Università Degli Studi di Milano
Department of Computer Science
Ph.D. in Computer Science [INF/01]

Ph.D. Thesis:

Mobile Edge Computing Network Optimization

Candidate:
Marco Premoli

Tutor:
Prof. Alberto Ceselli
Coordinator:
Prof. Paolo Boldi

XXX cycle
A.A. 2016/2017

Abstract

Smart mobile devices are becoming more and more important in every aspects of human life, and mobile application are becoming more and more resources demanding with a widening gap between the required resources and those available on mobile devices. To bridge this gap, *Mobile Edge Computing* paradigm has been introduced to bring IT applications, computational and storage resources to the periphery, or edges, of the cellular mobile network. Several complementary technologies have been presented to implement Mobile Edge Computing, all of them considering the deployment of virtualization facilities within mobile access and backhaul network.

In this thesis we face several optimization problems related to the planning of a Mobile Edge Computing Network. In the first part of the thesis we present the Mobile Edge Computing Network Design Problem (MNDP), that considers the design of a full mobile access and backhaul network together with the location of the virtualization facilities. Two variants of the problem are considered: either assuming a static condition of the network or dynamic variations of traffic demands and the human mobility that causes these variations. Matheuristics are proposed to solve MNDP and best practices are drawn on real world data.

In the second part of the thesis, we face a tactical side of the optimal Mobile Edge Computing network planning, that is the routing in time of access point traffic to specific Mobile Edge Computing facilities on a fixed network structure. We present exact Branch-and-Price algorithm to solve the problem, experimenting on real-world dataset.

Finally, driven by the fact that the knowledge of mobile user mobility represents a key data for the MNDP, in the third part of the thesis we face the problem of estimating human mobility given very aggregated data, that is the network traffic demand variations in time. We propose mathematical programming formulation and column generation algorithm to solve this problem, experimenting on both real-world and synthetic datasets.

Contents

List of Figures

List of Tables

Acronyms

AP Access Point.

b-MNDP Dynamic Planning Bulk VM Migration MNDP.

CDF Cumulative Distribution Function.
CG Column Generation.
CN Core Networks.

d-UTPP Demand Conservation Variant of the UTPP.
DASP Dynamic Assignment and Switching Problem.
DW Dantzig-Wolfe Decomposition.

EC Edge Computing.
ETSI European Telecommunications Standardization Institute.

f-UTPP Flow Conservation Variant of the UTPP.

ILP Integer Linear Programming.
IoT Internet-of-Things.

l-MNDP Dynamic Planning Live VM Migration MNDP.
LP Linear Programming.

MCC Mobile Cloud Computing.
MEC Mobile Edge Computing.

MILP Mixed Integer Linear Programming.

MNDP MEC Network Design Problem.

POI Points of Interest.

QoE Quality of Experience.

r-MNDP Dynamic Planning VM Replication MNDP.

RMP Restricted Master Problem.

s-MNDP Static Planning MNDP.

SLA Service Level Agreement.

UE User Equipment.

UTPP User Trajectories Prediction Problem.

VMs Virtual Machines.

Introduction

Mobile devices have become ubiquitous in people's everyday life, with a resulting striking growth of mobile connectivity and global mobile data traffic over recent years, and further expansion to be expected in the near future [2]. Mobile applications become more and more resource-hungry, with a widening gap between the required resources and those actually available on mobile devices.

In a first attempt to bridge this gap, cloud computing paradigm was proposed to expand mobile device resources: the Mobile Cloud Computing (MCC) concept has been introduced to allow User Equipment (UE) to offload computation and storage to distant centralized clouds facilities, connecting through the Internet. While this strategy allows to save battery consumption and enables a variety of applications, it also produces an increase in the latency experienced by the user and hence a worsening of her Quality of Experience (QoE).

The Edge Computing concept deals with such a phenomenon by moving cloud services closer to users. It provides IT and cloud-computing capabilities in the edge of network, in close proximity to mobile subscriber. In particular Mobile Edge Computing (MEC) focuses on the inclusion of virtualization facilities within the cellular access and backhaul network [3, 4] to which mobile users can connect to run cloud services on Virtual Machines (VMs).

While MEC has attracted a lot of attention in recent years for its beneficial prospects, it is still a rather immature technology and there are many challenges that need to be addressed to reach a successful implementation [5]: algorithms for the offloading decision from the UE to MEC facilities, standardization of communication protocols, security issues, connection with legacy networks, management of users' mobility, the optimal allocation of computing resources and the effective design of the MEC network. Moreover, while the term MEC (or Multi-Access Edge Computing) is used by industrial fora and standardization bodies, several complementary concepts have been presented in parallel, differing on the low-level details but being similar

in the general idea: the deployment of virtualization facilities close to the mobile user.

While extensive research has been undertaken for the optimal planning of telecommunication networks, the understanding of the particular features of MEC networks is still very limited. On the other hand, these particular features make general methods from the literature unsuitable.

In this thesis we face key methodological problems related to the planning of a MEC Network. We consider a high-level representation, thus abstracting from details on actual technologies. We employ (i) a mathematical programming approach, devising new formulations and algorithms and (ii) a data analytics validation, testing the suitability of our methods on real data.

The problems we face involve research on both modeling and algorithmic sides. The planning of a MEC network is indeed rich of application details: our modeling choices take into consideration the trade-off between the adherence to fine-grained features and the corresponding complexity added to the resulting model. In Chapter 1 we give a brief introduction on Edge Computing paradigm, we present and motivate our modelling choices for the MEC network planning, in particular for the MEC network topology and the Virtual Machine Mobility Technologies. On the algorithmic side we rely on decomposition techniques, dynamic variable generation and modern paradigms known as Matheuristics: in Chapter 2 we present a brief introduction on those concepts.

After these introductory chapters, we structure the main body of our work in three parts.

Part I

Overview This part revolves around the strategical planning of a Mobile Edge Computing network. We defined it formally as a combinatorial optimization problem, the MEC Network Design Problem (MNDP), that finds simultaneously: (i) an optimal network design, including MEC facilities placement and (ii) an optimal assignment of Access Point (AP) to MEC facilities and routing of the traffic from and to the MEC facilities. From a practical perspective, tactical assignment and routing decisions (ii) are included only to improve the accuracy of strategical design decisions (i); the objective is to minimize a linear combination of total network devices activation costs.

Content In Chapter 3 we focus on the core combinatorial optimization problem of the MNDP, keeping as main target that of devising effective algorithms.

We consider two variants of MNDP, namely *static planning* and *dynamic planning*. Static planning assumes that access points have constant traffic demand, while dynamic planning considers a planning horizon to be split in time slots, in which each access point can have different demand. We consider changes in demand to be given by users moving through access points, and synchronization operations between data in different facilities to be needed accordingly. That is, we explicitly include in our model the orchestration of virtual machines among facilities. Due to the complexity of the resulting formulations, methods from the literature are hard to adapt.

We propose matheuristics solution algorithms for both cases, relying on column generation, rounding, aggregation, and calls to general purpose solvers for performing local search.

In Chapter 4 we tackle the problem from an application point of view, performing data analytics on real-world dataset provided by an industrial partner. We draw best practices on different planning options, performing sensitivity analyses on several model parameters and evaluating different orchestration and virtual machine mobility policies.

State of the Art and Contributions We introduce for the first time at the state of the art a comprehensive strategical framework for the MNDP. The characteristics of MNDP lead to new complex combinatorial optimization problems:

- theoretically, it is strongly NP-Hard, generalizing the traditional uncapacitated facility location problem and its capacitated and single-source variants;
- computationally, it lies on the cutting edge problems currently under investigation in the facility location literature [6]: successful methods are known only when at most two facility levels are considered [7], while in our models a third location level must be included, and routing optimization and latency bounds must be additionally taken into account.
- the dynamic planning variant considers multi-period (or dynamic) facility location and the orchestration of VM hosting facilities; from the telecommunication research point of view, adaptive VM orchestration problems have been studied, either for an off-line or an on-line scheduling [8]; however, in these works, the networks are always assumed to be given.

We show that our approach is efficient enough to deal with real-world network and we

suggest the use of our algorithms as tools for a subsequent predictive analytics step: in our analyses we differentiate between different traffic engineering and performance goals for reference mobile cloud services, analyzing: (i) the use of network facilities resources and (ii) the compliance with users' SLA.

Preliminary results concerning our algorithmic approach were presented in [9]. Main results concerning our analyses were presented in [10] and published in [11].

Part II

Overview This part deals with the tactical side of the MEC network planning, that is, the routing in time of AP traffic to specific MEC facilities on a fixed network structure.

Mobile APs need to be assigned (i.e. their packets need to be routed) to one or more MEC facilities, with a cost in terms of latency for the users they provide connections to. Assignments can be changed over time, but a certain network cost is required for data synchronization operations.

Content In Chapter 5 we present a data-driven MEC management optimization framework for the assignment of APs to MEC facilities, combining: (i) preprocessing and data-mining, (ii) an optimization core component, and (iii) validation by simulation modules.

In the optimization core component, we formally define a new combinatorial problem and we propose ad-hoc mathematical models and exact branch-and-price algorithms that, although exact in nature, performs well also as a matheuristics when combined with early stopping.

Moreover, we exploit the effectiveness of our algorithms in a data analytics framework embedding clustering, optimization and simulation. We show the effectiveness of such a framework using real network data.

State of the Art and Contributions Our data-driven framework is one of the first examples of integration of data analytics for the autonomic management of telecoms network infrastructure [12]. Few works considering such integrations have been presented so far in the telecommunication research area [13, 14], none of which is taking into consideration our application.

The problem of optimally assigning access points to MEC facilities over time turns out to be a novel multi-period variant of a generalized assignment problem from the combinatorial optimization literature: our techniques lie on the edge of recent approaches for the multi-period extension of the Generalized Assignment Problem [15, 16] and the multi-period location problems [17, 18, 19];

We provide initial insights for MEC resources management, evaluating our framework with real-world datasets.

Results were presented in [20] and appeared as technical report [21], that is now submitted for publication.

Part III

Overview In the final part of the thesis, we face a data prediction problem. Accurate algorithms require accurate data, and one of the insights from Part I is that knowledge of users mobility represents a key data for the optimal design of a MEC network. These are however very hard to retrieve due to both availability and privacy issues. In this part we face the problem of estimating user mobility patterns given very aggregated data, still relying on mathematical programming approach.

Content In Chapter 6 we propose a user mobility prediction model, that requires that only mobile access points demands in a time horizon are known, together with the knowledge of some statistical features of mobility distributions: from these aggregated data we try to rebuild fine-grained user trajectories.

We model such a problem as that of finding a suitable set of paths-over-time on a time-dependent graph, belonging to the class of the well known flow over time (or dynamic network flows) [22]. We propose extended mathematical programming formulations and column generation algorithms.

We experiment on both real-world and synthetic datasets.

State of the Art and Contributions To the best of our knowledge, this represent the first mathematical programming approach of such a problem, that is usually modelled with probabilistic and stochastic processes requiring a considerable amount of information [23].

Our approach proves to be efficient enough to tackle real world instances and accurate enough to faithfully estimate mobility on the synthetic datasets for a specific

scenario: the rush-hour time range in a urban-size region.

We found that estimate of users mobility can can be retrieved with adequate accuracy by very aggregated data and simple (a-priori) statistical distributions on human trajectory features, provided suitable mathematical models and combinatorial algorithms are used.

Preliminary results were presented in [24].

Finally, in Chapter 7 we draw a few conclusions, and we trace future perspectives.

Chapter 1

Edge Computing and User Mobility

In this chapter we give a brief introduction on Mobile Edge Computing paradigm and we present and motivate our modelling choices for the MEC network planning, in particular for the MEC network topology and the Virtual Machine Mobility Technologies.

