Optimized Assignment Patterns in Mobile Edge Cloud Networks

Alberto Ceselli (alberto.ceselli@unimi.it)
Dipartimento di Informatica, Università Degli Studi di Milano, Crema, Italy

Marco Fiore (marco.fiore@ieiit.cnr.it)
CNR-IEIIT, Torino, Italy

Marco Premoli (marco.premoli@unimi.it)
Dipartimento di Informatica, Università Degli Studi di Milano, Crema, Italy

Stefano Secci (stefano.secci@upmc.fr)
UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

June 15th, 2017

Abstract

Given an existing Mobile Edge Cloud (MEC) network including virtualization facilities of limited capacity, and a set of mobile Access Points (AP) whose data traffic demand changes over time, we aim at finding plans for assigning APs traffic to MEC facilities so that the demand of each AP is satisfied and MEC facility capacities are not exceeded, yielding high level of service to the users. Since demands are dynamic we allow each AP to be assigned to different MEC facilities at different points in time, accounting for suitable switching costs. We propose a general data-driven framework for our application including an optimization core, a data pre-processing module, and a validation module to test plans accuracy. Our optimization core entails a combinatorial problem that is a multi-period variant of the Generalized Assignment Problem: we design a branch-and-price algorithm that, although exact in nature, performs well also as a matheuristics when combined with early stopping. Extensive experiments on both synthetic and real-world datasets demonstrate that our approach is both computationally effective and accurate when employed for prescriptive analytics.

Keywords: Mobile Edge Computing, Prescriptive Analytics, Generalized Assignment, Branch-and-Price.

1 Introduction

Communication networks have been undergoing a rapid evolution in the last few years, embracing software-based system virtualization as a way to optimize physical resource efficiency and network operational expenditures, and to increase the user's quality of experience. A particularly promising area of research is the one deriving from the installation of application servers into the access network, by means of a virtualization system, to decrease the network distance between users and servers, hence offering higher reliability and performance to network connection, besides an ideally lower energy foot-print on the Internet infrastructure. In telecommunications standardization bodies and open source code communities, this rising area is referred to as Mobile Edge Computing (MEC) [1].

Thanks to MEC deployment, the user experience can indeed be augmented because the server access latency can be drastically reduced, hence favoring collaborative networking, tactile Internet (i.e., communications made possible by an extremely low latency such as augmented reality applications) [2], and more generally computation offloading to a nearby cloud with the goals to control mobile device energy consumption and to run computationally heavy applications using tiny low-power devices [3].

Motivation and problem description In a MEC infrastructure, virtualization clusters – called 'MEC facilities' or 'MEC hosts' in the standardization documents, or cloudlets in academic jargon – are connected to access network nodes to deliver access to mobile application servers run as Virtual Machines (VMs). Various innovative operations to deal with changing mobile access demands can be applied and include Access Point (AP) to MEC facility dynamic assignment, VM capacity rescaling (addition or removal of computing power in terms of live memory or virtual processors) and VM migration (a VM state is moved from one MEC facility to another one). An 'orchestrator' is in charge of implementing such decisions into the MEC virtualization layer. Each orchestration action comes at a cost, often referred to as migration or switching cost, as it requires synchronizing states across a geographical network under stringent performance guarantees. The technology to perform MEC orchestration operations is becoming mature [4, 5]. However, the MEC orchestrator intelligence is still being developed, with as major goal to perform both reactive decisions to cope with sudden, unpredicted, changes, and proactive decisions to anticipate expectable network impairments.

We precisely address the MEC orchestration challenge from an algorithmic perspective. Our aim is to propose algorithms to take robust decisions about AP-MEC assignments and related traffic routing, while taking into consideration the corresponding VM switching costs.

More in details, our dynamic routing application contains a combinatorial core: APs have associated mobile traffic demand, that changes over time. Each MEC facility has a certain capacity, limiting the overall amount of demand it can serve simultaneously. APs must be assigned to MEC facilities; each assignment implies a cost for each user connected to the AP in terms of latency for communicating with the associated MEC server. Due to capacity limits it might be not always a good decision to assign each AP to its MEC facility of minimum latency; furthermore, since demand changes over time, an assignment pattern would hardly remain an efficient one over the whole planning horizon. We therefore leave the option of *changing* assignments over time, taking into account that each change implies a *switching* cost for the network, for example in terms of signaling to move session data of active users. An optimization problem therefore arises, that is to assign APs to MEC facilities over time, respecting capacity constraints and minimizing a combination of users (assignment) and network (switching) costs.

Literature Review Our final aim is to conceive a data-driven MEC management optimization framework: data analytics for network management is indeed a relevant subject to our research. This represents an emerging field in computer networks, fostered by the expected integration of analytics in the next generation of mobile networks [6]. In our framework we employ clustering techniques to unlink from noisy raw measurement data. In particular, we identify a limited number of typical configurations of the data traffic demand across the APs, thus understanding suitable time discretization patterns. As a matter of fact, the data traffic demand in mobile networks is characterized by significant fluctuations in space and time, due to the diverse activities of subscribers at different times and locations [7]. To accommodate such a variability, 5G systems will build on new networking paradigms such as Cloud Radio Access Networks (C-RAN), Software Defined

Networking (SDN) and Network Function Virtualization (NFV) that allow the dynamic (re-)allocation of resources [8]. There is thus a need for analytics that mine traffic metadata, discover relevant knowledge about the network status, and ultimately drive the resource management process [9, 10]. However, we currently miss substantial demonstrations of how data analytics can be leveraged in mobile network architectures.

Our proposal is to equip the system with an optimization core, exploiting preprocessed data as input, and producing solutions whose structure is explicitly encoded by mathematical programming models. Our setting requires to tackle a multi-period extension of the famous Generalized Assignment Problem (GAP) [11]. We point to [12] for a detailed review on the GAP and its extensions. Despite the large body of research available on the GAP, we are not aware of many papers directly dealing with its multi-period extensions. In [13] the authors face a single-source allocation problem with a flexible model and an effective branch-and-price algorithm; however, their model does not allow to handle limited capacity, which is a crucial feature in our application. The multi-period allocation problem discussed in [14], in which a dual ascent technique from [15] is adapted to a telecommunication networks applications, is similarly missing the feature of handling limited capacities.

Although our problem does not require to decide the location of the facilities, which is instead assumed to be optimized in a prior strategic planning [16] and given in input, one may expect features and computational challenges similar to those of multi-period location problems [17]. Recent approaches on that field include [18]: the authors face a multi-period concentrator location and dimensioning problem, providing MILP formulations and reduction techniques, and solving to optimality in less than one hour of computation instances with up to 30 clients, 10 candidate location sites and 15 time periods, or 100 clients, 30 candidate locations and 5 time periods. In [19] the authors introduce exact methods for a capacitated multi-period facility location problem in which however, unlike our case, the demand of each client can be fractionally served by multiple facilities. Large scale instances with up to 200 facilities, three periods and an arbitrary number of clients could be solved with their algorithms. We finally mention the recent contribution of [20], where the authors propose MILP formulations and local search heuristics for an uncapacitated p-median location problem involving two periods: in the first the location of facilities is given, while in the second it can be changed at a price. Clients are always assigned to the nearest facility. They provide good approximations to instances with up to 400 facilities in a few minutes of computation.

Contribution summary and paper outline Our research provides three types of contributions. The first is architectural: we design a data-driven framework for the dynamic selection of AP to facility assignment in MEC networks, which relies on *(i)* historical data clustering analytics, *(ii)* dedicated optimization techniques, and *(iii)* validation by simulation. The second is algorithmic: we introduce mathematical programming formulations and ad-hoc exact solution methods for a relevant multi-period extension of GAP. The third is use case-oriented: we evaluate our framework with real-world datasets, which provides practical insights for MEC resource management. We devote a substantial part of the paper to the technical insights of our optimization core.

The manuscript is structured as follows. We first detail our framework (Section 2). Then we focus on the optimization core component, introducing a compact Mixed Integer Linear Programming (MILP) formulation for our problem, proving a few structural properties and providing an extended counterpart by means of Dantzig-Wolfe decomposition (Section 3); we also design column generation procedures with ad-hoc pricing algorithms, rounding heuristics, reduction techniques and branching rules to be embedded in a whole exact solution method (Section 4). We show first of all that our algorithms are effective

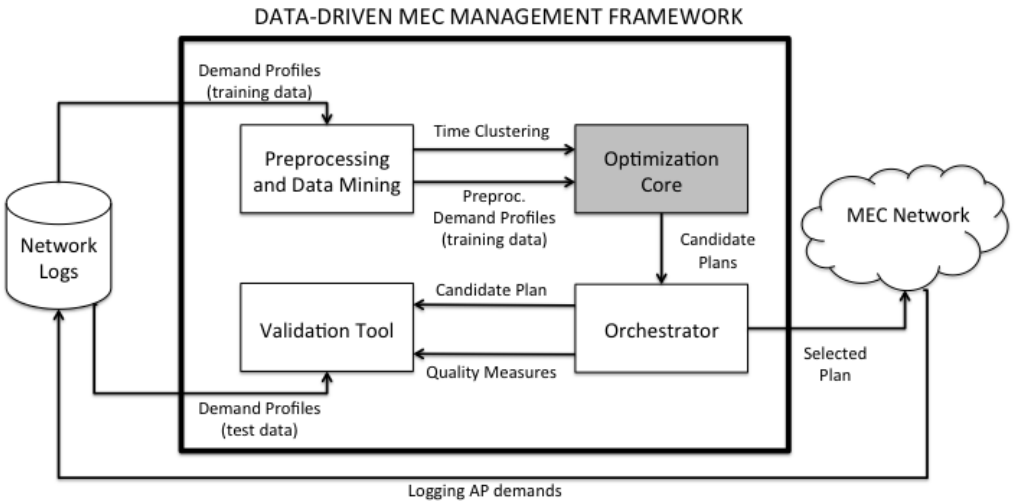


Figure 1: A data-driven MEC management optimization framework.

from a computational point of view (Section 5). We then demonstrate the effectiveness of our optimization tools in practical scenarios, using real-world traffic demands collected by a major mobile network operator in Milan, Italy (Section 6). We finally draw some conclusions (Section 7).

