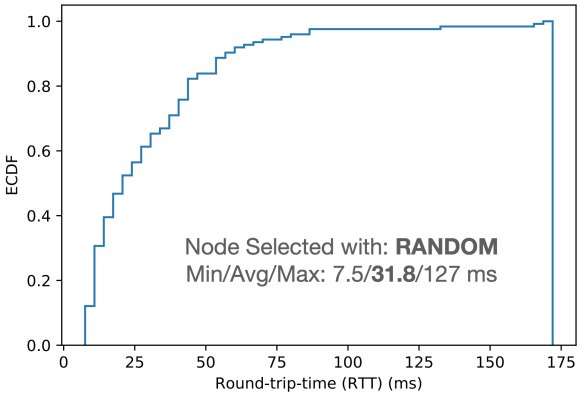
5.7. Discussion

Although the state-of-the-art methods discussed in this paper (e.g., BASP, K-Means, Random) provide good quality solutions, a key point to obtain good performance in terms of bandwidth is to select an optimal value for the k input parameter. It is important to notice that a higher value of k could lead to an inefficient use of the resources. Hence, more than one execution could be needed to assess the trade-off between performance and resources used.

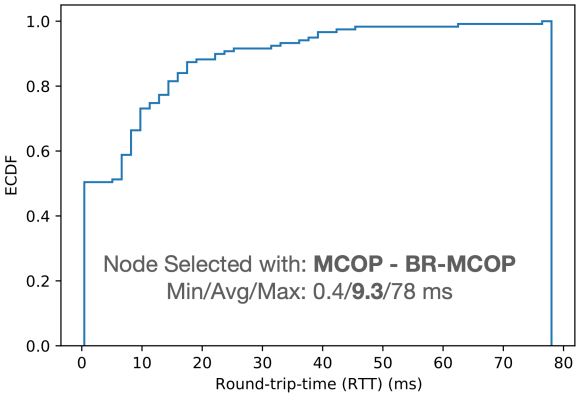
Figure 16 depicts the intra-cluster bandwidth obtained with different service placement heuristics. It can be seen that the gap between some heuristics (BR-MCOP) slightly grows as k increases, and for others it does not change at all. Thus, we can conclude that the increase of k does not represent a growth in the performance for all heuristics.

The methods proposed in this paper (MCOP and BR-MCOP) attempt to overcome this issue, selecting *the best value for k automatically*, aiming to reduce the number of resources used, considering a minimum loss of performance. Notice that the MCOP method, which is the baseline of the BR-MCOP method, provides a good quality solutions, both in terms of efficiency and average bandwidth, in about three seconds, outperforming the state-of-the-art methods. Therefore, it can be an applied as an effective method in real life networks. As a counterpart, the proposed methods use a prediction quality model, which uses historical data to compute the quality of a node. Hence, when a new node is added to the network, the proposed methods might not take into account the node, and be discarded as a candidate node to place services, until it gathers enough data to provide a reliable quality prediction of the node. The state-of-the-art methods overcome this limitation allowing to place services as soon as a new node is included in the network. For instance, the BASP [3] method, which provides the best known solution in the literature for *Guifi.net* network, is able to use all the nodes as soon as a new node is included in the network, since it is based on the physical distance of the nodes. Although this is a

[4]https://elgg.org/

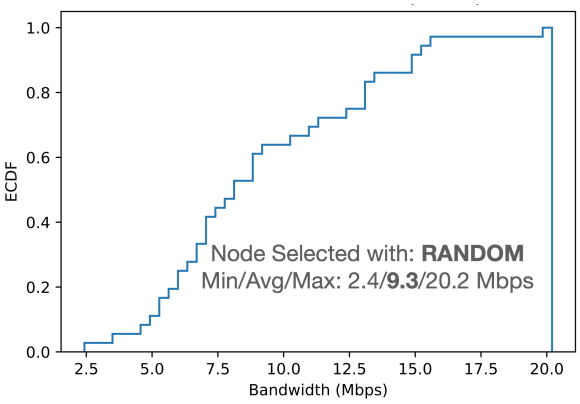


(a) RTT (Round-trip-time) link distribution of the node placed with the RANDOM heuristic

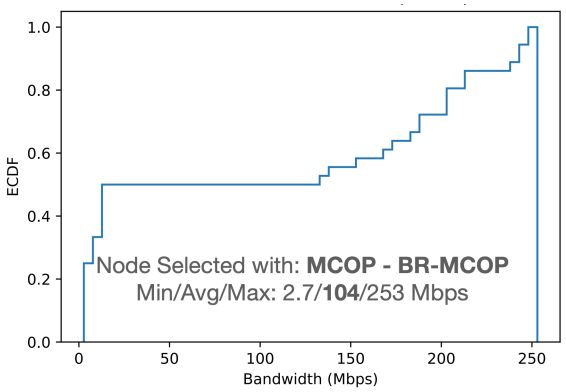


(b) RTT (Round-trip-time) link distribution of the node placed with the MCOP - BR-MCOP heuristic

Figure 13: RTT comparison when nodes placed with RANDOM and MCOP - BR-MCOP.