The Edge Computing (EC) paradigm [5, 25, 26, 27, 28], covers a wide range of complementary technologies to provide computational/storage resources in proximity to users, aiming to achieve several advantages:

- reducing latency for users improving their QoE;
- reducing data traffic in Core Networks (CN) decreasing the risk of congestion;
- exploiting location-aware information to provide ad-hoc services, either user-oriented or operator-oriented.

All EC technologies share the same main idea: to provide virtualization facilities throughout the network, from the edge to the edge of the CN. Virtual Machines (VMs) operating in the virtualization facilities are used to offload computation from mobile devices, such as User Equipment (UE) or sensors or Internet-of-Things (IoT) components, or to perform virtualized network operation not related to a specific device. These virtualization facilities can be used in a hierarchical fashion together with centralized cloud facilities: exploiting these latter as last resources if the edge facilities are not available, or to perform delay-tolerant applications.

EC includes both wired and wireless technologies: in the latter case, it is not

bound to a specific wireless link technology: wireless EC facilities can be accessed via WiFi and deployed within a LAN, or accessed via the cellular Access Point and deployed in the Radio Access Network or in the edge of the backhaul cellular network.

EC is still in its infancy and many challenges must be addressed before its implementation to achieve full operation:

- optimal placement of EC facilities;
- computation offloading decision (when to offload and how much of the computation to offload);
- allocation of resources within EC facilities when computation offloading is required;
- standardization of protocols to manage signalling and control operation over the network;
- security and privacy of users data;
- interoperability of different EC technologies;
- mobility among facilities of application, VMs and users' data, either to follow users movements or to balance the load of the network;
- consistency of users' data while using multiple EC technologies;

among others.

1.1 Wireless Edge Computing Technologies Overview

In the following we will present an overview of the wireless EC technologies that play a role in the topics of the thesis.

Cloudlet An early wireless EC concept proposed in 2009 is cloudlet [29]. It is defined as a trusted, resource-rich computer or cluster of computers well-connected to the Internet and available for use by nearby mobile devices. A cloudlet represents a container for virtual machines (VMs): connected users are associated with VMs supporting low-latency application offloading use-cases. Cloudlet concept is expected to be supported by 3-tier hierarchical network provisioning as presented in [30] and [31]. In this hierarchy the cloudlet is the primal resource for the enhancement of the mo-

bile device capabilities, while a remote cloud is used as last available resource, or for delay-tolerant resource-intensive applications.

In its first presentation, a cloudlet was expected to be accessed via a single hop WiFi connection. This can represent a disadvantage since mobile UEs have to switch between WiFi and mobile cellular network while offloading services [32].

Benefits of cloudlet usage on users' QoE are presented in [33, 34, 35] where authors compare performances of different types of applications on different layers of the 3-tier hierarchy. In [33] the authors show that application placement can significantly impact performance and user experience: moving applications closer to the users is strongly beneficial. The authors of [34] question, by quantitative experimental results, benefits from consolidating computing resources in large data centers when strict latency constraints are required. Considering multi-hop WiFi networks, in [35] the authors show that the cloudlet-based approach always outperforms the cloud-based one when no more than two wireless hops are used to transfer data, and that up to a maximum of four hops the cloudlet-based approach is the best one for most of the instances. A further survey on research on cloudlet based mobile computing is available in [36].

Mobile Ad-Hoc Cloud In this case the computation is performed by the combination of several nearby UEs which compose an ad-hoc network. In this case a further issues has to be addressed: UEs has to be motivated to allow access to their computational power, hence consuming battery and limiting performance for user own applications. Further efforts on synchronization among UEs and on security and privacy protection are necessary. Surveys on mobile ad-hoc cloud can be found in [5, 37].

Fog Computing It is defined as "*A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum*" [38]. Differently from other EC paradigm, Fog Computing involves dynamic pooling of resources and data sources of possibly multiple devices that reside in the so-called *cloud-to-thing continuum*, that is, anywhere between the cloud and the end-point device. It is supposed to work over wireless and wireline networks and also inside these networks, and to support multiple industries application domains. Fog architecture aims to offer several advantages:

- security: additional security to ensure safe, trusted transactions;
- cognition: awareness of client-centric objectives to enable autonomy;

- agility: rapid innovation and affordable scaling under a common infrastructure;
- latency: real-time processing and cyber-physical system control;
- efficiency: dynamic pooling of local unused resources from participating end-user devices.

Mobile Edge Computing In the case where EC is considered in the cellular network, the term Mobile Edge Computing (MEC) is used. The MEC concept was introduced by the European Telecommunications Standardization Institute (ETSI) in [3, 4], that now uses the term Multi-Access Edge Computing, and it aims to provide IT and cloud-computing capabilities within the mobile access and backhaul network, in close proximity to mobile subscriber. The standardization focus of ETSI is on the definition of an open environment to allow seamless integration of applications across MEC platform, and in particular in the definition of an interface of the MEC framework that actual implementations should expose. Its goal is to allow new revenue streams by exposing the edge of Radio Access Network of Mobile Service Provider to authorized third-parties for application hosting.

No constraints on deployment are defined by the standardization bodies, while several proposals are presented in literature [5, 27], differentiating by: (i) the location of the facilities that perform the computation (MEC facilities in the remainder); (ii) the way that the system is managed and controlled and; (iii) the location of the controller facilities. Two main ideas for the placement of MEC facilities have been proposed:

- deploy only in mobile access points;
- hierarchical deployment over the access and backhaul mobile network, from the edge of the CN to the access point.

In the same way, two main ideas for the control/signalling of the MEC network have been presented:

- a fully centralized control: either performed in the highest level of MEC facilities in the hierarchy or in a cluster of MEC facilities in access points;
- a distributed control.

1.2 MEC Network Topology

In this thesis we consider a MEC Network topology that contains a hierarchical deployment of MEC facilities from the edge of the CN to the access points, abstracting from any particular implementation already in the literature.

Accordingly to ETSI [3, 4], the distribution of computing resources into mobile access networks should be carefully designed to take into account infrastructure properties. Mobile access networks could be any form of wireless access network disposing of a backhauling wireline infrastructure through which MEC facilities can be interconnected. Following the guidelines in [39, 40, 41, 42], a broadband access and backhauling network, such as a cellular network, can be modeled as a two-level hierarchical network: **access points** on the field are connected to **aggregation nodes**, which are then connected to **core nodes**, as depicted in Figure 1.1. The APs could be WiFi only, cellular only, or a mix of these common mobile access technologies. MEC facilities can reasonably be placed at either field, aggregation or core level, with connections between an AP and its MEC facility potentially crossing twice each level. Placing a MEC facility at a location could mean turning on already installed servers, and not only physically installing new machines. Similarly, changing AP to MEC facility assignments would in practice correspond to a re-routing of virtual links over the transport network infrastructure, and not physically changing the interconnection.

Various physical interconnection network topologies between APs, aggregation nodes and core nodes are commonly adopted: tree, ring or mesh topologies, as well as intermediate hybrid topologies. Moreover, with the emergence of 4G, there is a trend to further mesh backhauling nodes. A variety of network protocol architectures are typically adopted, from circuit-switched networks to carrier-grade packet-switched networks. The common denominator of such architectures is the ability to create a virtual topology of links directly interconnecting pairs of nodes at a same level with a guaranteed tunnel capacity. Nowadays, with the convergence towards packet-switching carrier-grade solutions at the expense of legacy circuit-switched approaches, bit-rates for pseudo-cables links is set to giga-Ethernet granularities (typically 1 or 10 Gbps).

In this framework, we believe to be appropriate modelling the MEC network as a superposition of stars of virtual links for the interconnection of aggregation nodes to APs and for the interconnection of core nodes to aggregation nodes, even if nodes can have no physical direct connection. Under the same virtual link provisioning trend, core nodes can be considered as interconnected to each other by a

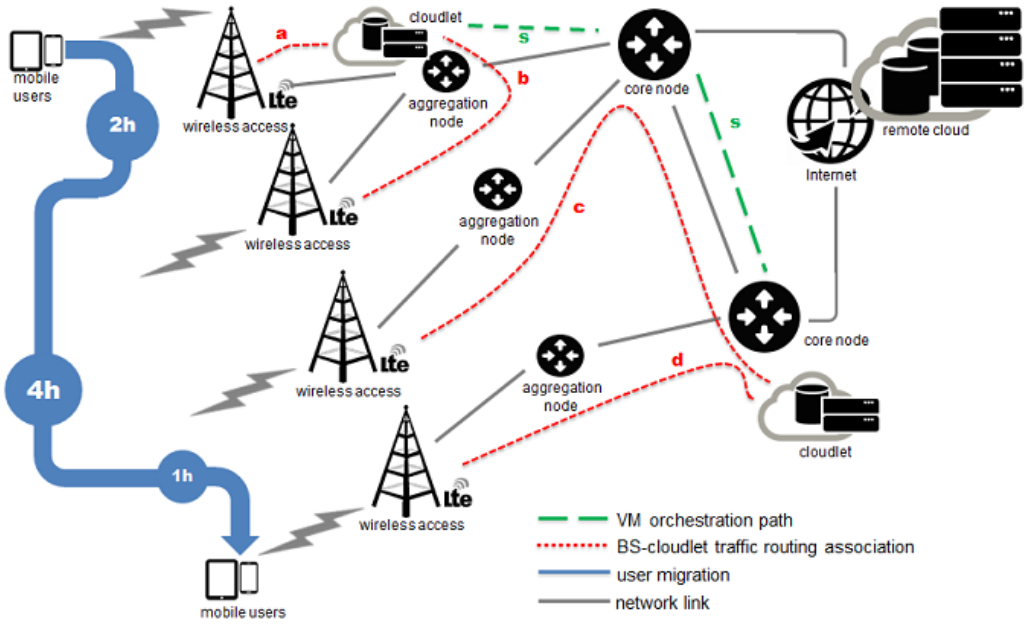


Figure 1.1: Example of a simplified MEC Network.

full mesh topology of virtual links, as depicted in Figure 1.1. The choice between single source or fractional assignment models is often crucial in network optimization. In our application, single source assignment is more pertinent. As far as we know, partitioning of traffic from one AP to multiple aggregation nodes, and from one aggregation node to multiple core nodes is not the dominating current practice in backhauling networks; still, such features would not change significantly the nature of our models and algorithm described in the next two sections. It is worth noting that the decisions of associating APs to aggregation nodes and placing aggregation nodes can be fully compatible with the current trend of dynamically reprogramming the cellular backhauling network [43]. Likewise, another customization could correspond to the routing re-optimization for a given MEC facility placement. Moreover, those decisions can also realistically embed association and placement functions in cloud-based Evolved Packet Core architectures [44].