2 A data-driven MEC management optimization framework

We propose a high-level data-driven MEC management optimization framework whose algorithmic core architecture is sketched in Figure 1. The framework receives as input a representation of the (time-varying) data traffic demand recorded at each AP of the MEC network. The framework output are (multiple) assignment patterns of APs to MEC facilities, meant to be enforced by the orchestrator on the MEC network over time through switching operations.

The input data is collected as historical records of the mobile traffic demands in the MEC network. Part of the data is reserved for testing by the Validation Tool module. Part instead is used for training, *i.e.*, it is fed to the Preprocessing and Data Mining module, which filters by meta-data and produces a suitable time (and possibly space) discretization and a corresponding data aggregation. The aggregated profiles are sent to the Optimization Core module, which leverages them to compute candidate assignments. The Orchestrator module, in turn, receives the candidate assignments and queries the Validation Tool module for an evaluation on test data, so as to determine their quality. Based on the result, the Orchestrator module finally chooses a suitable assignment and implements it in the MEC network.

Preprocessing and Data Mining We assume that switching can occur only at certain points in time (*e.g.*, once every fifteen minutes), due to practical limitations of the MEC technology: this introduces an implicit time discretization of the system. In order to identify suitable discrete-time profiles of the traffic demand, different strategies can be employed in the Preprocessing and Data Mining module. The simplest option is to aggregate the demand observed at each AP during every time step in the training data. This returns one profile for each time step: since switching between assignments cannot occur at shorter timescales than the time step, this is the highest resolution useful to the Optimization Core – and the one deemed to return the highest-quality result. However, it

also creates a very large number of profiles (*e.g.*, in the order of thousands for hour-long time steps over months of historical data) that may be computationally too expensive to manage.

Another option to identify suitable discrete-time profiles of the traffic demand is to use temporal clustering analytics on the historical data, so as to group together time slots that feature very similar distributions of the mobile traffic demand across the APs. In this case, the module returns a limited number of profiles, each of which corresponds to the typical demand observed in a large set of time slots. It is then possible to reduce the computational cost at the Optimization Core, by feeding it with a small number of profiles. However, this comes at the expense of assignment quality, since typical profiles can only approximate the actual MEC network load at a specific time step. We provide an example of temporal clustering analytics, which builds on the methodology of [21], in Section 6.1.

Optimization Core Once the traffic demand profiles are defined, the Optimization Core builds effective dynamic assignment plans on top of those. Plans include, for each time slot, the set of APs to be connected to which MEC facility and, as a by-product, the set of switching operations to be performed between subsequent time slots. We consider different operational options:

- single versus split assignment: in the *single assignment* variant, an AP is associated to exactly one MEC facility in each point in time, while in the *split assignment* variant the AP demand can be served simultaneously by different facilities. That is, in the first case we suppose that, whenever connecting to a certain AP, each user is routed to the same facility, while in the latter case an AP can route the traffic of each user independently to different facilities.

- linear versus periodic assignment: in the *periodic* variant we explicitly take into account the potential switching cost between the last and the first point in time of our plan, since it is meant to be repeated over a longer planning horizon. This is not the case in the *linear* model, which assumes no system periodicity.

Validation Tool Once a set of candidate plans is produced by the Optimization Core, a Validation Tool is used to check their quality on test data. In our case, test data consist of a few weeks of raw traffic demand data: the Validation Tool evaluates the plan by simulating its application in those weeks, and computing quality measures.

Orchestrator The MEC orchestrator is the functional element in charge of actually sending VM orchestration instructions to MEC hosts, monitoring the MEC system status and the MEC network link states as well. Legacy cloud orchestrator systems typically take orchestration decisions based on simple best-fit policies, as data-center network resources are often over-dimensioned and a large set of clusters is made available; the placement and assignment decision logic is therefore typically not bound to computing facility location. Such orchestration algorithms however cannot be readily applied to a MEC context essentially because of the geographical nature of MEC networks and the capacity limitation of MEC facilities. This is typically done at the orchestrator subsystem level by adding an abstraction layer, with a dedicated descriptive language to map computing resources to physical location of servers.

3 Optimization models

A key component of the data-driven MEC management optimization framework is the optimization core. Its task is to find suitable assignments of APs to MEC facilities over

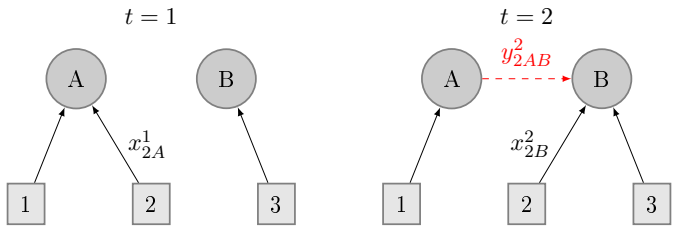


Figure 2: x and y variables

time, together with corresponding user VMs migration patterns. For the sake of readability, we focus on the single-assignment linear-plan variant: a discussion on how to adapt our models and methods to the other variants is provided in Section 4.6.

Let us denote as A the set of APs and as K the set of MEC facilities. We assume the planning horizon to be discretized in a set T of time slots. For each AP $i \in A$, let us indicate as d_i^t the mobile traffic demand that has to be accommodated by AP i at time $t \in T$, and as $m_{i,k}$ the physical distance between AP i and MEC facility $k \in K$. Let C_k be the capacity of MEC facility $k \in K$, and $l_{k',k''}$ be the network distance between MEC facilities $k', k'' \in K$. We assume $l_{k,k} = 0$ for each $k \in K$, where with network distance we mean a distance that can is directly proportional to the network latency (including packet processing latency at intermediate nodes) and the physical distance.

Let $x_{i,k}^t$ be binary variables taking value 1 if traffic from AP $i \in A$ at time $t \in T$ is routed to MEC facility $k \in K$, 0 otherwise. Let $y_{i,k',k''}^t$ be binary variables taking value 1 if AP $i \in A$ is associated to MEC facility $k' \in K$ at time $t-1$, and switches to MEC facility $k'' \in K$ at time t.

Our Dynamic Assignment and Switching Problem (DASP) can be formulated as follows:

$$\min \alpha \sum_{t \in T} \sum_{i \in A} \sum_{\substack{(j,k) \in \\ K \times K}} d_i^t l_{jk} y_{ijk}^t + \beta \sum_{t \in T} \sum_{i \in A} \sum_{k \in K} d_i^t m_{ik} x_{ik}^t \tag{1}$$

$$\text{s.t.} \sum_{i \in A} d_i^t x_{ik}^t \leq C_k \qquad \forall t \in T, \forall k \in K \tag{2}$$

$$\sum_{k \in K} x_{ik}^t = 1 \qquad \forall i \in A, \forall t \in T \tag{3}$$

$$x_{ik}^t = \sum_{l \in K} y_{ilk}^t \qquad \forall i \in A, \forall t \in T \setminus \{1\}, \forall k \in K \tag{4}$$

$$x_{ik}^t = \sum_{l \in K} y_{ikl}^{t+1} \qquad \forall i \in A, \forall t \in T \setminus \{T\}, \forall k \in K \tag{5}$$

$$x_{i,k}^t \in \{0,1\} \qquad \forall i \in A, \forall k \in K, \forall t \in T \tag{6}$$

$$y_{i,k',k''}^t \in \{0,1\} \qquad \forall i \in A, \forall k', k'' \in K, \forall t \in T \tag{7}$$

the objective (1) aims at finding a trade-off between the minimization of network- and user-related costs. The former is generated by the change of AP-MEC facility associations in consecutive time slots, which produces control overhead due to the necessity of migrating VMs. The latter is instead the latency experienced by the user with the current AP-MEC facility association. Parameters α and β represent the relative weight of the network- and user-related costs in the objective function. Constraints (2) impose that the overall demand assigned to MEC facility k at time t does not exceed its capacity. Constraints (3) impose that each AP is connected to a single MEC facility during a time slot. Constraints (4) and (5) link x and y variables in a flow conservation fashion: when $x_{ik}^t = 0$, that is AP i is not assigned to MEC facility k at time t, they impose that no switching operation is

made; when $x^t_{ik} = 1$, instead, they impose that a single switching operation assigns i to k at time t and reassigns it at time $t+1$ (possibly involving the same MEC facility, in which case the switching cost is zero).

A sample instance with three APs (squares), two MEC facilities (circles) and two time-slots (left and right parts) is depicted in Figure 2: AP 2 is assigned to MEC facility A at $t = 1$ and MEC facility B at time $t = 2$, therefore a switching operation from A to B needs to be performed.

Model (1) – (7) has a few interesting features.

Observation 1 *The DASP can be seen as a multi-period generalization of the Generalized Assignment Problem (GAP).*

In fact, when $|T| = 1$, the DASP reduces to a GAP.