(a) Bandwidth link distribution of the node placed with the RANDOM heuristic



(b) Bandiwdth link distribution of the node placed with the MCOP - BR-MCOP heuristic

Figure 14: Bandwidth distribution comparison when nodes placed with RANDOM and MCOP - BR-MCOP.

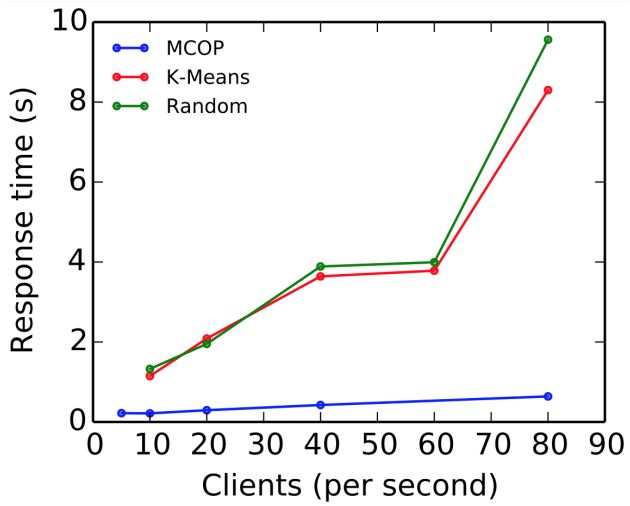


Figure 15: Response Time of Update activity: Random vs. K-Means Vs. MCOP.

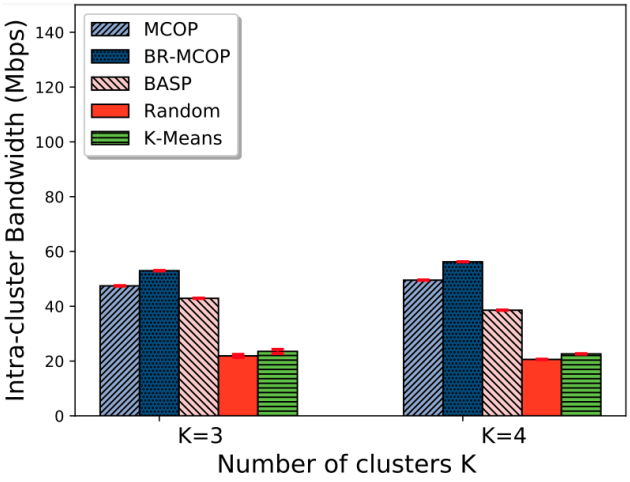


Figure 16: Comparison with the state-of-the-art methods.

limitation of our methods, it is not necessarily a problem, since as long as a new node does not reach the minimum requirements to place services, it should never be used. In this sense, the proposed methods are conservative with regards to new nodes and wait until a decision can be taken whether they should be used for the placement of services.

6. Conclusions and future research

This paper has motivated the need for advanced optimization methods to place services in community networks (CNs). In particular, a fast and efficient methodology, aware of the irregular topology, the heterogeneity of resources and their unreliable behaviour is required for suitable placement. Here we have presented the Multi-Criteria Optimal Placement (MCOP) method, which is able to deal with large-scale CNs. This method has been extended by means of biased randomization techniques to diversify the search and improve its performance, giving as a result the BR-MCOP method. A comprehensive set of computational experiments have been carried out under several scenarios using real traces of *Guifi.net*.

Several lines of future research stem from this work. For instance, it would be interesting to analyze the effect of stochastic variables (such as stochastic bandwidths or latencies) on the CNs' performance. This analysis would allow us to improve the reliability of the deployments. Another line worth exploring is to focus on the resilience of CNs and design procedures to guarantee critical services and divide the remaining resources among users when the CN is affected, for instance, by extremely bad weather conditions. Finally, due to the increasing number of IoT devices, it would be also purposeful to assess our placement methods in Fog computing environments.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities (PGC2018-097599-B-100).