1.3 Virtual Machine Mobility Technologies

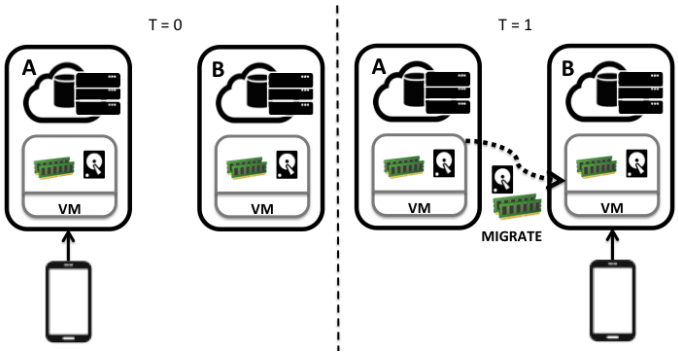
One of the features that the MEC has to address is the management of the mobility of MEC applications, either to follow users movement or to balance the load of the

network. In a general case, this means the management of VMs mobility among MEC facilities. In this thesis we consider user-related VMs, which require to maintain the data specific for the users, rather than virtual network functions. In particular we consider three VM mobility technologies at the state of the art:

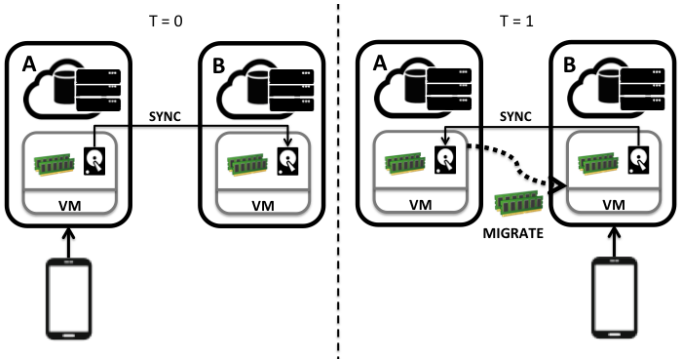
- *VM bulk migration* [45]: consists in migrating the whole VM stack including disk and memory, stopping the VM for a long period to transfer it.
- *VM live migration* [46, 47, 48]: stops the VM only for a small amount of time required to transfer the most recently used memory, not requiring an entire one-shot disk transfer, but a permanent disk storage synchronization among source and destination locations.
- *VM replication* [49]: consists in a permanent synchronization of both disk storage and memory among source and destination locations, not requiring the point transfer neither of the disk nor of the most recently used memory.

An example of these technologies is presented in Figure 1.2: in time $T = 0$ a UE is connected to a VM in MEC facility A and in time $T = 1$ it moves to a region served by MEC facility B. Following VM Bulk Migration (Figure 1.2a) no action is required in time 0, and only when the UE moves to the region served by MEC facility B the full VM stack, including disk and memory, is migrated among the facilities. Following VM Live Migration (Figure 1.2b), in time 0 the disk of VM in facility A is synchronized with the disk of the VM in facility B; when the UE moves to MEC facility B at time 1, the sole migration of the memory is required and the direction of disk synchronization is reversed from B to A. Finally, considering VM replication (Figure 1.2c), in time 0 both the disk and the memory of VM in facility A are synchronized with disk and memory of VM in facility B; when the UE moves to facility B, no further migration is required and the direction of synchronizations is reversed.

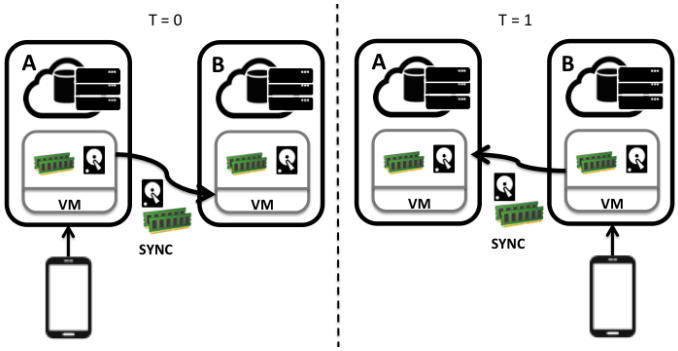
We assume VM orchestrations to be performed in a Cloud Stack platform in a centralized way. Given that the main purpose of our models is the medium-term planning of the mobile edge cloud network, the inclusion of VM orchestration has the aim to provide a correct dimensioning of the network. At the same time, examples of VM orchestrator are already implemented in Openstack platform [50, 51]. Therefore an actual implementation of such a system is out of scope of this work.



(a) VM Bulk Migration



(b) VM Live Migration



(c) VM Replication

Figure 1.2: Examples of VM Mobility Technologies

Chapter 2

Column Generation and Matheuristics

As outlined in the Introduction, in this thesis we deal with new strategical and tactical MEC Network planning problems, presented in Chapters 3 and 5, that can be modelled as NP-Hard combinatorial optimization problems. The numerical resolution of these problems yield several issues related to both their complexity and the size of the instances to be optimized.

To tackle these problems we follow the recent trend of hybridizing mathematical programming techniques with metaheuristics, obtaining so-called *hybrid-metaheuristics* or *matheuristics* [52, 53]. In particular, we exploit the structure of our problems, which are suited to be decomposed in different (and, in our case, easy) subproblems. We formalize them with formulations resulting from decomposition techniques, that in our case have led to an increase of the set of variables. To algorithmically tackle such formulations, we employ dynamic generation of variables (also known as Column Generation (CG)) techniques. By combining metaheuristics and Column Generation we obtain both *good quality solutions* in a reasonable time, and *quality guarantees* on them. In the case of the tactical planning problem presented in Chapter 5 we could also embed them in a branch-and-bound framework, thereby obtaining effective *exact* algorithms.

In this chapter we present a brief introduction on the concept of decomposition and Column Generation, and a general outline on how we used them in the design of our matheuristics.

2.1 Dantzig-Wolfe Decomposition

The problems we face in this thesis share a similar structure: they are suited to be decomposed in different subproblems. We exploit this characteristic to formalize them with decomposed mathematical programming formulations. In particular, these formulations consider elements from a set of combinatorial objects P, to each of which a variable is associated. To give an example, P can encode the set of all possible paths in a network. In principle, due to their flexibility, such formulations can be the direct result of a modelling step. More often, however, they are results of reformulation techniques such as Dantzig-Wolfe Decomposition (DW) [54].

To ease the exposition, we report an example inspired by [55], by considering the Capacitated Shortest Path Problem (CSPP). Let $G(N, A)$ be a directed graph, having for every arc $(i, j) \in A$ a non-negative cost of traversal $c_{i,j}$ and a traversal time $t_{i,j}$. We aim at finding the minimum cost path in G from a node $s \in N$ to a node $t \in N$, with a constraint on the total amount of traversal time T.

We can formulate the CSPP as an integer mathematical programming model: let us introduce a binary variable $x_{i,j}$ for every arc in G, taking value 1 if the arc is chosen for the solution path, 0 otherwise. The CSPP can be modeled as follows:

$$\min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{2.1}$$

$$\text{s.t.} \sum_{j\in N:(s,j)\in A} x_{s,j} = 1 \tag{2.2}$$

$$\sum_{j\in N:(i,j)\in A} x_{i,j} - \sum_{j\in N:(j,i)\in A} x_{j,i} = 0 \qquad \forall i \in N \setminus \{s,t\} \tag{2.3}$$

$$\sum_{j\in N:(j,t)\in A} x_{j,t} = 1 \tag{2.4}$$

$$\sum_{(i,j)\in A} t_{i,j}x_{i,j} \leq T \tag{2.5}$$

$$x_{i,j} \in \{0,1\} \tag{2.6}$$

where (2.1) is the objective function for the minimum cost path; (2.2) states that one unit of flow must exit the source s; (2.4) states that one unit of flow must enter the target node t; (2.3) is the flow conservation constraints: for every intermediate node in the path if a unit of flow enters then a unit of flow must exit; (2.5) is the resource

constraints on the total amount of time; finally (2.6) limits the value of variables $x_{i,j}$.

Formulation (2.1)–(2.6) represents a so-called *natural* model for the CSPP, where natural has the meaning of directly mapping single elements of the original graph to variables and constraints. In fact, in the CSPP case, one variable is included for each arc of the graph. Natural formulations are known to produce often poor bounds. To improve it a few options are discussed in the literature to obtain *extended* formulations. One of them is to replace each variable with a set of variables, each encoding a particular resource usage level. This is the case, for instance, of the approach discussed in [56]. Another possible approach is to perform a partial convexification of the feasibility region by exploiting an inner-representation for a subset of the constraints, thereby obtaining extended formulations with one variable for each extreme point of the convexified region: it is the case of Dantwig-Wolfe Decomposition.

We can observe that (2.5) is a *complicating* constraint: without it the problem reduces to a shortest path problem. While in the general case shortest path problems with additional constraints are NP-Hard, some particular cases show simple shortest path structure, and are therefore polynomially solvable with textbook algorithms, some others are still polynomially solvable although such a result is not trivial, some others remain NP-Hard [57]. In this case the shortest path problem defined by (2.1)-(2.4), (2.6) has non-negative arc costs $c_{i,j}$ and no additional constraints, and hence can be solved to optimality in polynomial time. We can decompose CSPP following DW principles relaxing constraint (2.5): each solution satisfying the remaining constraints (2.2)-(2.4), (2.6) encode a path in G connecting source s and target t. In details, let us rewrite (2.1)–(2.6) as:

$$\min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{2.7}$$

$$\text{s.t.} \sum_{(i,j)\in A} t_{i,j}x_{i,j} \leq T \tag{2.8}$$

$$(x_{i,j}) \in \Omega = \{(x_{i,j}) \mid (2.2), (2.3), (2.4)\} \tag{2.9}$$

$$x_{i,j} \in \{0,1\} \tag{2.10}$$

Ω represents the feasible region with respect to constraints (2.2)–(2.4); let use replace Ω with its convex hull $\texttt{conv}(\Omega)$. The relaxation of the integrality conditions (2.10) leads to a final bound of (2.7)–(2.9) that is not weaker than the Linear Programming (LP) relaxation of the original problem (2.1)–(2.6).

Let P be the set of extreme points of $\texttt{conv}(\Omega)$. P has a particular combinatorial interpretation: it is the set of all paths connecting nodes s and t in graph G, and for each path $p \in P$ let $\hat{x}_{p,i,j} \in \{0,1\}$ be the coefficient encoding the path, that is, $\hat{x}_{p,i,j}$ equals 1 if the arc $(i,j) \in A$ belongs to path p, 0 otherwise. Each element of Ω can be represented as a linear combination of points in P and hence we can replace (2.9) with the following:

$$x_{i,j} = \sum_{p \in P} \hat{x}_{p,i,j} \lambda_p \tag{2.11}$$

with $\lambda_p \geq 0$. Replacing every variable $x_{i,j}$ with its linear convex combination (2.11) we obtain the following formulation of the CSPP:

$$\min \sum_{p \in P} \sum_{(i,j) \in A} c_{i,j} \hat{x}_{p,i,j} \lambda_p \tag{2.12}$$

$$\text{s.t.} \sum_{p \in P} \sum_{(i,j) \in A} t_{i,j} \hat{x}_{p,i,j} \lambda_p \leq T \tag{2.13}$$

$$\sum_{p \in P} \lambda_p = 1 \tag{2.14}$$

$$\lambda_p \geq 0 \qquad \forall p \in P \tag{2.15}$$

where (2.12) is the objective function that aims to find the minimum cost path; (2.13) states that the overall traversal time of a path can not exceed threshold T; (2.14) states that one path has to be chosen among the set of all paths P, and, finally, (2.15) limits the value of variables λ.

Column Generation Let us refer to the formulation (2.12)–(2.15) as the Master Problem (MP). The MP has to deal with the combinatorial size of set P: even solving its continuous LP relaxation would be computationally impossible on networks of realistic size. In order to solve this LP relaxation, however, we can exploit *column generation* techniques (CG).

CG is an iterative process: the full set of points P and the corresponding set of variables λ_p is replaced by a subset $\bar{P}$ to form a Restricted Master Problem (RMP); the LP relaxation of RMP is solved and dual information are used to search for elements of P not included in $\bar{P}$, whose corresponding variables have negative reduced costs. These are in turn candidates for improving the LP solution. This search is called *pricing*. If such negative cost variables are found, they are added to $\bar{P}$ and the process iterates. Otherwise CG stops: since no negative reduced cost

variable can be found, the optimal solution of the RMP LP relaxation is optimal for the LP relaxation of the MP as well.