Observation 2 *For $t > 1$, constraints (3) are redundant.*

Indeed, for $t > 1$, they are implied by constraints (4) and (5) and $\sum_{k\in K} x^1_{ik} = 1$ for each $i \in A$. It is easy to check it by induction over t: for each $i \in A$, constraints (5) ensure that if a $k \in K$ exists such that $x^{t-1}_{ik} = 1$ then $\sum_{l\in K} y^t_{ikl} = 1$; then by aggregating constraints (4), we obtain that if $\sum_{k\in K}\sum_{l\in K} y^t_{ikl} = 1$ then $\sum_{k\in K} x^t_{ik} = 1$. In turn, since the x variables are binary, $\sum_{k\in K} x^t_{ik} = 1$ implies that a $k \in K$ exists, such that $x^t_{ik} = 1$. All we need to additionally enforce is the base case $t = 1$.

Proposition 1 *When all x^t_{ik} variables take integer values, the y^t_{ikl} variables also (automatically) take integer values in any feasible solution. The converse is also true.*

In fact, for each $i \in A$ and each $t \in T$, due to constraints (5) if $x^t_{ik} = 0$ then $y^{t+1}_{ikl} = 0$ for each $l \in K$. If $x^t_{ik} = 1$, assume by contradiction that a feasible solution exists, containing fractional y^{t+1}_{ikl} values; each fractional y^{t+1}_{ikl} will appear in a different constraint of family (4), that can be feasible only if $x^{t+1}_{ik} = 1$ for more than a single $k \in K$, violating constraints (3), yielding infeasibility and thus leading to a contradiction. The converse is trivially implied by both constraints (4) and (5).

That is, in the search for optimal solutions by means of algorithms exploiting continuous relaxations, branching on y^t_{ikl} variables is unnecessary.

4 Optimization Algorithms

Unfortunately, when the size of the MEC network is large, even solving the continuous relaxation of model (1) – (7) turns out to be computationally hard. Therefore, we devise an ad-hoc exact solution approach based on decomposition.

Following the Dantzig-Wolfe reformulation principle [22], let

$$\mathcal{P}^i = \{(x^t_{ik}, y^t_{ikl}) : (3), (4), (5), (6), (7)\}$$

represent the convex hull of the feasible region respect to constraints (3) – (7). Let Ω^i be the set of corresponding extreme integer points, and for each $p \in \Omega^i$ let $\tilde{x}^{t,p}_{ik}$ and $\tilde{y}^{t,p}_{ijk}$ be the coefficients encoding point p. Each element of $\mathcal{P}^i$ can be represented as a linear convex combination of points in Ω^i. Therefore we introduce a set of variables $z^p \geq 0$, expressing multipliers in such a combination, and we reformulate the continuous relaxation of (1) – (7) as the following *Master Problem* (MP):

$$\min \sum_{i\in A}\sum_{p\in\Omega^i}\left(\alpha\sum_{t\in T}\sum_{\substack{(j,k)\in\\ K\times K}} d_i^t l_{jk}\tilde{y}_{ijk}^{t,p} + \beta\sum_{t\in T}\sum_{k\in K} d_i^t m_{ik}\tilde{x}_{i,k}^{t,p}\right) z^p \tag{8}$$

$$\text{s.t. } -\sum_{i\in A}\sum_{p\in\Omega^i} d_i^t \tilde{x}_{ik}^{t,p} z^p \geq -C_k \qquad \forall t\in T, \forall k\in K \tag{9}$$

$$\sum_{p\in\Omega^i} z^p = 1 \qquad \forall i\in A \tag{10}$$

$$z^p \geq 0 \tag{11}$$

The MP has an exponential number of variables. We optimize it by column generation: we replace Ω^i by a small representative subset $\bar{\Omega}^i$ (see subsection 4.1) and we solve the Restricted Master Problem (RMP) obtained in this way; then, for each $i \in A$, we search if any element of Ω^i exists whose corresponding variable has negative reduced cost, by solving a *pricing problem* (see subsection 4.2): any such element is added to $\bar{\Omega}^i$ and the process is iterated. Otherwise we stop: the solution obtained by restricting to $\bar{\Omega}^i$ is optimal also for the full problem.

Such a solution provides a valid lower bound to the DASP. It might indeed be fractional. In such a case we run rounding heuristics (see subsection 4.3) to obtain a corresponding upper bound. If upper and lower bounds do not match, we first perform probing to potentially fix variables (see subsection 4.4) and then, when needed, we enter a recursive tree search phase (see subsection 4.5). Our algorithms can be easily adapted to the split-assignment and periodic-plan variants (see subsection 4.6).

4.1 Initialization

In order to populate the initial sets $\bar{\Omega}^i$, as well as obtaining an initial primal bound, we run a simple greedy heuristic that builds the solution time-slot by time-slot and AP by AP. The corresponding pseudo-code is reported as Algorithm 1. In particular, for each time slot, APs are sorted by non-increasing demand and each AP is associated to a profitable MEC facility following this order. The choice for the most profitable MEC facility to which to associate an AP i at time t follows these rules: let $\bar{k}$ be the MEC facility to which the AP i was associated in the previous time slot $t-1$:

1. if $t > 1$ and the demand of the AP i does not exceed the residual capacity of the MEC facility $\bar{k}$, assign i to $\bar{k}$;

2. otherwise, find the nearest MEC facility (in terms of distance m_{ik}) to which the AP demand does not exceed the residual capacity; if no such a MEC facility exists, stop in a FAIL state.

This algorithm always terminates in $\mathcal{O}(|T||A|\log(|A|)|K|)$ time. Unfortunately, as for a fixed t the problem is a special instance of GAP, even the problem of finding an arbitrary feasible solution is NP-Hard. Indeed, the algorithm might stop in a FAIL state, without producing feasible solutions. However, in our computational experiments that never happened.

Nevertheless, to complete the population of the initial RMP, we insert also a single dummy column of very high cost, having coefficient 0 in each constraint (9). This ensures RMP feasibility also after branching.

Algorithm 1 Greedy Binary AP-MEC facility assignment

$\bar{k}_a = \text{none}\ ,\ \forall a \in A$ {MEC facility associated to AP a in previous time-slot}
for all $t \in T$ **do**
  $A^s = sortDec(d_a^t | a \in A)$ {sort AP for non-increasing demand at time t}
  $c_k = 0\ ,\ \forall k \in K$ {used capacity of MEC facility k}
  for all $a \in A^s$ **do**
    $k = \bar{k}_a$ {first choice is the previous assignment}
    if $k = \text{none} \vee c_k + d_a^t \leq C$ **then**
      $k = nearestAvailable(a, d_a^t, c_k)$ {get nearest MEC facility with enough residual capacity}
    end if
    $x_{a,k}^t = 1.0$
    $c_k = c_k + d_a^t$ {update used capacity of chosen MEC facility}
    if $t > 0 \wedge k \neq \bar{k}_a$ **then**
      $y_{a,\bar{k}_a,k}^t = 1.0$
    end if
    $\bar{k}_a = k$
  end for
end for

4.2 Pricing algorithms

Let $\lambda_{t,k}$ be the (non-negative) dual variables corresponding to constraints (9), and η_i be the (free) dual variables corresponding to constraints (10).

For each $\hat{i} \in A$, the problem of finding the element of $\Omega^{\hat{i}}$ corresponding to the variable of minimum reduced cost can be formulated as follows:

$$\min \pi_i = -\eta_{\hat{i}} + \alpha \sum_{t \in T} \sum_{\substack{(j,k) \in \\ K \times K}} d_{\hat{i}}^t l_{jk} y_{\hat{i}jk}^t + \\ + \sum_{t \in T} \sum_{k \in K} \left(\beta d_{\hat{i}}^t m_{ik} + d_{\hat{i}}^t \lambda_{t,k}\right) x_{\hat{i},k}^t \tag{12}$$

$$\text{s.t.} \sum_{k \in K} x_{\hat{i}k}^t = 1 \qquad \forall t \in T \tag{13}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}jk}^t \qquad \forall t \in T \setminus \{1\}, \forall k \in K \tag{14}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}kj}^{t+1} \qquad \forall t \in T \setminus \{T\}, \forall k \in K \tag{15}$$

$$\boldsymbol{x} \in \{0,1\}, \boldsymbol{y} \in \{0,1\} \tag{16}$$

Proposition 2 *The pricing problem* (12) - (16) *possesses the integrality property.*

In fact, according to Observation 2, constraints (13) can be removed for $t > 1$, and constraints (14) used to replace $x_{\hat{i}k}^t$ in (15) and then removed. The remaining is basically a network flow matrix, which is known to be totally unimodular [22].

On one hand, Proposition 2 implies that the lower bound obtained by the MP through column generation is equivalent to that obtained by optimally solving the continuous relaxation of the original model (1) – (7). On the other hand, it allows to employ polynomial time Linear Programming solution algorithms, making us to expect the solution process to be fast. Indeed, we could exploit its structure even further, as the elements of Ω^i have a

Algorithm 2 Pricing Algorithm

for all $i \in A$ **do**
  $c_k^1 = a_{ik}^1 \ \forall k \in K$ {cost of path starting at MEC facility k}
  $p_k^1 = \{k\} \ \forall k \in K$ {path starting at node k' at time}
  for all $t \in 2..T$ **do**
    for all $k \in K$ **do**
      $k^* = \arg\min_{k' \in K}(c_{k'}^{t-1} + b_{ik'k}^t)$
      $c_k^t = c_{k^*}^{t-1} + b_{ik^*k}^t + a_{ik}^t$
      $p_k^t = p_{k^*}^{t-1} \cup \{k^*\}$
    end for
  end for
  $k^* = \arg\min_{k \in K}\{c_k^{|T|}\}$ {minimum reduced cost related to AP i}
  $\pi^* = c_{k^*}^{|T|} - \eta_i$
  if $\pi^* < 0 - \epsilon$ **then**
    add variable related to minimum cost path p_{k^*} to the model
  end if
end for

particular combinatorial interpretation: they correspond to all feasible *association* paths, that is sequences of MEC facilities to which the AP i is assigned in consecutive time-slots. More in details, we build a directed layered graph $G(N, A)$, with a layer for each time-slot, as follows. Each layer has one node for each MEC facility; each pair of nodes in consecutive layers are connected by an arc. Each node $(t, k) \in T \times K$, modeling the assignment to MEC facility k at time t, has an associated traversal cost given by $a_{ik}^t = d_i^t(\beta m_{ik} + \lambda_{t,k})$, while each arc connecting nodes (t, j) and $(t+1, k)$ has an associated traversal cost given by $b_{ijk}^t = d_i^{t+1}\alpha l_{j,k}$. We also add a dummy source σ (resp. sink τ) nodes, having one outgoing arc to each node in layer $t = 1$ (resp. one incoming arc from each node in layer $t = |T|$) of zero cost. Figure 3 sketches the structure of G for a certain AP $\hat{i}$ on a sample instance with two MEC facilities (A and B): a potential solution assigns i to A at time $t = 1$, to B at time $t = 2$ and so forth.