References

[1] P. Micholia, M. Karaliopoulos, I. Koutsopoulos, L. Navarro, R. Baig, D. Boucas, M. Michalis, P. Antoniadis, Community networks and sustainability: A survey of perceptions, practices, and proposed solutions, IEEE Communications Surveys Tutorials (2018) 1–1.

[2] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, L. Navarro, A technological overview of the guifi.net community network, Computer Networks 93 (2015) 260 – 278, community Networks.

[3] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, J. Crowcroft, A lightweight service placement approach for community network micro-clouds, Journal of Grid Computing.

[4] R. Marler, J. Arora, Survey of multi-objective optimization methods for engineering, Structural and Multidisciplinary Optimization 26 (6) (2004) 369–395.

[5] A. A. Juan, J. Faulin, J. Jorba, D. Riera, D. Masip, B. Barrios, On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics, Journal of the Operational Research Society 62 (2010) 1085–1097.

[6] R. Baig, R. Roca, L. Navarro, F. Freitag, Guifi.net: A network infrastructure commons, in: Proceedings of the Seventh International Conference on Information and Communication Technologies and Development, ICTD '15, 2015, pp. 27:1–27:4.

[7] R. Baig, R. Roca, F. Freitag, L. Navarro, guifi.net, a crowdsourced network infrastructure held in common, Computer Networks 90 (2015) 150 – 165.

[8] M. Selimi, A. M. Khan, E. Dimogerontakis, F. Freitag, R. P. Centelles, Cloud services in the guifi.net community network, Computer Networks 93 (2015) 373 – 388, community Networks.

[9] R. Baig, F. Freitag, L. Navarro, Cloudy in guifi.net: Establishing and sustaining a community cloud as open commons, Future Generation Computer Systems.

[10] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, J. Crowcroft, Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking, in: Proceedings of the 2nd ACM Conference on Information-Centric Networking, ACM-ICN '15, 2015, pp. 9–18.

[11] R. P. Centelles, M. Selimi, F. Freitag, L. Navarro, Dimon: Distributed monitoring system for decentralized edge clouds in guifi.net, in: 12th IEEE Conference on Service-Oriented Computing and Applications (SOCA), 2019, pp. 1–8. doi:10.1109/SOCA.2019.00009.

[12] G. Wang, T. E. Ng, The impact of virtualization on network performance of amazon EC2 data center, in: Infocom, 2010 proceedings IEEE, IEEE, 2010, pp. 1–9.