In our CSPP, CG has therefore to find through pricing new negative reduced cost variables λ_p, which represent a path $p \in P$ from source s to target t. Exploiting duality theory [58], given dual variable π corresponding to constraint (2.13) and dual variable θ corresponding to constraint (2.14), the reduced cost of variable λ_p is given by $\sum_{(i,j)\in A} c_{i,j}\hat{x}_{p,i,j} - \pi \sum_{(i,j)\in A} t_{i,j}\hat{x}_{p,i,j} - \theta$. The corresponding pricing problem is, as expected, a polynomial time complexity minimum cost shortest path problem.

At the end of CG process, a solution for the LP relaxation of the extended formulation of the problem is given, with two further benefits: (i) the final RMP solution is a valid lower bound for the original problem; it can never be worse than the LP relaxation value of the original compact formulation and it is often much better; (ii) the CG retrieves a set of promising combinatorial entities (i.e. the elements of $\bar{P}$), which can be used as a guide to build upper bounds, that are good feasible solutions. These valuable outcomes can be used within a matheuristic framework to tackle the original problem.

Quality of relaxation and pricing complexity In the presented reformulated model (2.12)–(2.15) the corresponding pricing problem possesses the *integrality property*. That is, an optimal solution for its LP relaxation always exists in which the variables take integer values. When this condition is true, the final bound retrieved by the CG is equal to that given by the LP relaxation of the original problem (2.1)–(2.6). Even in this case CG might still be useful if the size of the original problem is too large to be tackled by the available computational resources, while pricing subproblems are easy to solve at optimality.

Indeed, keeping our running example, a better bound could be retrieved by considering a different decomposition of the problem whose pricer does not posses integrality property: in this case the exact resolution of the pricer may not by straightforward, however the resulting final bound is usually tighter than the LP relaxation of the original problem.

For example, we can decompose CSPP following DW principles relaxing flow

conservation constraints (2.3). Let us rewrite (2.1)–(2.6) as:

$$\min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{2.16}$$

$$\text{s.t.} \sum_{j\in N:(i,j)\in A} x_{i,j} - \sum_{j\in N:(j,i)\in A} x_{j,i} = 0 \qquad \forall i \in N \setminus \{s,t\} \tag{2.17}$$

$$(x_{i,j}) \in \bar{\Omega} = \{(x_{i,j}) \mid (2.2), (2.4), (2.5)\} \tag{2.18}$$

$$x_{i,j} \in \{0,1\} \tag{2.19}$$

Still, we replace $\bar{\Omega}$ with its convex hull $\texttt{conv}(\bar{\Omega})$, and we consider the set of the extreme points $\bar{P}$ of this latter. $\bar{P}$ in this case has no particular combinatorial interpretation; for every point $p \in \bar{P}$ let $\hat{x}_{p,i,j} \in \{0,1\}$ be the coefficient encoding the membership of arc $(i,j) \in A$ to the element p. We can replace every point in Ω as a convex combination of the extreme points in $\bar{P}$, replacing (2.18) with:

$$x_{i,j} = \sum_{p\in\bar{P}} \hat{x}_{p,i,j}\lambda_p \tag{2.20}$$

with $\lambda_p \geq 0$. We can replace (2.18) with (2.20) and relax integrality conditions (2.19) obtaining the following master problem:

$$\min \sum_{p\in\bar{P}} \sum_{(i,j)\in A} c_{i,j}\hat{x}_{p,i,j}\lambda_p \tag{2.21}$$

$$\text{s.t.} \sum_{p\in\bar{P}} \Big(\sum_{j\in N:(i,j)\in A} \hat{x}_{p,i,j} - \sum_{j\in N:(j,i)\in A} \hat{x}_{p,j,i} \Big) = 0 \qquad \forall i \in N \setminus \{s,t\} \tag{2.22}$$

$$\sum_{p\in\bar{P}} \lambda_p = 1 \tag{2.23}$$

$$\lambda_p \geq 0 \qquad \forall p \in \bar{P} \tag{2.24}$$

The pricing problem to find new elements in $\bar{P}$ with negative reduced cost can be modelled considering dual variables ρ_i related to constraints (2.22) and the dual variable θ from constraint (2.23). The corresponding model is the following:

$$\min -\theta + \sum_{(i,j)\in A} c_{i,j}x_{i,j} - \sum_{i\in N\setminus\{s,t\}} \rho_i \sum_{j\in N} x_{i,j} + \sum_{i\in N\setminus\{s,t\}} \rho_i \sum_{j\in N} x_{j,i} \tag{2.25}$$

$$\text{s.t. } (2.2), (2.4), (2.5)$$

$$x_{i,j} \in \{0,1\}$$

This pricing problem does not possess the integrality property. Its exact resolution may require computationally demanding algorithms to run in every CG iteration. However the final bound retrieved by this reformulation (2.21)–(2.24) is usually tighter than the bound retrieved by the master model (2.12)–(2.15).

2.2 Matheuristics

A common approach to complex combinatorial optimization problem is to trade quality guarantees for computing complexity. Indeed, practical problems often require to employ either heuristics or their generalization known as *metaheuristic*. The concept of metaheuristics is mainly qualitative. In general metaheuristics are seen as heuristics guiding other heuristics. The following tentative characterisation of metaheuristic is given in [59]:

> " *[...] We outline fundamental properties which characterize metaheuristics:*
> - *Metaheuristics are strategies that "guide" the search process.*
> - *The goal is to efficiently explore the search space in order to find (near-)optimal solutions.*
> - *Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes. Metaheuristic algorithms are approximate and usually non-deterministic.*
> - *They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.*
> - *The basic concepts of metaheuristics permit an abstract level description.*
> - *Metaheuristics are not problem-specific.*
> - *Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.*
> - *More advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.*
>
> *[...]* "

In short, metaheuristic refers to heuristic algorithm that are not specifically expressed for a particular problem, but rather to a wide class of problems. The main reason to use such algorithms is their ease of customization for a wide range of problems to quickly get feasible solutions. On the other hand, they usually lack of guarantee about the quality of the given solution.

The modern approach to the design of metaheuristics is hybrid in nature [52, 53]: the rationale behind the hybridization of different techniques is to exploit the complementary character of different optimization strategies.

Heuristic algorithms may include both components which are specializations of metaheuristics, and others in which subproblems are formulated as Mathematical Programs and solved with generic techniques; these particular combinations are indicated in the literature as *matheuristics*. Matheuristics can grant several benefits. For instance, mathematical programming techniques can be used to easily model and tackle heavily constrained subproblems, while traditional techniques can be more suited to quickly improve them over a small search space. Furthermore, while metaheuristic focus on finding primal solutions, generally disregarding dual information, the use of Mathematical Programming bounds in matheuristics can produce quality guarantees, or at least estimates of quality, on the solutions produced. More involved combinations are of course possible: dual information can be used to reduce the search space, or even for guiding intensification.

Column Generation based Matheuristics The best use of matheuristics has still to be understood: there is not a direct guidance on which algorithm to use given a specific problem; the actual implementation of a problem-specific algorithm can be a time-consuming task and a broad knowledge is required in algorithms, mathematics, statistics and advanced programming. Moreover, changes in problem specification could make the algorithm ineffective and the resulting software could be difficult to maintain and tune.

Inspired by approaches like [60, 61], we managed to cope with these drawbacks by embedding extended formulations and column generation in a matheuristic framework.

In Chapters 3 and 5 we present two matheuristics which exploit CG in different ways. In both cases we deal with a Mixed Integer Linear Programming (MILP) problem and we want to restore integrality from the CG fractional solution.

In Chapter 3 CG is used together with a simple rounding in an interleaved process, presented in Algorithm 1, which is in turn a component of a larger matheuristic. We execute CG on our problem P and get a valid lower bound and valuable columns x; in order to restore integrality on the fractional solution we round columns in an iterative process; at each iteration a column x is chosen to be rounded and fixed to a value $\bar{x}$; the rounding is propagated to fix further variables that would lead to infeasibility; CG is executed on the problem with fixed variables $P(x, \bar{x})$ to generate

Algorithm 1 CG and Rounding Matheuristic Framework

solve CG of problem $P(x)$ and get valid lower bound
repeat
  round a subsets (even just one) fractional variables x and fix them to $\bar{x}$
  solve CG of problem $P(x, \bar{x})$
until feasible solution of P is found $\vee$ no more variable to fix
a valid lower bound is given by the first CG process
a valid upper bound is given at the end of the rounding iterations

new valuable columns; the process iterates until either a feasible solution for P is found or no more variables can be fixed. There is no guarantee on the success of this process, and a valid upper bound for the problem is given at the end of the rounding iterations.

In Chapter 5 we devise a different matheuristic framework exploiting CG, the structure of which is presented in Algorithm 2. Such a matheuristic is further embedded as a component in an exact algorithm. While CG iterates, the fractional solution of the LP relaxation of the RMP is fed to a secondary heuristic algorithm that tries to build a feasible solution. At the end of CG, the last value or the RMP LP relaxation is a valid lower bound for the problem, while the best solution found by the secondary heuristic is a valid upper bound.

In general, more involved options may be conceived. For instance, Algorithm 2 can be nested in Algorithm 1, or the rounding of Algorithm 1 can be used as H heuristic of Algorithm 2.

Algorithm 2 CG based Matheuristic Framework

given heuristic H(σ)
repeat
  solve LP of restricted master problem RMP, get solution σ and duals π
  use H(σ) to build feasible solution
  solve pricer subproblem SP(π) to get minimum reduced cost column c
  if $c < 0$ **then**
    add column to RMP
  else
    exit loop
  end if
until terminal condition
best feasible solution found with H(σ) is valid upper bound for the problem
if SP(π) solved at optimality and no more negative reduced costs columns found **then**
  last solution of RMP is valid lower bound for the problem
end if

Part I

Strategical MEC Network Planning

Chapter 3

Optimization Algorithms for MEC Network Design

In this chapter we focus on the core combinatorial optimization problem of strategically planning a MEC network, taking into account both infrastructural properties and expected network usage. Our main target is devising effective algorithms, which are meant, in turn, to be used as tools for a subsequent predictive analytics step.

We here give a formal mathematical formulation of the MEC Network Design Problem (MNDP). Our model finds simultaneously: (i) an optimal network design, including MEC facilities placement and (ii) an optimal assignment of APs to MEC facilities and routing of the traffic from and to the MEC facilities. The objective is to minimize a linear combination of overall installation costs. From a practical perspective, tactical assignment and routing decisions (ii) are included only to improve the accuracy of strategical design decisions (i); in fact more accurate data analytics models are introduced in Chapter 4 to refine assignment and routing after a design is fixed.

We propose two variants for the MNDP:

- *Static Planning*: network status is considered static in time; neither user mobility nor virtual machine mobility are taken into account when planning MEC facility placement, and associations of APs to MEC facility.
- *Dynamic Planning*: variations in the network load during the planning time horizon are taken into account together with user mobility. Adaptive VMs orchestration is included in a generalized way to consider three different technologies: VM bulk migrations, VM live migrations and VM replications.

Our problem turns out to be hard from both a theoretical and computational point of view. We present mathematical programming based matheuristics for the two variants of MNDP, combining column generation, iterative rounding, very large scale neighborhood local search and problem reduction. We undertake an extensive experimental campaign on benchmarks drawn from the literature. Our results indicate that our algorithms are effective in producing optimized MEC networks with limited computing resources.

Preliminary results were presented in [9].

3.1 Introduction

The MNDP is the joint problem of the design of a two-layer hierarchical network together with the location of further virtualization facilities within this network. We consider a solution to be feasible if: traffic demands of APs are satisfied by MEC facilities, network limited resources are not exceeded, the resulting network follows a set of topological rules and users' Service Level Agreement (SLA) are respected.

We propose two variants of the MDNP: the *Static Planning* and the *Dynamic Planning* variants.