In fact, an optimal pricing solution corresponds to a shortest $\sigma - \tau$ path in G.

Proposition 3 *For each $i \in A$, the pricing problem can be solved in $O(|T||K|^2)$ time.*

In fact, for solving the pricing problems we devise a simple dynamic programming algorithm, which is presented as Algorithm 2.

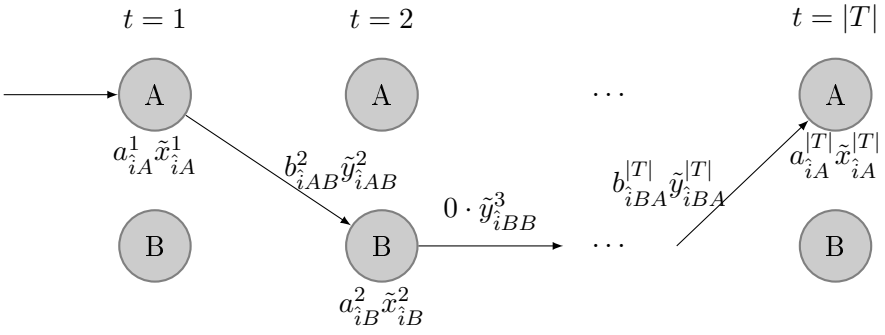


Figure 3: Pricing Problem Structure

Algorithm 3 Rounding heuristic

Input: variable values $\tilde{\boldsymbol{x}}$ from a RMP fractional solution
Output: $\hat{\boldsymbol{x}} = 0, \hat{\boldsymbol{y}} = 0$ {integer solution}
for all $t \in T$ **do**
  $r_k = C_k \ \forall k \in K$ {residual capacity of MEC facility k}
  $\tilde{A} = sort(A, \max_{k \in K} \tilde{x}^t_{ik})$ {sort the set of AP by non-increasing value of fractional assignment to the 'most desirable' MEC facility}
  for all $i \in \tilde{A}$ **do**
    {consider APs in such an order}
    if $\{k \in K | d^t_i \leq r_k\} = \emptyset$ **then**
      FAIL {no MEC facility with enough capacity: exit with FAIL status}
    else
      $k = \arg\max_{k \in K | d^t_i \leq r_k} \tilde{x}^t_{ik}$ {get highest assignment}
      $\hat{x}^t_{i,k} = 1.0$ {fix assignment with MEC facility}
      $r_k = r_k - d^t_i$ {update residual capacity}
    end if
  end for
end for
$\hat{\boldsymbol{y}} = compute\_shift(\hat{\boldsymbol{x}})$ {compute $\hat{y}$ variable values to be consistent with $\hat{x}$}

4.3 Rounding Heuristics

In order to find good primal bounds, a simple rounding algorithm (presented in Algorithm 3) is executed at every column generation iteration. Let $\tilde{\boldsymbol{z}}$ be the (possibly fractional) variable values of the RMP at a certain iteration: we can compute the values of the corresponding $\tilde{\boldsymbol{x}}$ variables as

$$\tilde{x}^t_{ik} = \sum_{p \in \Omega^i} \tilde{x}^{t,p}_{ik} \tilde{z}^p.$$

For each time-slot $t \in T$, for each APs i the highest $\tilde{x}^t_{ik}$ is retrieved. For each AP, sorted by descending highest $\tilde{x}$ value, the assignment is made with the MEC facility corresponding to the highest $\tilde{x}$ and with enough residual capacity. Although no guarantee in feasibility is given, our computational experiments revealed it to be highly effective.

4.4 Variables fixing.

We also experimented with probing techniques to potentially perform problem reduction during the column generation process. In particular, we employed Lagrangean probing to fix variables at a *pricing* level. The main idea is to run the Pricing Problem Resolution Algorithm twice: the first time as described in 2, that is considering each layer $t = 1 \dots |T|$ in forward order; the second time, instead, considering the layers in backward order, that is initializing $c^{|T|}_k = 0$ and updating each $c^t_k = \min_{k^* \in K} c^{t+1}_{k^*} + d^{t+1}_i(\alpha l_{k,k^*} + \beta m_{i,k^*} + \lambda_{t+1,k^*})$. In this way, the cost χ^t_k of the best path *in which at time t an assignment is forced to MEC facility k* can be computed by summing the forward and backward labels c^t_k.

A valid dual bound LB can be computed *at each column generation iteration* as

$$LB = \rho - \sum_{i \in A} \pi_i$$

where ρ is the value of the last RMP solution. Let UB be the value of the best primal (integer) solution found so far.

For each $i \in A$, let $s(t)$ be the MEC facility at which AP i has been assigned at time t in the optimal pricing solution returned by Algorithm 2. We perform the following fixes:

- for each $t \in T$ and $k \in K$ if $LB + \chi^t_{s(t)} - \chi^t_k \geq UB$, then if assignment to MEC facility k was forced, no improvement in the primal bound would ever be obtained. Therefore node k can be removed from layer t without losing optimization power, that means fixing variable $x^t_{ik} = 0$ in the original model;

- for each $t \in T$ if $LB + \chi^t_{s(t)} - \min_{k \in K \setminus \{s(t)\}} \chi^t_k \geq UB$, then if such an assignment was forbidden, no improvement in the primal bound would ever be obtained. Therefore all nodes $k \neq s(t)$ can be removed from layer t without losing optimization power; that means fixing variable $x^t_{is(t)} = 1$ in the original model.

A similar fixing procedure is run on arcs of the pricing graph, thereby allowing to fix $y^t_{k'',k'}$ variables in the original model.

From an implementation point of view, we always allowed a relative tolerance of 5e−4 in the fixing test, to prevent numerical troubles. We run the fixing procedure at the end of the column generation process of every node of the search tree; additionally, at the root node, we run it whenever an improving primal solution is found.

4.5 Branch-and-price

When upper and lower bounds at the end of the column generation process do not match, we proceed to branching. We branch on *original* variables $\boldsymbol{x}$ rather than on variables of the MP. Fixing variable $x^t_{i,k}$ to value 0 corresponds to fixing to value 0 all variables $z^p \in \Omega_i$ that assign AP i to MEC facility k at time t. Similarly, fixing variable $x^t_{i,k}$ to value 1 corresponds to fixing to value 0 all variables $z^p \in \Omega_i$ that do not assign AP i to MEC facility k at time t. Neither forbidding nor forcing assignments change the structure of the pricing problem: these conditions are easily included within the dynamic programming algorithm by simply removing nodes from the pricing graph. According to Proposition 1, no branching on y variables is needed.

We considered the following two branching rules:

1. considering all possible assignments of AP i at time t, take the pair (i', t') which has greatest number of variables $x^{t'}_{i',k}$ with strictly positive value, that is, that AP whose assignment at a certain time is split among the greatest number of different MEC facilities. Sort variables $x^{t'}_{i',k}$ by non-increasing value and partition this ordered set in two: the first set containing variables in the odd positions of ordered set and the second set containing variables in the even positions. A left (resp. right) branch is created, fixing to zero all the variables in the first (resp. second) set.

2. select the variable $x^t_{i,k}$ whose value is closer to 0.5, i.e. the variable related to the most fractional assignment. Create two branches fixing the selected variable respectively to value 0 or to value 1.

We always consider branching rule 1 first, triggering rule 2 only when all (i, t) pairs have at most two corresponding fractional $x^t_{i,k}$ variables. During preliminary experiments, a simple depth-first exploration policy showed to perform best. When rule 2 is used, the $x^t_{i,k} = 1$ branch is explored first.

4.6 Split assignment and periodic plans

We first observe that global optimal split-assignment plans can be obtained by simply stopping at the root node, and considering the (potentially fractional) solution of the column generation master problem.

Furthermore, as discussed in the Introduction, the application is periodic in nature: the decision maker creates a plan, that is meant to be repeated over time. Such a periodic variant can be managed by minor modifications to our models and algorithms. In particular, model (1) – (5) needs to be enriched, adding constraints

$$x_{ik}^{1} = \sum_{l \in K} y_{ilk}^{|T|} \qquad \forall i \in A, \forall k \in K$$

to the family (4) and constraints

$$x_{ik}^{|T|} = \sum_{l \in K} y_{ikl}^{1} \qquad \forall i \in A, \forall k \in K$$

to the family (5): these link the assignments made in the last and first time-slots assignment, hence closing the period of assignments. The column generation master problem does not change. The pricing problem, instead, requires to be adapted, as cycles rather than paths need to be generated. We therefore modified our pricing routine as presented in Algorithm 4. Exactly solving the modified pricing problem via dynamic programming requires $\mathcal{O}(|T||K|^3)$ time for every AP. The main idea is to tentatively fix the assignment at time 1 to all the $|K|$ possible MEC facilities, to solve each reduced problem, and to choose the best among the $|K|$ solutions found in this way. Since it is possible to create a layer $|T| + 1$ as a copy of the layer $t = 1$, after fixing the assignment the pricing problem reduces to finding a minimum cost shortest *path* from the single fixed node in layer $t = 1$ to the single fixed node in layer $|T| + 1$.