[13] D. Ersoz, M. S. Yousif, C. R. Das, Characterizing network traffic in a cluster-based, multi-tier data center, in: 27th International Conference on Distributed Computing Systems (ICDCS '07), 2007, pp. 59–59.
[14] K. LaCurts, S. Deng, A. Goyal, H. Balakrishnan, Choreo: Network-aware task placement for cloud applications, in: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, 2013, pp. 191–204.
[15] J. Duan, Y. Yang, Placement and performance analysis of virtual multicast networks in fat-tree data center networks, IEEE Transactions on Parallel and Distributed Systems 27 (10) (2016) 3013–3028.
[16] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, Volley: Automated data placement for geo-distributed cloud services., in: NSDI, USENIX Association, 2010, pp. 17–32.
[17] H. Ghanbari, M. Litoiu, P. Pawluk, C. Barna, Replica placement in cloud through simple stochastic model predictive control, in: 2014 IEEE 7th International Conference on Cloud Computing, 2014, pp. 80–87.
[18] Y. Wu, M. Tornatore, S. Ferdousi, B. Mukherjee, Green data center placement in optical cloud networks, IEEE Transactions on Green Communications and Networking 1 (3) (2017) 347–357.
[19] F. Glover, Tabu search — part I, ORSA Journal on computing 1 (3) (1989) 190–206.
[20] O. A. Wahab, N. Kara, C. Edstrom, Y. Lemieux, Maple: A machine learning approach for efficient placement and adjustment of virtual network functions, Journal of Network and Computer Applications 142 (2019) 37 – 50. doi:https://doi.org/10.1016/j.jnca.2019.06.003.
URL http://www.sciencedirect.com/science/article/pii/S1084804519301924
[21] G. Rjoub, J. Bentahar, O. A. Wahab, Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments, Future Generation Computer Systems 110 (2020) 1079 – 1097. doi:https://doi.org/10.1016/j.future.2019.11.019.
URL http://www.sciencedirect.com/science/article/pii/S0167739X19304054
[22] G. Rjoub, J. Bentahar, O. Abdel Wahab, A. Bataineh, Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud, in: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), 2019, pp. 189–196. doi:10.1109/FiCloud.2019.00034.
[23] S. Pasteris, S. Wang, M. Herbster, T. He, Service placement with provable guarantees in heterogeneous edge computing systems, in: IEEE Conference on Computer Communications (INFOCOM 2019), Institute of Electrical and Electronics Engineers Inc., 2019, pp. 514–522. doi:10.1109/INFOCOM.2019.8737449.
[24] V. Farhadi, F. Mehmeti, T. He, T. L. Porta, H. Khamfroush, S. Wang, K. S. Chan, Service placement and request scheduling for data-intensive applications in edge clouds, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1279–1287.
[25] M. Steiner, B. G. Gaglianello, V. Gurbani, V. Hilt, W. Roome, M. Scharf, T. Voith, Network-aware service placement in a distributed cloud environment, in: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 2012, pp. 73–74.
[26] L. Zhang, X. Tang, The client assignment problem for continuous distributed interactive applications: Analysis, algorithms, and evaluation, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 785–795.
[27] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: Infocom, 2012 proceedings, IEEE, 2012, pp. 963–971.
[28] L. Jiao, J. Li, T. Xu, W. Du, X. Fu, Optimizing cost for online social networks on geo-distributed clouds, IEEE/ACM Transactions on Networking 24 (1) (2016) 99–112.
[29] Q. Xia, W. Liang, Z. Xu, The operational cost minimization in distributed clouds via community-aware user data placements of social networks, Comput. Netw. 112 (2017) 263–278.
[30] H. Khalajzadeh, D. Yuan, J. Grundy, Y. Yang, Improving cloud-based online social network data placement and replication, in: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), 2016, pp. 678–685.
[31] D. Vega, R. Meseguer, G. Cabrera, J. M. Marquès, Exploring local service allocation in community networks, in: 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014, pp. 273–280.
[32] M. E. Coimbra, M. Selimi, A. P. Francisco, F. Freitag, L. Veiga, Gelly-scheduling: Distributed graph processing for service placement in community networks, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, ACM, New York, NY, USA, 2018, pp. 151–160. doi:10.1145/3167132.3167147.
[33] Z. Al-Arnaout, Q. Fu, M. Frean, An efficient replica placement heuristic for community wmns, in: 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014, pp. 2076–2081.
[34] K. Herrmann, Self-organized service placement in ambient intelligence environments, ACM Trans. Auton. Adapt. Syst. 5 (2) (2010) 6:1–6:39.
[35] Ó. García, D. I. Tapia, S. Rodríguez, J. M. Corchado, Ambient intelligence application scenario for collaborative e-learning, in: N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, M. Ali (Eds.), Trends in Applied Intelligent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 407–416.
[36] K. Herrmann, K. Geihs, G. Mühl, Ad hoc service grid, in: E. Lawrence, B. Pernici, J. Krogstie (Eds.), Mobile Information Systems, Springer US, Boston, MA, 2005, pp. 261–274.
[37] W. Tarneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, E. Elmroth, Dynamic application placement in the mobile cloud network, Future Generation Computer Systems 70 (Supplement C) (2017) 163 – 177.
[38] G. Karagiannis, A. Jamakovic, A. Edmonds, C. Parada, T. Metsch, D. Pichon, M. Corici, S. Ruffino, A. Gomes, P. S. Crosta, et al., Mobile cloud networking: Virtualisation of cellular networks, in: Telecommunications (ICT), 2014 21st International Conference on, 2014, pp. 410–415.
[39] G. James, D. Witten, T. Hastie, R. Tibshirani, An introduction to statistical learning, Vol. 112, Springer, 2013.
[40] S. Even, Graph Algorithms, 2nd Edition, Cambridge University Press, New York, NY, USA, 2011.
[41] S. Hougardy, The floyd-warshall algorithm on graphs with negative cycles., Inf. Process. Lett. 110 (8-9) (2010) 279–281.
[42] R. Martí, J. A. Lozano, A. Mendiburu, L. Hernando, Multi-start methods, Handbook of Heuristics (2016) 1–21.
[43] E. Dimogerontakis, J. Neto, R. Meseguer, L. Navarro, L. Veiga, Client-side routing-agnostic gateway selection for heterogeneous wireless mesh networks, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 377–385. doi:10.23919/INM.2017.7987301.