Static Planning MNDP

In the Static Planning varian of the MNDP the network status is considered static in time; neither user mobility nor virtual machine mobility are taken into account when planning MEC facility placement, and assigning of APs to MEC facility. In Figure 3.1 we present an example of its decision process:

Fig. 3.1a the network on which to design a MEC network is composed by four APs (`A1` to `A4`) and by candidates location for aggregation nodes (dashed circles), core nodes (dashed squares) and MEC facilities (light-colored clouds); dashed lines represent feasible links connecting candidate locations; APs `A2` and `A3` have associated fixed demand (identified by UEs `U1` `U2` and `U3`)

Fig. 3.1b a possible solution is given: candidates locations `N1` and `N2` are activated as aggregation nodes (colored circles), `C1` and `C2` as core nodes (colored squares) and `K1` and `K2` as MEC facilities (black clouds). Activated nodes are connected by feasible links (black lines) reflecting topological rules defined in Chapter 1.2, while APs are assigned to MEC facilities (red dashed lines) to fulfil their

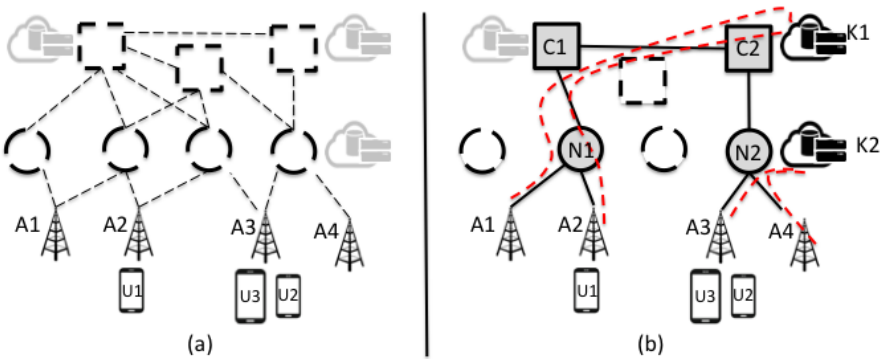


Figure 3.1: Static Planning Example

demands. Users' SLA is modelled with constraints on AP-MEC facilities assignment paths.

Dynamic Planning MNDP

In many realistic MEC scenario, AP demands change over time; therefore in the Dynamic Plannign case of the MNDP, variations in the network load during the planning time horizon are taken into account together with user mobility.

As users move during the planning horizon, they connect to different APs, changing the network load distribution, with the necessity to re-plan the network to re-balance the system. Moreover as they move they may distance themselves from their VM, worsening their QoEs and violating their SLA.

We recur to time discretization, creating multi-period variants of our models in which the assignment of APs to MEC facilites can change over time, while aggregation, core and MEC facility locations never change.

In order to re-balance the system and to enforce SLA we introduce adaptive VMs orchestration in a generalized way to consider three different technologies: VM bulk migrations, VM live migrations and VM replications.

In Figure 3.2 an example of the Dynamic Planning variant is presented:

Fig. 3.2a the network on which the MEC network has to be build is composed by two APs (`A2` and `A3`) and by candidates nodes for aggregation nodes (dashed circles), core nodes (dashed squares) and MEC facilities (light-colored clouds). AP `A2` has associated demand of an UE `U1`, that moves in a discretized time planning

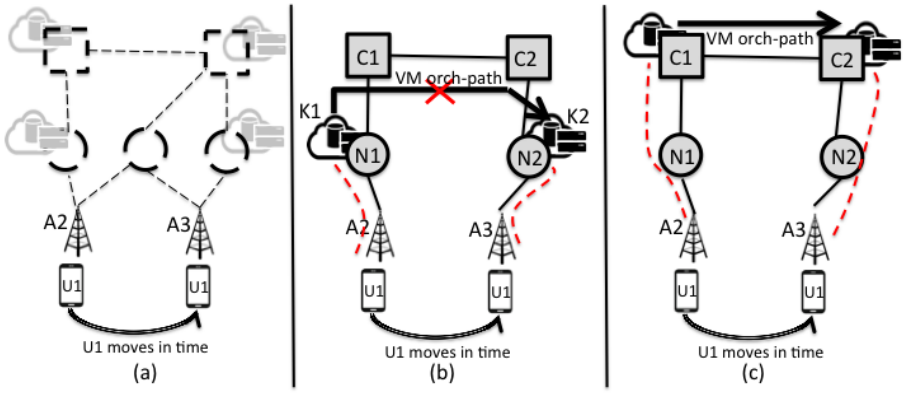


Figure 3.2: Dynamic Planning Example - VM Orchestration Path

horizon to reach AP `A3`;

Fig. 3.2b a possible solution is presented: candidates location `N1` and `N2` are activated as aggregation nodes, `C1` and `C2` as core nodes and `K1` and `K2` as MEC facilities; APs are assigned to MEC facilities (`A2` to `K1` and `A3` to `K3`, resp.; red dashed lines). However, as UE `U1` is moving in time, we need to take into account an orchestration strategy for its VM among the facilities to which he will connect to within the time horizon: in this case `U1` is connected first to `K1` and then to `K2`, hence a VM orchestration path is established between the two facilities (thick black arrow). On this path we pose constraints to model users' SLA: long orchestration paths lead to worsening in users perceived latency when VM need to be synchronized. In this figure we assume the orchestration path to be infeasible: a new solution need to be devised;

Fig. 3.2c a new possible solution is presented: MEC facilities are activated in core nodes `C1` and `C2`, rather than in aggregation nodes as in the previous solution. While AP-MEC facilities assignment paths are longer, the VM orchestration paths is now feasible with respect to users' SLA constraints.

Moreover, in the Dynamic Planning variant we also allow AP-MEC facility assignments to change in time, in order to rebalance the network load. Finding a good solution for the Dynamic Planning variant of the MNDP is not trivial and counterintuitive decision could be the most valuable. In Figure 3.3 a second example of the Dynamic Planning variant is presented:

Fig. 3.3a a solution to the Dynamic Planning MDNP is presented: while APs `A1` and `A2` are assigned to MEC facility `K1`, APs `A3` and `A4` are assigned to `K2`. AP `A2` has

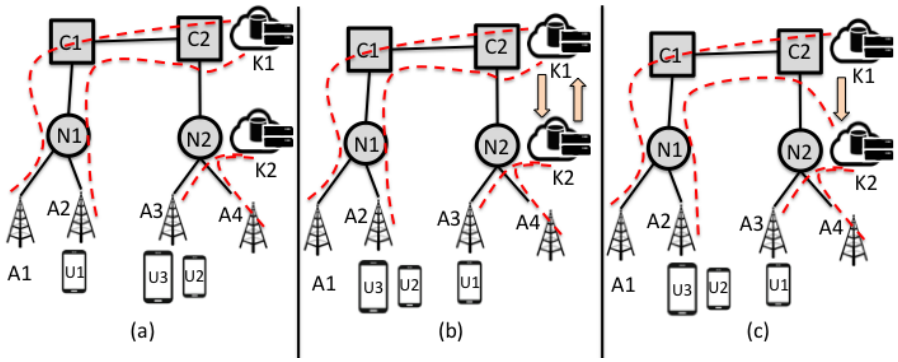


Figure 3.3: Dynamic Planning Example

an associated UE U1, while AP A3 has two connected UEs U2 and U3; this is a starting point of our network;

Fig. 3.3b in a successive time, U2 and U3 moves to the AP A2 that is served by a different MEC facility; at the same time U1 takes the contrary route to A3 and has to change MEC facility too. We need two VM orchestration paths from K1 to K2, synchronizing the VM of U1 from its former facility to its new facility, and from K2 to K1, to synchronize the VMs of U2 and U3. This choice could overload the link between aggregation node N2 and core node C2 that is loaded with all the orchestration traffic: a better solution might exists;

Fig. 3.3c in this solution we choose to change the assignment of AP A2 from facility K1 to facility K2 when U2 and U3 moves to this AP; this is a counterintuitive decision, as we are assigning A2 to a farther facility; however, if this solution is feasible with respect to the users SLA and facility capacity, it allows U2 and U3 to be assigned to their former facility, hence without the need to orchestrate their VMs among facilities. This solution requires to orchestrate only the VM of U1 from its former facility K1 to its new facility K2.

Literature Review

These problem characteristics lead to new complex combinatorial optimization problems. In fact, our models generalize both the Single-Source Capacitated Facility Location (SSCFL) Problem and two-level facility location (2FL) models [7, 62, 63]. Indeed, we combine features of both SSCFL and 2FL: the problem of optimally locating aggregation and core nodes can be seen as a 2FL, while the problem of optimally locating capacitated MEC facilities and assigning AP demands to them over

the network can be seen as a generalization of SSCFL.

Both SSCFLP and 2FL are NP-Hard; however, while state of the art methods for SSCFL can successfully tackle large instances [64], optimizing 2FL is much more involved: capacitated version of 2FL have been tackled both heuristically and with exact algorithm in several variants arising from application needs. From the heuristic point of view, authors of [6] presented a ILP based matheuristic for the 2FL with single source constraints at both levels and dimensioning of the facilities to solve instances with up to 200 customers and 50 potential sites of facilities; a similar version of the problem but considering multiple capacity values for every facility was faced by authors of [65] with a Lagrangean relaxation based heuristic to solve instances with up to 400 customers and 100 facilities. Few exact algorithm have been presented, dealing with small size networks: authors of [66] deal with a two-echelon facility location problem in which each customer is served by an uncapacitated facility in the first echelon and by a capacitated facility in the second echelon and present a Lagrangean relaxation based branch-and-bound providing optimal solution for small networks of 20 and 50 customers and less than 10 facilities. Even fewer works dealing with more than two levels have been presented, providing approximation algorithm for arbitrary k-level [7].

However our problem is even more general, considering that the MEC location decisions build a third level, standing on top of the optimization of aggregation and core locations, and that routing optimization and latency bounds have to be considered.

In the dynamic planning case, multi-period (or dynamic) facility location problems tend to be large and thus more difficult to tackle, even for instances of moderate size [63, 67]. Managing a dynamic planning is even more involved: since demand is changing due to users moving between different APs, when a particular user switches from APs connected to different MEC facilities, the VM hosting her services must be orchestrated. Several adaptive VM orchestration problems have been studied, either for an off-line or an on-line scheduling [8]; however, in these works, the networks are always assumed to be given.

In the following, we first introduce the modeling of the network topological rules presented in Chapter 1.2 (in Section 3.2) that represents the base of our models; then we add routing aspects (in Section 3.3), thereby completing them for the Static Planning variant, of which we present our matheuristic in Section 3.4. In Section 3.5 we discuss how this modelling extends to the Dynamic Planning variant, and

its variant of matheuristic in Section 3.6. The experimental analysis is presented in Section 3.7. Finally in Section 3.8 conclusions of the chapter are presented.

3.2 Network Design Formulation

Input (Problem Data). We assume that a set of suitable locations has been identified for hosting network facilities as described in Chapter 1.2. Formally, let B be the set of AP locations. Let I, J and K be the set of candidates sites where aggregation, core nodes and MEC facility can be installed, respectively. We assume that a MEC facility can be deployed only in a location where either an AP or an aggregation node or a core node has been deployed: hence $K \subseteq (B \cup I \cup J)$.

Since we assume a superposition of stars as network topology (as described in Chapter 1.2), network physical links are established connecting AP to aggregation nodes, aggregation nodes to core nodes, and pairs of core nodes: hence let $E \subseteq (B \times I) \cup (I \times J) \cup (J \times J)$ be the set of feasible links between nodes.

Let l_i, m_j, c_k be the fixed cost for activating an aggregation node in $i \in I$, a core node in $j \in J$ and a MEC facility in $k \in K$, respectively.