Any other detail of the algorithm remains unchanged.

5 Computational evaluation

We implemented our algorithms in C++, using CPLEX 12.6 [23] to solve the master LP subproblems, running tests on an Intel i7 4GHz workstation equipped with 32 GB of RAM.

Our first investigation is computational, benchmarking the effectiveness of our algorithms in comparison to the branch-and-cut ILP solver of CPLEX using formulation (1) – (7).

5.1 Dataset

We have access to a dataset of real-world mobile traffic demands [24], encompassing two months with a time granularity of fifteen minutes. The geographical area covered by the dataset extends for more than 2500 km^2. The demand is not associated to access points of the mobile network, whose location is unknown, but rather to a geographical tessellation of the area in 1419 rectangular cells of different sizes, with smaller (and more dense) cells in the center of the area. We select the centers of every rectangular cell as elements of the set A of access points locations.

Then, we create ten clusters of access points using a standard k-means model, taking as input the euclidean distances between APs, optimizing it with the classical heuristics of [25]. The centers of these clusters are selected to define the locations of the set K of MEC facilities. The network distances m_{ik} and l_{jk} are computed as euclidean distances accordingly, and rounded to the nearest integer (hence supposing packet processing latency is negligible).

Given this network infrastructure, we generate different problem instances by randomly drawing demands in each AP in different ways.

In details, we create two random datasets.

Algorithm 4 Periodic Pricing Algorithm

for all $i \in A$ **do**
  $\pi^* = +\infty$
  $p^* = \emptyset$
  for all $\hat{k} \in K$ **do**
    $c^1_{\hat{k}} = a^1_{i\hat{k}}$ {fix MEC facility $\hat{k}$ at $t = 1$}
    $c^1_k = +\infty \; \forall k \in K \setminus \{\hat{k}\}$ {forbid MEC facility k at $t = 1$}
    $p^1_k = \{k\} \; \forall k \in K$ {path starting at node k at time $t = 1$}
    for all $t \in 2..T$ **do**
      for all $k \in K$ **do**
        $k^* = \arg\min_{k' \in K}(c^{t-1}_{k'} + b^t_{ik'k})$
        $c^t_k = c^{t-1}_{k^*} + b^t_{ik^*k} + a^t_{ik}$
        $p^t_k = p^{t-1}_{k^*} \cup \{k^*\}$
      end for
    end for
    for all $k \in K$ **do**
      $c^{|T|}_k = c^{|T|}_k + b^1_{ik\hat{k}} + a^1_{i\hat{k}}$
    end for
    $k^* = \arg\min_{k \in K}\{c^{|T|}_k\}$ {minimum reduced cost related to AP i when starting at $\hat{k}$}

    if $c^{|T|}_{k^*} - \eta_i < \pi^*$ **then**
      $\pi^* = c^{|T|}_{k^*} - \eta_i$
      $p^* = p^{|T|}_{k^*}$
    end if
  end for
  if $\pi^* < 0$ **then**
    add variable related to minimum cost path p^* to the model
  end if
end for

Dataset A is synthetic, and aims at stressing our algorithms from a pure computational point of view. We consider a planning horizon of one day, split in 96 consecutive fifteen-minute time slots. Let $\underline{d}$ (resp. $\bar{d}$) be the minimum (resp. maximum) demand observed in any AP and time slot in [24]. Demands d^t_i for each AP i at time t are drawn uniformly at random independently in each fifteen-minute time slot, in the range $[\underline{d}, \bar{d}]$. That is, demands do not follow particular trends, even if falling into the same range of real data.

Dataset B is realistic, reproducing the main features of the starting data. We choose a single day at random from the two months included in the dataset [24], we perform a direct query to the demand of each AP at each time slot in that day, and then we perturb all demands with noise, uniformly drawn at random in the interval $[-5\%, +5\%]$.

All demand values are rounded to the nearest integer. Besides the initial horizon of 96 time-slots, we consider planning time horizons of 48, 24 and 12 time-slots by merging respectively 2, 3 or 4 subsequent time-slots, setting their demands as the average on the merged time-slots. Five instances are generated in both datasets A and B.

We also consider a dataset of raw real demands.

Dataset C is obtained by considering a random week taken from the dataset [24], and merging the time-slots in either 168 slots of 1 hour each (1h), 84 slots of 2 hours (2h), 56

slots of 3 hours (3h), 42 slots of 4 hours (4h) or 38 slots obtained by the clustering methods described in subsection 6.2 (clust). The demand of each AP in each slot t is taken as the maximum over the the fifteen-minute time slots merged in t.

For every instance, each MEC facility capacity C_k is set to $\left(\max_{t\in T}\sum_{i\in A} d_i^t/|K|\right)\cdot 1.05$, and parameters α and β are both set to value 0.5.

5.2 Column Generation profiling

We first report on the computational behaviour of our Column Generation algorithm (CG). In this test we consider the single-assignment non-periodic variant.

In Tables 1a, 1b and 1c we include the details of the root node column generation process, for each instance of datasets A, B and C, respectively. A best known solution value z^* is taken from a previous run of exact algorithms (see subsection 5.3). Besides instance details (columns '$|T|$' and 'inst'), we include the relative gap between the primal bound value (resp. dual bound value) and z^*, the number of column generation iterations needed to reach convergence, the overall CPU time spent for solving the pricing problems and the CPU time required to complete the column generation process. As benchmark we also report the performances of CPLEX 12.6.3 ILP solver, when stopped at the root node, including the corresponding primal and dual bound gaps and the time required to complete its root node computation.

We set a time limit of two hours to each computation, marking in the tables as 'T.L.' those computations hitting that limit.

We first note that CG has good convergence behaviour: less than 90 iterations are always enough to complete the computation. The high number of pricing subproblems yields to a high number of generated columns, but thanks to our dynamic programming algorithm, the overall pricing time remains low (below 10 seconds in all cases but one).

By rounding in CG we are always able to obtain good integer solutions (that is, below 1% from best known solutions in all cases but 4). CPLEX is not consistent: in a few instances (e.g. block $|T| = 48$ of the Realistic Dataset B) it is able to find very good primal solutions, while in other cases (e.g. block $|T| = 24$ of Dataset B, or block $|T| = 48$ of Dataset A), only very weak primal bounds can be obtained.

We also observe that CG and CPLEX dual bounds are always similar. That is, on one hand the integrality property of our pricing problem warns that no improvement can be obtained by CG with respect to the continuous relaxation of the original formulation; on the other hand, CPLEX generic cuts have no significant effect on strengthening the same continuous relaxation bound.

Finally, in more than 37% of the instances, CPLEX is unable to terminate the root node computation within the time limit, while CG always completes the computation. When both CG and CPLEX terminate, the CPU time required by CG is up to two orders of magnitude lower that required by CPLEX.

We highlight that the (possibly fractional) solution found by CG is a global optimal solution for the split-assignment model variants: no further computing is needed in that case.

5.3 Exactly solving the DASP

In a second round of experiments we let both our Branch-and-Price (BaP) and CPLEX 12.6 ILP solver (CPX) run for two hours, also exploring their branching trees. In Tables 2a, 2b, 2c we report the results of this experiment on each instance of datasets A, B and C, respectively. In each Table we report the relative gap between the primal bound PB (resp. the dual bound DB) at the end of computation and the best known integer solution value z^*, and the number of explored branch-and-bound nodes, for both our BaP and CPX.

		CG Root						CPLEX Root		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# iter	# cols	t^p	t	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	t
12	1	0.130%	1.212%	19	10808	0	4	0.724%	1.197%	864
	2	0.735%	1.955%	14	8715	0	4	244.932%	1.949%	410
	3	0.793%	1.351%	13	7074	0	3	0.167%	1.345%	328
	4	0.945%	0.999%	14	7515	0	3	2.824%	0.994%	344
	5	1.206%	1.630%	13	7892	0	3	3.052%	1.627%	356
24	1	0.061%	1.475%	26	17027	1	15	0.249%	1.471%	2826
	2	1.202%	1.537%	19	11429	1	11	274.940%	1.533%	1943
	3	0.867%	2.019%	19	12296	0	13	267.368%	2.015%	1852
	4	0.846%	1.864%	19	12525	0	12	123.542%	1.860%	1861
	5	0.909%	1.764%	18	12077	0	12	267.486%	1.761%	1781
48	1	0.463%	1.819%	41	27035	6	114	246.767%	1.816%	T.L.
	2	0.221%	2.468%	32	20897	1	90	270.039%	2.465%	T.L.
	3	0.746%	2.336%	33	21468	4	95	269.274%	2.332%	T.L.
	4	0.295%	2.036%	31	20496	0	74	282.422%	2.033%	T.L.
	5	0.265%	2.514%	35	22707	0	106	258.676%	2.510%	T.L.
96	1	0.176%	2.274%	68	47500	5	949	259.445%	-	T.L.
	2	0.055%	2.633%	59	42621	6	913	-	-	T.L.
	3	0.034%	2.779%	61	43107	9	863	-	-	T.L.
	4	0.187%	2.528%	61	42692	5	850	-	-	T.L.
	5	0.124%	2.568%	60	41913	6	815	272.834%	-	T.L.