Output (Decision Variables). We introduce two sets of variables. The first set corresponds to location binary variables: x_i take value 1 if an aggregation node is set in $i \in I$; y_j take value 1 if a core node is set in $j \in J$; z_k take value 1 if a MEC facility is set in $k \in K$. The second set corresponds to network topology binary variables:

- $t_{s,i}$ take value 1 if an **AP link** is established between an AP s and an aggregation node i;
- $w_{i,j}$ and $w_{j,i}$ simultaneously take value 1 if an **aggregation link** is established between an aggregation node i and a core node j;
- $o_{m,n}$ take value 1 if a **core link** is established between two core nodes m and n.

In order to model already existing or forbidden links, the corresponding variables can be fixed to value 1 and 0, respectively.

A complete notation table for the network design formulation can be found in Table 3.1.

Sets	
B	set of Access Points locations
I	set of aggregation nodes candidate locations
J	set of core nodes candidate locations
K	set of MEC facilities candidate locations
Data	
l_i	activation costs for aggregation node $i \in I$
m_j	activation costs for core node $j \in J$
c_k	activation costs respectively for MEC facility $k \in K$
Variables	
$x_i \in \mathbb{B}$	take value 1 if an aggregation node is set in $i \in I$
$y_j \in \mathbb{B}$	take value 1 if a core node is set in $j \in J$
$z_k \in \mathbb{B}$	take value 1 if a MEC facility is set in $k \in K$
$t_{s,i} \in \mathbb{B}$	take value 1 if an AP link is established between AP $s \in B$ and aggregation node $i \in I$
$w_{i,j} \in \mathbb{B}$	take value 1 if an aggregation link is established between aggregation $i \in I$ and core node $j \in J$
$o_{j,j'} \in \mathbb{B}$	take value 1 if a core link is established between a pair of core nodes $j, j' \in J$

Table 3.1: MNDP - Network Topology Notation Table

Objective function. Since the main purpose of the MNDP is the MEC network design, the model goal (3.1) is to minimize installation costs of all network facilities. We do not include the links installation costs as we do not take into consideration the cellular infrastructure dimensioning.

$$\min \sum_{i \in I} l_i x_i + \sum_{j \in J} m_j y_j + \sum_{k \in K} c_k z_k \tag{3.1}$$

Constraints. A complete MEC *network topology* results as a by-product of our model, in terms of arrangement of links. As specified in Chapter 1.2 we model this network as a superposition of stars: this has to be intended as a *topological rule*, which constrains the resulting arrangement of links.

Each AP is connected to a single aggregation node, and each aggregation node to a single core node (as depicted in Figure 1.1), while a full mesh is built among cores. The following set of constraints enforce our topological rules to be respected: each link (i, j) can be used only for one purpose (i.e. AP link, aggregation link or core link) - (3.2), while (3.3) and (3.4) enforce that core nodes and MEC facility nodes are also aggregation nodes: in particular this is possible only if a core node $j \in J$ (resp. MEC facility $k \in K$) is also a candidate facility for an aggregation node $i \in I$, and hence only if it exists an aggregation node $i \in I$ such that $i = j$ (resp. $i = k$);

$$t_{i,j} + w_{i,j} + o_{i,j} \leq 1 \qquad \forall (i,j) \in E \mid i \neq j \tag{3.2}$$

$$x_j \geq y_j \qquad \forall j \in J \mid \exists i \in I : i = j \tag{3.3}$$

$$x_k \geq z_k \qquad \forall k \in K \mid \exists i \in I : i = k \tag{3.4}$$

if (i, j) is an AP link then j is an aggregation node - (3.5), and, similarly, each AP is connected to either itself when chosen as aggregation, or a different node otherwise - (3.6) and (3.7);

$$t_{s,i} \leq x_i \qquad \forall (s,i) \in E \mid s \in B \wedge i \in I \tag{3.5}$$

$$t_{i,i} = x_i \qquad \forall i \in I \mid \exists s \in B : s = i \tag{3.6}$$

$$\sum_{\substack{s \in B \\ \mid\ s \neq i \wedge (s,i) \in E}} t_{s,i} = 1 - x_i \qquad \forall i \in I \tag{3.7}$$

given an aggregation link (i, j): it must be symmetric - (3.8); i is an aggregation

node - (3.9); and either i or j is a core node - (3.10);

$$w_{i,j} = w_{j,i} \qquad \forall (i,j) \in E \mid i \in I \wedge j \in J \tag{3.8}$$
$$w_{i,j} \leq x_i \qquad \forall (i,j) \in E \mid i \in I \wedge j \in J \tag{3.9}$$
$$w_{i,j} \leq y_i + y_j \qquad \forall (i,j) \in E \mid i \neq j \wedge (i \in J \vee j \in J) \tag{3.10}$$

given a core link (i,j): both i and j are core nodes - (3.11) and conversely if both i and j are core nodes, (i,j) is a core link - (3.12). Moreover no loops are considered at core links - (3.13). We remark that (3.11) can be disaggregated potentially improving LP relaxation bound at the price of doubling their number.

$$2o_{i,j} \leq y_i + y_j \qquad \forall (i,j) \in E \mid i,j \in J \wedge i \neq j \tag{3.11}$$
$$y_i + y_j - 1 \leq o_{i,j} \qquad \forall (i,j) \in E \mid i,j \in J \wedge i \neq j \tag{3.12}$$
$$o_{j,j} = 0 \qquad \forall j \in J \tag{3.13}$$

each aggregation node has an adjacent aggregation link, thereby connecting to a core node - (3.14), which can be the node itself - (3.15), at most one aggregation link can be connected to non-core nodes ($y_i = 0$), while an arbitrary number can be connected to core ones ($y_i = 1$) - (3.16).

$$\sum_{j \in J : (i,j) \in E} w_{i,j} \geq x_i \qquad \forall i \in I \tag{3.14}$$
$$w_{i,j} = y_j \qquad \forall j \in J \mid \exists i \in I : i = j \tag{3.15}$$
$$\sum_{\substack{i \in I \\ \mid (j,i) \in E \wedge i \neq j}} w_{j,i} \leq (1 - y_j) + |I| \cdot y_j \qquad \forall j \in J \tag{3.16}$$

We remark that these set of constraints (3.2)–(3.16) are meant only as a subset of the constraints of the full model. For instance, setting $z_k = 0$ for each $k \in K$ always respects them; however setting at least one of them to one by assignment constraints which are explained later in the Chapter.

3.3 Static Planning Formulation

Input (Problem Data). Each AP $s \in B$ can connect to a MEC facility located in $k \in K$ by a set of paths $\bar{S}^{sk}$ (see paths a, b, c and d in Fig. 1.1). Path $p \in \bar{S}^{sk}$ can traverse multiple sites and with $j \in p$ we denote that site j, that can be either an AP, an aggregation node or a core node, is traversed by path p.

For each AP $s \in B$, let n_s and b_s be the number of users connected to s and their overall bandwidth consumption. We assume that servicing each user requires the activation of one VM, and therefore n_s represents also the number of VMs needed for AP s.

Let C be the number of VMs that each MEC facility can host. Let $d_{i,j}$ and $u_{i,j}$ be the latency (latency or length are used interchangeably hereafter) and bandwidth capacity of each link $(i,j) \in E$. Let $U \in [0,1]$ be the parameter representing the maximum link utilization (percentage) in the network; indeed, as a common practice in IP traffic engineering with non deterministic loads, links need to have a level of over-provisioning so that they are robust against traffic fluctuations (due to failures, traffic peaks, etc) and hence the risk of congestion, which is particularly important for real-time and interactive services as those considered by MEC [3, 4].

Finally, we consider static and identical SLAs for all users, defined as the maximum allowed latency a user may experience, assuming it to be represented by three types of constraints: (i) maximum sum of link length in a path $\bar{D}$; (ii) maximum number of hops in a path $\bar{H}$ that according to [35] affects the effectiveness of MEC facilities; (iii) maximum distance allowed between nodes in the network to establish a link $\bar{d}$. We remark that with respect to this definition, and in particular with constraint (iii), some pairs of core nodes although connected one another may appear simultaneously in no feasible path.

Output (decision variables). To model routing decisions we introduce an additional set of binary variables: $r_p^{s,k}$ take value 1 if users in AP $s \in B$ are served by MEC facility in $k \in K$, and the corresponding traffic is routed along path $p \in \bar{S}^{sk}$.

Constraints. Feasible paths are those that satisfy SLA latency requirements defined previously. In order to enforce that only feasible paths are considered, we replace each set $\bar{S}^{sk}$ with the following set:

$$S^{sk} = \{p \in \bar{S}^{sk} : \sum_{(i,j)\in p} d_{(i,j)} \leq \bar{D} \wedge |p| \leq \bar{H} \wedge \ d_{(i,j)} \leq \bar{d} \ \forall (i,j) \in p\} \tag{3.17}$$

where by $|p|$ we denote the number of links forming path p.

Constraints (3.18) – (3.20) impose that each path from AP $s \in B$ to MEC facility $k \in K$, traversing either an aggregation node $i \in I$ or a core node $j \in J$, can be selected only if that network facility is installed in the corresponding site.

$$-\sum_{k\in K}\sum_{p\in S^{sk}|i\in p} r_p^{s,k} \geq -x_i \qquad \forall s\in B, \forall i\in I \tag{3.18}$$

$$-\sum_{k\in K}\sum_{p\in S^{sk}|j\in p} r_p^{s,k} \geq -y_j \qquad \forall s\in B, \forall j\in J \tag{3.19}$$

$$-\sum_{p\in S^{sk}} r_p^{s,k} \geq -z_k \qquad \forall s\in B, \forall k\in K \tag{3.20}$$

Constraint (3.21) sets to 1 the number of MEC facilities used by a single AP, as AP-level load-splitting is typically not performed in backhauling networks. For computational reason we relax the equality in (3.21) to a *greater or equal* inequality. (3.22) enrich (3.20) by further imposing that active MEC facilities provide at most C VMs. Constraints (3.23) ensure that capacity of link (i, j) is not exceeded.

$$\sum_{k\in K}\sum_{p\in S^{s,k}} r_p^{s,k} = 1 \qquad \forall s\in B \tag{3.21}$$

$$-\sum_{s\in B}\sum_{p\in S^{s,k}} n_s r_p^{s,k} \geq -Cz_k \qquad \forall k\in K \tag{3.22}$$

$$-\sum_{s\in B}\sum_{k\in K}\sum_{\substack{p\in S^{s,k}\\|(i,j)\in p}} b_s r_p^{s,k} \geq -u_{i,j}U(w_{i,j}+o_{i,j}+t_{i,j}) \qquad \forall (i,j)\in E \tag{3.23}$$

Finally we introduce constraints (3.24)–(3.26), which are not required to ensure the feasibility on the solution but are helpful during the resolution process. These impose a lower bound on the number of facilities to activate, represented as the three parameters $\underline{Z}$, $\underline{Y}$ and $\underline{X}$, which are related to the lower number of MEC facilities, core nodes and aggregation nodes, respectively. Methods used to compute the actual value for these parameter will be discussed in Section 3.7.1.

$$\sum_{k\in K} z_k \geq \underline{Z} \tag{3.24}$$

$$\sum_{j\in J} y_j \geq \underline{Y} \tag{3.25}$$

$$\sum_{i\in I} x_i \geq \underline{X} \tag{3.26}$$

Sets	
S^{sk}	set of feasible paths connecting AP $s \in B$ and MEC facilities $k \in K$, as defined in (3.17)
Data	
n_s	number of users connected to AP $s \in B$
b_s	bandwidth consumption in AP $s \in B$
$d_{i,j}$	length of link $(i,j) \in E$
$u_{i,j}$	bandwidth capacity of link $(i,j) \in E$
$\bar{D}$	maximum sum of links' length in a path
$\bar{H}$	maximum number of hops in a path
$\bar{d}$	maximum distance allowed between nodes to establish a link
U	maximum percentage of usable capacity of a link
C	VMs capacity of a MEC facility
Variables	
$r_p^{s,k} \in \mathbb{B}$	take value 1 if AP $s \in B$ is served by MEC facility in $k \in K$ through path p

Table 3.2: s-MNDP Notation Table

Overall, (3.1) – (3.26) represent the static planning variant of the MNDP, that we will refer to as s-MNDP. The notation for the s-MNDP is summarized in Table 3.2.

3.4 s-MNDP Matheuristic

Our path-based formulation offers great modeling flexibility and present computational challenges at once. In particular, the number of feasible paths in set S^{sk} grows very fast with the network size. In order to obtain good feasible solutions in limited computing time, we designed an Integer Linear Programming (ILP) based matheuristics, whose structure is presented in Figure 3.4.

Our algorithm consists in an iterative process based on six phases: (a) an initial feasible solution of a simplified problem on the clustered network is retrieved; (b) in order to reduce the cardinality of the network (i.e. the number of AP to consider) a clustering is performed; (c) on the clustered reduced network the continuous re-

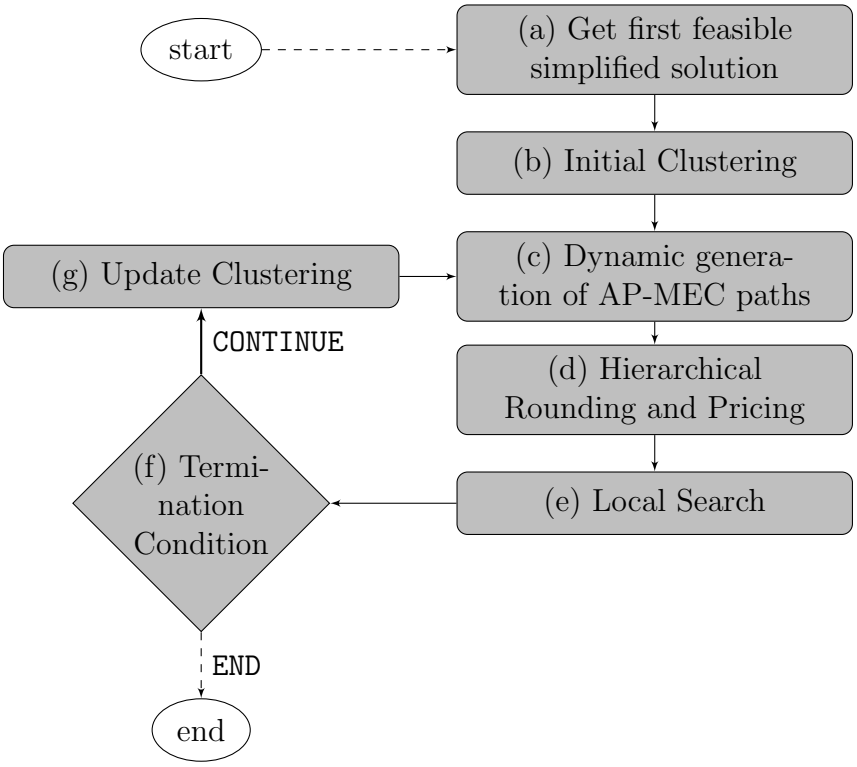


Figure 3.4: Overall Structure of s-MNDP Matheuristic

laxation s-MNDP is optimized by column generation; (d) in order to get an integer solution for our problem a hierarchical rounding and pricing process is executed; (e) on this integer solution a local search refinement is executed. As the solution is given on a clustered network, we want to change the clustering in order to find a possibly better solution on the original network; in (f) we check if a termination condition is satisfied: in (g) we update the clustering exploiting the information given by the fractional solution of the continuous relaxation of the problem and we restart the resolution process from step (c).

Hereafter we detail each step.

3.4.1 Capacitated Vertex Covering Rounding

A feasible solution for the s-MNDP is given by a simplified network in which a MEC facility is always a core node (and hence an aggregation node) for all connected APs.

The routing paths are always represented by a single hop link between nodes: hence link capacity constraints do not have to be considered as links are not shared among different paths. The resolution process needs to find only MEC facilities locations, without considering the location of cores and aggregation nodes and the selection of paths. Moreover we only have to enforce that the sum of demands of APs assigned to a MEC facility is less than its capacity C.

We model this heuristic with a mathematical programming approach as a Capacitated Vertex Covering (CVC) Problem. Let us introduce variable $\beta_{s,k}$, with value 1 if AP $s \in B$ is associated to MEC facility $k \in K$ and let us use the already defined variable z_k with value 1 if MEC facility is set in node $k \in K$. We formulate our problem as follows:

$$\min \sum_{k \in K} z_k \tag{3.27}$$

$$\text{s.t.} \sum_{k \in K} \beta_{s,k} = 1 \qquad \forall s \in B \tag{3.28}$$

$$\beta_{s,k} \leq z_k \qquad \forall s \in B, \forall k \in K \tag{3.29}$$

$$\beta_{s,k} = 0 \qquad \forall s \in B, \forall k \in K \mid \nexists (s,k) \in E \tag{3.30}$$

$$\sum_{s \in B} n_s \beta_{s,k} \leq C \qquad \forall k \in K \tag{3.31}$$

$$\boldsymbol{z} \in \{0,1\}, \boldsymbol{\beta} \in \{0,1\} \tag{3.32}$$

We minimize the number of activated MEC facilities (3.27); (3.28) impose that every AP is assigned to a MEC facility; (3.29) link the assignment variables $\beta_{s,k}$ to MEC facility activation variables z_k; (3.30) impose that an AP is not assigned to a MEC facility if a direct link does not exist in the network; finally (3.31) impose that the sum of demands of APs assigned to a MEC facility does not exceed its capacity C.

A feasible solution to our CVC is found with the following hierarchical rounding, also presented in Algorithm 3: given the fractional solution $\tilde{S}$ of the continuous relaxation of model (3.27) - (3.32), follow this steps:

(i) Sort fractional variables $\tilde{z}_k \in \tilde{S}$ by descending value;

(ii) Following the ordering fix a single free variable $\tilde{z}_k$ to value 1;

(iii) Given the MEC facility $k' \in K$ of the currently fixed variable $\tilde{z}'_k$, sort all corresponding variables $\tilde{\beta}_{s,k'}$ in descending order

Algorithm 3 CVC-round Heuristic for s-MNDP

$\tilde{S}$ = fractional solution of continuous relaxation of model (3.27) – (3.31)
A_s = False $\forall s \in B$ {store if AP s has been assigned to a MEC facility}
$Z^s = sortDesc(\tilde{z}_k \in \tilde{S})$ {sort fractional z_k by descending value}
for all $\tilde{z}_k \in Z^s \mid \tilde{z}_k > 0$ **do**
  fix $\tilde{z}_k$ to value 1
  $C_k = 0$
  $V^s = sortDesc(\tilde{\beta}_{s,k} \in \tilde{S})$ {sort fractional β_{sk} of current k by descending value}
  for all $\tilde{\beta}_{sk} \in V^s \mid \tilde{\beta}_{sk} > 0 \wedge C_k + n_s < C$ **do**
    fix $\tilde{\beta}_{sk}$ to value 1
    $C_k = C_k + n_s$ {update current used capacity}
    A_s = True
  end for
end for
for all $s \in B \mid A_s$ = False **do**
  fix $\tilde{z}_s$ to value 1 {not assigned AP is elected as MEC facility}
  fix $\tilde{\beta}_{s,s}$ to value 1
end for

(iv) Fix variable $\tilde{\beta}_{s,k'}$ to value 1 following the ordering if enough residual MEC facility capacity is available. Continue fixing until no more variables $\tilde{\beta}_{s,k'}$ with strictly positive fractional value can be fixed;

(v) Continue from step (ii) until no more variables $\tilde{z}_k$ with strictly positive fractional value can be fixed;

(vi) If rounding process ends and some APs are not associated to a MEC facility, these APs are elected as a new MEC facility and are assigned to themselves.

With respect to the last point, more sophisticated policies can be devised to deal with non-assigned APs, however we preferred to keep this initial heuristic as simple as possible and to focus on the main algorithm.

This algorithm always retrieves a feasible solution, given that the simplest one is for APs to be assigned to themselves as MEC facilities, and hence to elect every node in the network as a MEC facility. We will refer to this rounding heuristic algorithm as **CVC-round**.

3.4.2 Clustering

To allow the use of the model with big size data instances, we propose an approach that makes use of clustering to reduce instance size.

We cluster original APs of the network by selecting centers: we build a new network where cluster representatives are elected as fixed aggregation nodes, and APs belonging to a cluster are assigned to their cluster representative (i.e. a fixed AP-aggregation node assignment is established). We limit the number of clusters to consider, and therefore the size of the network, to the value $\lceil |B| \cdot \alpha \rceil$, where $\alpha \in (0, 1)$ is a parameter to set. To ensure feasibility we enforce that a cluster is composed by APs whose sum of demands is less than the maximum MEC facility capacity C: that is, given a subset of APs $\mathcal{C} \subseteq B$, it can be a cluster if $\sum_{s \in \mathcal{C}} n_s \leq C$. Moreover the representative of a cluster is chosen among APs that reach all other APs in the cluster with a feasible network link (i.e. their distance is less than $\bar{d}$). If no AP in the subset satisfies this condition, the subset can not represent a feasible cluster. The actual representative is given by the AP that minimize the maximum distance with all other APs in the cluster: that is, for the subset of APs $\mathcal{C}$, the representative $\bar{r}$ is given by: $\bar{r} = \arg\min_{r \in \mathcal{C}: d_{r,j} \leq \bar{d} \ \forall j \in \mathcal{C}} \{\max_{j \in \mathcal{C}} d_{r,j}\}$. For example, in Figure 3.5, APs from a to h are taken into consideration to be aggregated in a cluster. To choose a representative, APs a and g represent a feasible choice, given that all APs can be reached with link of length less than $\bar{d}$ (all nodes are inside the circles of radius $\bar{d}$ centred in a and g); on the contrary, AP f is not a feasible representative, given that it can not be connected with APs c, d and e with links of length less than $\bar{d}$ (these nodes are outside the circle of radius $\bar{d}$ centred in f).

The distance between two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ is given by the maximum distance between the representative of $\mathcal{C}_1$ and all APs in the other cluster $\mathcal{C}_2$. As an example, in Figure 3.6, two clusters of APs $\mathcal{C}_1$ and $\mathcal{C}_2$ are shown, with representative r and s, respectively: the distance between $\mathcal{C}_1$ and $\mathcal{C}_2$ is given by the maximum distance between r (the representative of $\mathcal{C}_1$) and all APs in $\mathcal{C}_2$; in this example the farthest AP from r is g, hence $d_{\mathcal{C}_1,\mathcal{C}_2} = d_{r,g}$. On the contrary, the distance between $\mathcal{C}_2$ and $\mathcal{C}_1$ is given by the maximum distance between s (the representative of $\mathcal{C}_2$) and all APs in $\mathcal{C}_1$: in the example the farthest AP in $\mathcal{C}_1$ to s is c, hence, $d_{\mathcal{C}_2,\mathcal{C}_1} = d_{s,c}$. In general $d_{\mathcal{C}_2,\mathcal{C}_1}$ could be different from $d_{\mathcal{C}_1,\mathcal{C}_2}$, that is the resulting distance matrix is not symmetric.

Thanks to these precautions a solution to the restricted network is also feasible for the original network.