(a) Synthetic Dataset A

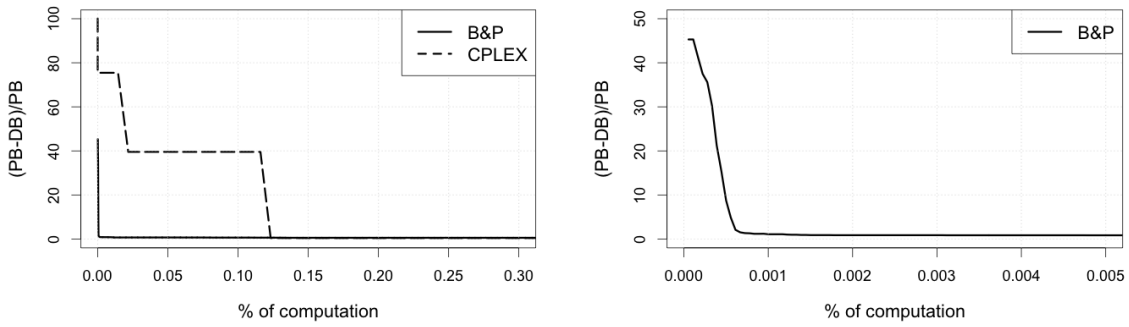
		CG Root						CPLEX Root		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# iter	# cols	t^p	t	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	t
12	1	0.809%	0.107%	12	6542	1	2	0.055%	0.105%	138
	2	0.917%	0.112%	11	6150	0	2	0.924%	0.110%	146
	3	1.043%	0.125%	12	6611	0	2	0.155%	0.123%	158
	4	0.797%	0.103%	11	5843	0	2	349.538%	0.102%	120
	5	0.781%	0.102%	11	6548	0	2	349.089%	0.101%	113
24	1	0.766%	0.425%	15	9713	0	4	305.781%	0.424%	532
	2	0.481%	0.280%	16	9715	0	3	306.562%	0.278%	587
	3	0.613%	0.576%	15	9492	1	3	304.188%	0.575%	622
	4	0.960%	0.183%	17	10345	0	3	306.874%	0.182%	615
	5	0.648%	0.221%	16	10217	0	4	307.199%	0.220%	413
48	1	1.031%	0.264%	27	15539	1	16	0.253%	0.262%	2113
	2	0.782%	0.344%	25	15636	2	15	0.557%	0.343%	2249
	3	0.500%	0.363%	26	15286	1	15	0%	0.362%	2674
	4	0.741%	0.314%	26	15791	3	18	0.049%	0.312%	2445
	5	0.829%	0.296%	29	16294	0	15	0.651%	0.295%	1878
96	1	0.077%	0.933%	51	29786	6	130	-	-	T.L.
	2	0.315%	0.766%	43	28272	3	125	-	-	T.L.
	3	0.249%	0.853%	46	28221	9	129	-	-	T.L.
	4	0.395%	0.692%	49	29320	5	126	-	-	T.L.
	5	0.154%	0.726%	48	29005	4	120	-	-	T.L.

(b) Realistic Dataset B

	CG Root						CPLEX Root		
inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# iter	# cols	t^p	t	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	t
clust	0.590%	0.092%	21	12928	2	9	0.429%	0.091%	468
4h	0.630%	0.272%	30	20690	2	25	0.175%	0.271%	5191
3h	0.513%	0.471%	37	25960	3	58	0.766%	0.470%	2814
2h	0.192%	0.724%	50	31706	2	133	-	-	T.L.
1h	0.097%	0.905%	85	56322	18	953	-	-	T.L.

(c) Raw Real Demand Dataset C

Table 1: Computational Results - CG Root vs. CPLEX Root



(a) BaP vs. CPLEX

(b) BaP Detail

Figure 4: Primal Bound

We report no computing time because, surprisingly, neither BaP nor CPX could bring the duality gap below 0.1% within the time limit, except for instance 4 with $|T| = 12$ of Dataset A, 5 with $|T| = 12$ of Dataset B and 'clust' of Dataset C, that CPX is able to close (but still using more than 90% of the available CPU time).

In terms of final dual bounds the results of both methods are very similar. Our explanation for this phenomenon is the following: the root dual bound is already very close to the integer optimum value, and therefore the real challenge is to find an optimal primal solution. At the same time, the high number of time-slots yields values of primal solutions on the order of magnitude of 10^8: as soon as the duality gap becomes small, numerical approximation issues prevent to coherently explore the remaining search tree.

In terms of searching for good primal solutions in the inner nodes of the branching tree, instead, BaP and CPX are not equivalent: in Figure 4a we plot the typical primal bound value (y axis) improvements as the computation (x axis) proceeds (instance 1, $|T| = 24$, dataset B); for the sake of comparison, the x axis report relative values with respect to the overall number of branch-and-bound nodes for CPX and the overall number of column generation iterations for BaP. Eventually, CPX is more numerically stable, offering after two hours of computation primal solutions values a few tenths of percentage points better (up to 0.5% of improvement). In turn, BaP allows to find near-optimal integer solutions much more quickly, that is in fact already at the root node. The behaviour of BaP in the early steps of computation is further detailed in Figure 4b: the quality of the primal bound steeply increase during the column generation iterations at the root node.

As a synthetic final assessment of our computational evaluation we can report that when the number of time-slots is small, and the planner has no particular need for quick optimization algorithms, both BaP and CPLEX might be viable alternatives to optimize the DASP.

BaP is faster, especially when used heuristically, stopping the computation either at the root node or after exploring a few nodes of the branch-and-bound tree. This makes it well suited also when quick optimization is needed, like in what-if-analyses.

When the number of time-slots increases, however, using CPLEX is not an option anymore.

We repeated all the experiments with a periodic-plan model, without finding substantial changes in the computational behaviour of the methods. Indeed, the periodic-plan model has main impact only in the structure of the pricing problem of BaP, but pricing involves only a small amount of the overall computational effort.

		B-&-P			CPLEX		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes
12	1	0%	1.208%	17661	0.411%	1.197%	203
	2	0%	1.951%	17038	0.777%	1.949%	371
	3	0.198%	1.347%	18572	0%	1.345%	452
	4	0.370%	0.997%	19198	0%	0.993%	548
	5	0.020%	1.626%	18013	0%	1.626%	699
24	1	0%	1.473%	8576	0.249%	1.471%	14
	2	0%	1.535%	8657	0.855%	1.533%	24
	3	0%	2.018%	9349	267.368%	2.015%	46
	4	0%	1.863%	9353	123.542%	1.860%	38
	5	0%	1.763%	10204	267.486%	1.761%	40
48	1	0%	1.818%	2123	246.767%	1.816%	0
	2	0%	2.468%	2724	270.039%	2.465%	0
	3	0%	2.335%	2558	269.274%	2.332%	0
	4	0%	2.036%	3133	282.422%	2.033%	0
	5	0%	2.514%	2528	258.676%	2.510%	0
96	1	0%	2.274%	113	259.445%	-	T.L.
	2	0%	2.633%	217	-	-	T.L.
	3	0%	2.779%	168	-	-	T.L.
	4	0%	2.528%	135	-	-	T.L.
	5	0%	2.568%	207	272.834%	-	T.L.

(a) Synthetic Dataset A

		B-&-P			CPLEX		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes
12	1	0.160%	0.105%	35112	0%	0.103%	2398
	2	0.235%	0.111%	30708	0%	0.109%	2579
	3	0.320%	0.124%	26374	0%	0.122%	1818
	4	0.406%	0.102%	26490	0%	0.100%	1932
	5	0.267%	0.100%	24714	0%	0.098%	1844
24	1	0.133%	0.424%	17961	0%	0.423%	144
	2	0.274%	0.279%	18354	0%	0.278%	109
	3	0%	0.576%	20016	0.717%	0.574%	225
	4	0.353%	0.183%	17481	0%	0.182%	232
	5	0.284%	0.220%	20446	0%	0.220%	522
48	1	0.312%	0.263%	10738	0%	0.262%	20
	2	0.249%	0.343%	10465	0%	0.343%	18
	3	0.113%	0.363%	10482	0%	0.360%	17
	4	0.227%	0.313%	10321	0%	0.311%	16
	5	0.226%	0.295%	10818	0%	0.295%	21
96	1	0%	0.932%	3133	-	-	T.L.
	2	0%	0.765%	3193	-	-	T.L.
	3	0%	0.852%	3398	-	-	T.L.
	4	0%	0.691%	3099	-	-	T.L.
	5	0%	0.726%	3299	-	-	T.L.

(b) Realistic Dataset B

	B-&-P			CPLEX		
inst.	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\lvert DB-z^*\rvert}{z^*}$	# nodes
clust	0.376%	0.091%	13849	0%	0.091%	812
4h	0.442%	0.272%	8985	0%	0.270%	30
3h	0.403%	0.471%	5086	0%	0.470%	33
2h	0%	0.724%	2864	-	-	T.L.
1h	0%	0.905%	370	-	-	T.L.

(c) Raw Real Demand Dataset C

Table 2: Computational Results - BaP vs. CPLEX

6 Practical case study

Our final aim is to assess the effectiveness of our optimization core in the context of the data-driven MEC management optimization framework. To this end, we rely on the complete real-world dataset in [24], and run actual analytics on it so as to generate the demand profiles. We then feed our optimization models with such profiles, which, ultimately, lets us measure the quality of the assignment plans it returns in a practical case.

6.1 Experimental setup

The real-world mobile traffic data covers a planning period of eight weeks, split in a set $\tilde{T}$ of fifteen-minute snapshots. The first approach we take in order to infer demand profiles exploits a well-known property of mobile traffic, *i.e.*, its weekly periodicity [26]. Namely, a single week is taken as training data, and the mobile traffic recorded in each fifteen-minute time-slot is considered as a profile. The tasks performed by the data mining and validation modules are kept as simple as possible, in order to highlight the effect of the optimization core.

Training Conceptually, the resulting $4 \times 24 \times 7 = 672$ training profiles are then used as input for our optimization algorithms. By optimizing over the training data we obtain a planning solution, which becomes our *assignment plan*: we apply such a plan to the remaining seven weeks of test data. As an example, optimizing over training data yields a solution including a specific assignment of APs to MEC facilities on Monday between 7:00 am and 7:15 am: we then blindly apply the same assignments on the 7:00-7:15 am time slots of each Monday in the test data, presuming that the demand configuration is similar on all Mondays at the same time. In a sense, this is a worst-case situation, in which the decision maker simply observes the system for one week before deciding on the planning for the remaining weeks.

Practically, we do not directly use the 672 fifteen-minutes time-slots for training. We assume, instead, that the preprocessing and data-mining module produces suitable aggregations. We experimented therefore with simple aggregation of the profiles, merging time-slots sequentially over 1, 2, 3 or 4 hours. Each aggregation step was performed by considering in each AP and in each aggregated time-slot the *maximum* demand value of that AP over the corresponding base time-slots. The optimized solution was then disaggregated in post-processing, simply replicating the same plan over the fifteen-minutes time-slots composing each aggregated time-slot. By choosing the maximum demand values during aggregation, we ensure that our optimized solutions remain feasible after disaggregation.

It is apparent that less aggregated profiles entail a higher potential for optimization: in the baseline scenario the assignment of APs to MEC facilities can be changed every 15 minutes, possibly assigning to the same MEC facility APs that experience very high peaks of demand in subsequent time slots. They are, however, more prone to overfitting, as peaks in particular hours of the training week do not necessarily correspond to peaks in the corresponding hour of testing weeks. In addition, they are more clearly expensive from a computational standpoint.

More aggregated instances, instead, are more conservative: two APs with very high demand peaks in the same four-hours time slot cannot be assigned to the same MEC facility, thus possibly preventing capacity violations in the test weeks if those peaks slightly move in time. They are also less demanding in terms of computational costs. However, aggregation forces the a same configuration for longer timeframes, making the optimization possibly oblivious of fast dynamics in the demand.

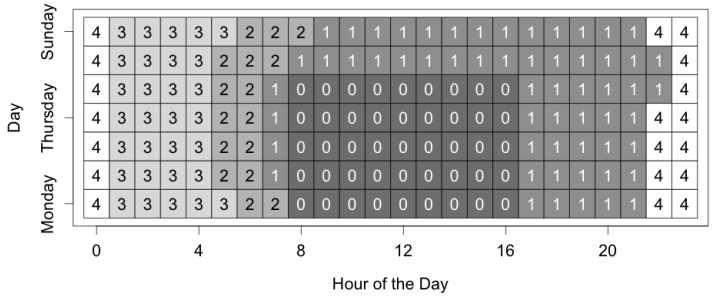


Figure 5: Time-Clustering resulting from [21]

In order to further balance the aggressive optimization process during the training phase of low aggregation instances, we found empirically useful to set the C_k capacity values to $\left(\max_{t\in T}\sum_{i\in A} d_i^t/|K|\right)\cdot 1.05$ in each test. That is, one-hour time-slot instances are optimized with lower capacity values.

Test To ensure fairness in the comparison, during the test phase only disaggregated solutions are considered. That is, independently on the aggregation used for training, only solutions defined over the original fifteen-minutes time-slots are compared. Similarly, the reference MEC capacity value was fixed to $\left(\max_{t\in \tilde{T}}\sum_{i\in A} d_i^t/|K|\right)\cdot 1.10$ in any comparison.

Clearly, neither the cost nor the capacity usage are guaranteed to remain the same when the planning obtained with training data is applied to test data, as fluctuations in the demand may occur across weeks. The quality of our solutions is therefore evaluated according to two measures: *(i)* the assignment and switching cost of the planning and *(ii)* the amount of violations in capacity constraints, both measured on test data. The latter is computed as follows:

$$\max_{t\in\tilde{T}} \sum_{k\in K} \frac{\max\{\sum_{i\in A} d_i^t x_{ik}^t - C_k, 0\}}{C_k}$$

that is, the overall amount of violation in capacity constraints, as a relative value respect to the available capacity, in the worst time-slot.

These measures are computed for each plan by the validation module through simple simulation on the seven weeks of test data.

Advanced Clustering Benchmark An alternative to employing our ad-hoc technique on the optimization module would be to perform a more aggressive aggregation over time, producing instances that are small enough to be optimized by CPLEX. We experimented on such an option, adapting the temporal clustering solution presented in [21] to our needs. Specifically, we take the following steps: *(i)* we generate a median week of mobile traffic demand, by computing the median load recorded at each AP during every hour of the week, using it as training data; *(ii)* we perform two separate hierarchical clusterings, respectively using the total volume and normalized distance metrics introduced in [21]; *(iii)* we find the intersection of the two cluster sets obtained at the previous point.

Figure 5 shows the resulting clustering of one-hour time slots: five typical demand profiles are identified, which can be associated to working hours (denoted by 0 in the figure), relax hours in the late afternoons and weekends (1), morning commuting hours (2), night hours (3) and late evening hours (4). Very few time-slots are actually produced this way, allowing CPLEX to optimize the corresponding instances. Consecutive time-slots

Inst.	Week							
	train	1	2	3	4	5	6	7
Bench-CPLEX	9.59%	7.03%	5.71%	4.25%	2.57%	3.88%	14.68%	9.51%
Bench-CG	9.59%	6.76%	5.48%	4.08%	2.45%	4.10%	14.34%	9.27%
4h	2.09%	1.54%	1.72%	0.66%	0.00%	2.54%	11.63%	2.92%
3h	1.94%	1.42%	1.48%	1.10%	1.25%	1.58%	11.07%	2.98%
2h	2.20%	2.58%	1.99%	1.19%	2.26%	2.82%	11.22%	4.15%
1h	2.84%	2.38%	4.17%	1.14%	1.92%	4.04%	10.65%	5.47%

Table 3: Time-Slot Peak Exceeded Capacity Over Available Capacity

belonging to the same cluster are then aggregated. The aggregated instance is optimized and the solution obtained in this way is then disaggregated in fifteen-minute time-slots plans, as in the previous case, during post-processing, before being compared on test data.

6.2 Experimental evaluation

Following the computational results of Section 5, our core optimization module always employs our Branch-and-Price as a heuristic (HBP), stopping its computation at the root node, except for the aggregated instances produced by time-clustering. Having $|T| = 38$, these are manageable efficiently even by general purpose solvers: CPLEX 12.6.3 ILP solver was then used to solve them to proven optimality. The scatter plot in Figure 6 summarizes our results. Each point represents the outcome on a single week of test data. Different shapes refer to different demand profile aggregation methods, obtained by merging 1, 2, 3 or 4 consecutive one-hour time slots, as well as using the Advanced Clustering and CPLEX (Bench-CPLEX). The y axis coordinate of each point represents the capacity violation measure, as introduced above; the x axis coordinate of each point represents the solution cost measure, expressed as percentage value of that of the optimal solution employing the Advanced Clustering Benchmark method. Therefore, negative (resp. positive) percentages map to a performance improvement (resp. reduction) with respect to the benchmark. As a reference we report also the results obtained by using HBP instead of CPLEX for optimizing the Advanced Clustering Benchmark instances (Bench-CG). The details about capacity violation measures are reported also in Table 3 for each week in the training set (columns) and for each demand profile aggregation method (rows).

A first remarkable result is that in all cases our HBP with sequential clustering allows for solutions with less capacity violations. In particular, HBP with either four-hours or three-hours aggregation offer almost no capacity violations, coming at the price of three to five percent increase in solutions cost. HBP with two-hours aggregation always outperforms the benchmark method both in terms of exceeded capacity and in terms of solution costs. HBP with one-hour aggregation further decrease costs at the price of slightly higher capacity violation.

We also experimented on using the full BaP instead of the truncated HBP. Our results are reported in Figure 7: no significant change was observed. Similarly, no significant change was observed by using HBP instead of CPLEX on the benchmark clustering.

6.3 Effect of periodic planning

Finally, we measure the impact in solution switching costs yielded by choosing a period model instead of a non-periodic one. In Table 4 we report, for each aggregation listed in the previous subsection, the change in switching costs between inner time-slots of the week (intra-week), between border time-slots of different weeks (inter-week), as well as the total switching cost, the total assignment cost and the overall cost of the solution. Using a

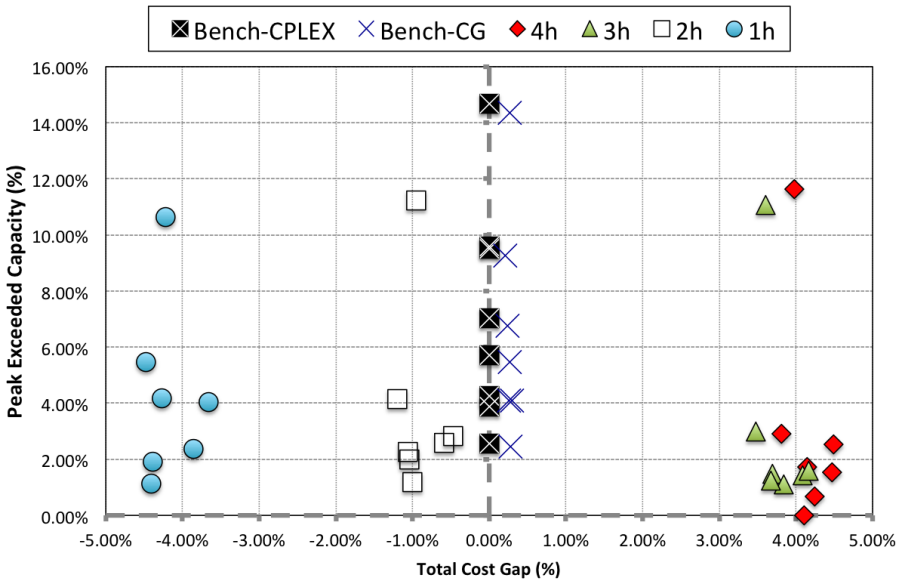
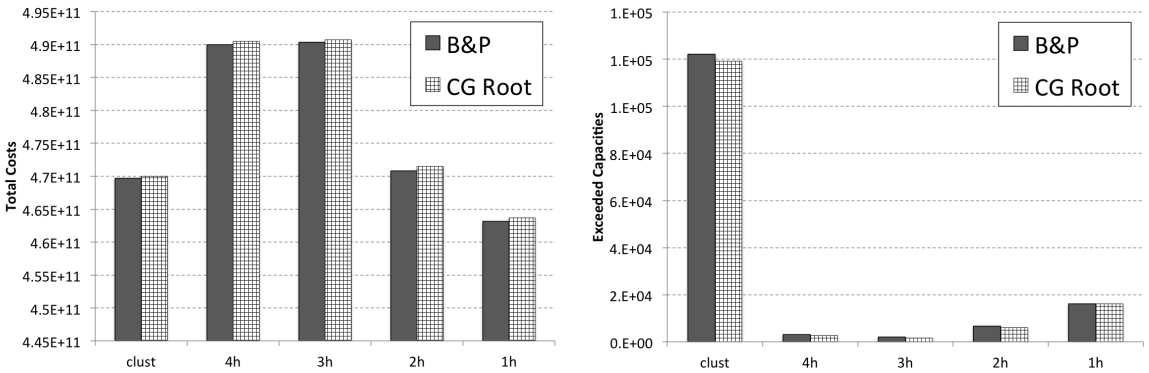


Figure 6: 1-Week Plan - Time Aggreg. Comparison



(a) Total Costs

(b) Exceeded Capacity

Figure 7: Mean Costs And Exceeded Capacity - CG Root vs. BaP

periodic model allows for huge improvements in inter-week switching costs, worsening intra-week and assignment costs only slightly. The effect on the overall total costs is however negligible, thus suggesting a form of asymmetry between assignment and switching costs in the objective function.

7 Conclusions

To tackle the complex problem of assigning APs to MEC facilities over time we have proposed a data-driven MEC management optimization framework, including an optimization core component, that is combined with preprocessing and data-mining and validation by simulation modules.

As a main result, we verified that instances arising in practical analyses strongly benefit from the explicit use of mathematical programming models in such an optimization core. The performances of the framework are enhanced even further when our ad-hoc algorithms are exploited: being much more effective than a general purpose solver like CPLEX, they allow to create candidate assignment plans with a finer time discretization; this proved to be beneficial in a training-and-test evaluation on real data.

From a computational point of view, although exact in nature, the main appealing feature of our algorithms is their ability of finding near-optimal integer solutions very quickly, providing at the same time good dual bound guarantees. This makes them well suited also for what-if analyses.

A promising future step is the tighter integration of temporal clustering and ad-hoc optimization algorithms, allowing for a higher number of time slots of potentially different size, obtained through data mining. From an application perspective, our good computational results open also the possibility of optimizing simultaneously more than a single service.

Acknowledgements

The authors wish to thank Angelo Furno and Razvan Stanica for their invaluable help in producing the time-slot aggregations of the Advanced Clustering Benchmark of Section 6. The project has been partially funded by Regione Lombardia - Fondazione Cariplo, grant n. 2015-0717, project `REDNEAT`.

References

[1] C. Chan N. Sprecher S. Abeta A. Neal M. Patel, B. Naughton. Mobile-edge computing introductory technical white paper. Technical report, Mobile-edge Computing (MEC) industry initiative., 2014. URL http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing/mec-poc.

inst.	inter-weeks	intra-week	tot. switch. costs	assign. costs	total costs
Bench-CG	803.11%	-16.53%	8.69%	0.02%	0.10%
4h	96.69%	4.20%	10.97%	-0.19%	-0.04%
3h	196.67%	2.60%	11.83%	-0.31%	-0.07%
2h	249.25%	-4.80%	2.81%	-0.08%	0.01%
1h	225.90%	-2.61%	1.67%	0.10%	0.18%

Table 4: Switching Cost Gap - Acyclic vs. Cyclic

[2] A. Aijaz, M. Dohler, A. H. Aghvami, V. Friderikos, and M. Frodigh. Realizing the tactile internet: Haptic communications over next generation 5g cellular networks. *IEEE Wireless Communications*, 24(2):82–89, April 2017. ISSN 1536-1284. doi: 10.1109/MWC.2016.1500157RP.

[3] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61, April 2017. ISSN 0163-6804. doi: 10.1109/MCOM.2017.1600863.

[4] Dirk Lindemeier. Mec proofs of concept. Technical report, European Telecommunications Standard Institute. URL http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing/mec-poc.

[5] S. Secci, P. Raad, and P. Gallard. Linking virtual machine mobility to user mobility. *IEEE Transactions on Network and Service Management*, 13(4):927–940, Dec 2016. ISSN 1932-4537. doi: 10.1109/TNSM.2016.2592241.

[6] Ec h2020 5g infrastructure ppp. pre-structuring model, version 2.0. Technical report, The 5G Public Private Partnership. URL https://5g-ppp.eu/5g-ppp-phase-2-pre-structuring-model/.

[7] A. Furno, M. Fiore, and R. Stanica. Joint spatial and temporal classification of mobile traffic demands. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017.

[8] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi. Mobile network architecture evolution toward 5g. *IEEE Communications Magazine*, 54(5):84–91, May 2016. ISSN 0163-6804. doi: 10.1109/MCOM.2016.7470940.

[9] H. Assem, T. Sandra Buda, and L. Xu. Initial use cases, scenarios and requirements. *H2020 5G-PPP CogNet, Deliverable D2.1*, 2015.

[10] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang. Big data-driven optimization for mobile networks toward 5g. *IEEE Network*, 30(1):44–51, January 2016. ISSN 0890-8044. doi: 10.1109/MNET.2016.7389830.

[11] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990. ISBN 0-471-92420-2.

[12] Dolores Romero Morales and H. Edwin Romeijn. The generalized assignment problem and extensions. In Ding-Zhu Du and Panos M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 259–311. 2005.

[13] Richard Freling, H. Edwin Romeijn, Dolores Romero Morales, and Albert P. M. Wagelmans. A branch-and-price algorithm for the multiperiod single-sourcing problem. *Operations Research*, 51(6):922 – 939, 2003.

[14] Ishwar Murthy and Phil K. Seo. A dual-ascent procedure for the file allocation and join site selection problem on a telecommunications network. *Networks*, 33(2):109 – 124, 3 1999. ISSN 1097-0037. doi: 10.1002/(SICI)1097-0037(199903)33:2<109::AID-NET3>3.0.CO;2-L. URL http:https://dx.doi.org/10.1002/(SICI)1097-0037(199903)33:2<109::AID-NET3>3.0.CO;2-L.

[15] Ishwar Murthy. Solving the multiperiod assignment problem with start-up costs using dual ascent. *Naval Research Logistics*, 40:325–344, 1993.

[16] S. Secci A. Ceselli, M. Premoli. Mobile edge cloud network design optimization. *IEEE Transactions on Networking*, 25(3), 2017.

[17] Stefan Nickel and Francisco Saldanha da Gama. Multi-period facility location. In Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, editors, *Location Science*, pages 289–310. Springer, 2015.

[18] Iric Gourdin and Olivier Klopfenstein. Multi-period capacitated location with modular equipments. *Comput. Oper. Res.*, 35(3):661–682, March 2008. ISSN 0305-0548. doi: 10.1016/j.cor.2006.05.007. URL http://dx.doi.org/10.1016/j.cor.2006.05.007.

[19] Jordi Castro, Stefano Nasini, and Francisco Saldanha-da Gama. A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method. *Mathematical Programming*, 163(1):411–444, 2017. ISSN 1436-4646. doi: 10.1007/s10107-016-1067-6. URL http://dx.doi.org/10.1007/s10107-016-1067-6.

[20] Russell Halper, S. Raghavan, and Mustafa Sahin. Local search heuristics for the mobile facility location problem. *Computers & Operations Research*, 62:210 – 223, 2015. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2014.09.004. URL http://www.sciencedirect.com/science/article/pii/S0305054814002512.

[21] Angelo Furno, Diala Naboulsi, Razvan Stanica, and Marco Fiore. Mobile demand profiling for cellular cognitive networking. *IEEE Transactions on Mobile Computing*, 16(3), March 2017.

[22] L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998.

[23] *IBM ILOG CPLEX 12.6 User Manual*. IBM corp., 2013. URL http://public.dhe.ibm.com/software/products/Decision_Optimization/docs/pdf/usrcplex.pdf. Accessed: 2016-11-01.

[24] G. Barlacchi and et al. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific Data*, 2(150055), 2015. doi: 10.1038/sdata.2015.55.

[25] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28 (1):100–108, 1979.

[26] Ram Keralapura, Antonio Nucci, Zhi-Li Zhang, and Lixin Gao. Profiling users in a 3g network using hourglass co-clustering. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 341–352, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0181-7. doi: 10.1145/1859995.1860034. URL http://doi.acm.org/10.1145/1859995.1860034.