To initialize the clustering we tested two approaches:

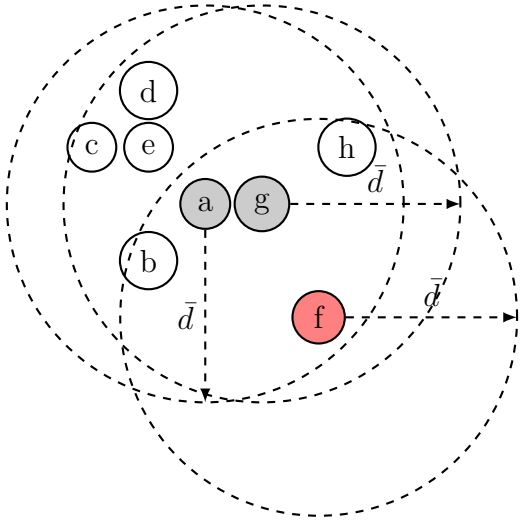


Figure 3.5: Cluster Representative Choice

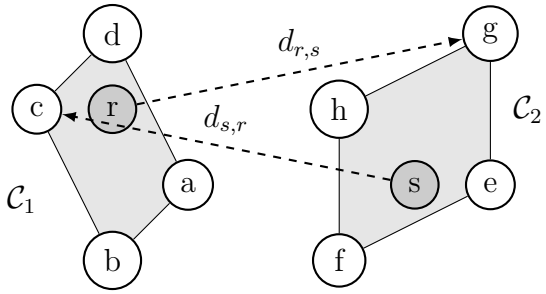


Figure 3.6: Distance Between Clusters

- using a k-medoid clustering method, in particular the `R` implementation of the Partitioning Around Medoid algorithm [68, 69], using the distances between nodes as dissimilarity matrix and using a high number of cluster $\lceil |B| \cdot \alpha \rceil$; compliance with constraints on sum of demands in each cluster is checked as a post-processing step;
- using an initial feasible solution found with the rounding heuristic CVC-round presented in Section 3.4.1.

3.4.3 Dynamic generation of paths

The number of feasible paths in $\bar{S}^{sk}$ to consider can be significant and may be intractable with the increase of the nodes in the network. Hence we consider the dynamic generation of feasible paths and their related variables r_p^{sk} with a column generation approach [70].

s-MNDP model (3.1) – (3.23) represents the Master Problem (MP). We replace each set S^{sk} by a small representative subset $\hat{S}^{sk}$. In our case this subset is built by the solutions produced by the initial heuristic algorithm CVC-round. We solve the corresponding Restricted Master Problem (RMP); for each AP $s \in B$, we search if any element in S^{sk} exists whose corresponding variable has negative reduced cost, by solving a *pricing problem*: any such element found through pricing is added to $\hat{S}^{sk}$ and the process is iterated. If no negative reduced cost element can be found, instead, we stop: the solution obtained by restricting to $\hat{S}^{sk}$ is also optimal for the full problem. Such a solution provides a valid lower bound to the s-MNDP. As we are dealing with the continuous relaxation of s-MNDP, the provided solution is fractional; however if an integer solution is provided, it is also the optimal solution for the s-MNDP.

For each $s \in B$ the problem of finding the element of S^{sk} corresponding to the variable of minimum reduced cost can be formulated as follows. Let us introduce new sets of variables that represent elements of the path: let $\bar{l}_{m,n}$ take value 1 if link (m, n) is used in the path; let $\bar{b}_k$ take value 1 if MEC facility k is used (i.e. is the endpoint of the path) and $\bar{a}_j$ take value 1 if core facility j is used (i.e. if the path passes through the node).

The costs to be minimized in the pricing objective function (3.33) are given by the non-negative dual variables corresponding to master problem constraints: $\nu_{i,s}$ from (3.18); $\mu_{j,s}$ from (3.19); $\gamma_{k,s}$ from (3.20); ω_s from (3.21); π_k from (3.22); and $\rho_{m,n}$ from (3.23).

$$\min_{s \in B} \; -\omega_s + \sum_{i \in I} \bar{l}_{s,i} \nu_{i,s} + \sum_{j \in J} \bar{a}_j \mu_{j,s} + \sum_{k \in K} \bar{b}_k (\pi_k n_s + \gamma_{k,s}) + \sum_{(m,n) \in E} \bar{l}_{m,n} \rho_{m,n} b_s \quad (3.33)$$

Constraints (3.34) – (3.38) respectively ensure that: (3.34) - a flow of one unit exit the source; (3.35) - the flow either exits from an intermediate node or remains in the node that becomes the cloud facility, (3.36) - an intermediate node is a core node; (3.37) - the flow can not enter the source; (3.38) - at least one core is selected.

$$\sum_{i \in I | i \neq s} \bar{l}_{s,i} = 1 \tag{3.34}$$

$$\sum_{(m,k) \in E} \bar{l}_{m,k} = \bar{b}_k + \sum_{(k,m) \in E} \bar{l}_{k,m} \qquad \forall k \in K \mid k \neq s \tag{3.35}$$

$$\sum_{(k,j) \in E} \bar{l}_{k,j} + \sum_{(j,k) \in E} \bar{l}_{j,k} - 1 \leq \bar{a}_j \qquad \forall j \in J \mid j \neq s \tag{3.36}$$

$$\sum_{j \in (I \cup J \cup K) | j \neq s} \bar{l}_{j,s} = 0 \tag{3.37}$$

$$\sum_{j \in J} \bar{a}_j \geq 1 \tag{3.38}$$

Finally constraints (3.39) – (3.41) ensure the feasibility of the paths, respectively in the maximum sum of links length of the path, the maximum number of hops and the maximum length of the single link.

$$\sum_{(m,n) \in E} d_{m,n} \bar{l}_{m,n} \leq \bar{D} \tag{3.39}$$

$$\sum_{(m,n) \in E} \bar{l}_{m,n} \leq \bar{H} \tag{3.40}$$

$$\bar{l}_{m,n} = 0 \qquad \forall (m,n) \in E \mid d_{m,n} \geq \bar{d} \tag{3.41}$$

Pricing problem (3.33) – (3.41) represents a resource constrained shortest path problem. While in general this class of problem is NP-Hard, in this case it can be conveniently solved with a dynamic programming approach, given that it can be mapped to a layered graph with a number of layers limited to the value of $\bar{H}$.

We build a directed layered graph $G(N, A)$ with a layer for each number of hops in the path. Let $\mathcal{L} = \left\{1, ..., \bar{H}\right\}$ be the set of layers; each layer reflects the hierarchy of nodes in network topology: hence the first layer represents the APs, the second layer the aggregation nodes, layers from the third to the $(\bar{H} - 1)$-th represent core nodes, while the last layer $\bar{H}$ represents the MEC facility. Our graph is directed and layered, there is no possibility for cycles and all paths are elementary. We exploit such a layered structure assigning to each node i in layer $l \in \mathcal{L}$ of graph G states of the dynamic programming execution, while arcs in the graph represents

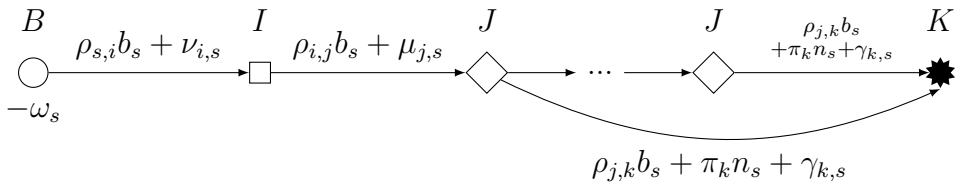


Figure 3.7: Layer Structure of the Dynamic Programming Algorithm for AP-MEC facility Association Path Variables $r_p^{s,k}$.

state transitions. Different states, associated with the same node (i, l), correspond to different feasible paths p reaching i at layer l. They differ by: the subset of nodes S traversed by the current path, and the length of the current path d. Hence each state can be represented by a label $((i, l), S, d)$. Moreover to every label a cost is associated, $c((i, l), S, d)$, corresponding to the cost of the path. Arcs in the graph, representing state transition, reflect feasible hops in the network topology: from first layer (APs) arcs connect to the second layer only (aggregation nodes); from the second layer arcs connect to the third layer only (core nodes); from all core nodes layers (from the third to the $(\bar{H} - 1)$-th) arcs connect to both the succeeding layer and the last layer $\bar{H}$, of the MEC facility. Moreover, each arc has an associated cost, given by the network link costs and the device activation costs defined in pricing objective function (3.33). The layer structure of the dynamic programming algorithm is sketched in Figure 3.7.

The main steps of our algorithm are the following:

Initialization The label for the source $s \in B$ is initialized with a cost equal to the corresponding dual variable ω_s and the following values:

$$c((s, 1), \{s\}, 0) = -\omega_s \tag{3.42}$$

Propagation Labels in the second layer, corresponding to aggregation nodes, are connected to labels in the first layer if and only if a feasible network link connects their nodes, that is a link between label $((i, 2), \{s, i\}, d_{s,i})$ of the second layer and label $(s, \{s\}, 1, 0)$ of the first layer exits if and only if $d_{s,i} < \bar{d}$. Label costs are initialized with the following value:

$$c((i, 2), \{s, i\}, d_{s,i}) = c((s, 1), \{s\}, 0) + \rho_{s,i}b_s + \nu_{i,s} \quad \forall i \in I \setminus \{s\} \mid d_{s,i} < \bar{d} \tag{3.43}$$

Labels in layers corresponding to core nodes, that is from the third to the $(\bar{H} - 1)$-th

layer, are extended according to the following rule:

$$c((j,l+1),S,d) = \min_{i \in S \setminus \{j\} | d_{i,j} < \bar{d}} \{c((i,l), S \setminus \{j\}, d - d_{i,j}) + \rho_{i,j} b_s\} + \mu_{j,s}$$
$$\forall j \in J, \forall l \in \{2..(\bar{H}-2)\} \mid d < \bar{D} \tag{3.44}$$

Labels in the last layer, corresponding to MEC facility, are linked to every node of every core node layers that can be connected through a feasible network link, and are extended according to the following rule:

$$c((k+\bar{H}),S,d) = \min_{j \in S \setminus \{k\} |\ d_{j,k} < \bar{d}} \{c((j,l), S \setminus \{k\}, d - d_{j,k}) + \rho_{j,k} b_s\} + \pi_k n_s + \gamma_{k,s}$$
$$\forall k \in K, \forall l \in \{3..(\bar{H}-1)\} \mid d < \bar{D} \tag{3.45}$$

Stopping The minimum cost path is given by:

$$\min_{k \in K} c((k,\bar{H}),S,d) \tag{3.46}$$
$$\text{s.t. } d < \bar{D} \tag{3.47}$$

We remark that, from a strictly worst-case time complexity point of view, exploiting the layered structure of G in the dynamic programming routines allows us to obtain algorithms which run in $\mathcal{O}(|A|)$, where A is the set of arcs of G, and are therefore more efficient than generic ones from the literature.

3.4.4 Hierarchical round and price

As the column generation process leads to a *fractional* solution $\bar{s}$, to obtain an integer feasible one, a hierarchical rounding on the variables is executed: (i) select the location variable $\bar{f}$ with higher fractional value in $\bar{s}$ that was not already fixed, and fix it to value one, (ii) propagate the rounding, by fixing to zero all variables that would lead to infeasibility when set to one, (iii) resume column generation, to dynamically generate new paths given the new fixed variables, (iv) if a new fractional solution is found, repeat rounding from step (i); instead, if no feasible solution can be found after fixing, reset $\bar{f}$ to value zero, undo rounding propagation and resume column generation; if a feasible solution is found, repeat rounding from step (i), otherwise stop rounding with `FAIL`. (v) Stop with `SUCCESS` whenever $\bar{f}$ has a fractional value in $\bar{s}$ that is lower than a small enough positive threshold ϵ.

Instead of choosing an arbitrary $\bar{f}$, we perform rounding according to the following hierarchy: (i) MEC facility location variables z_k, (ii) core nodes location variables y_j, (iii) aggregation nodes location variables x_i, (iv) paths variables $r_p^{s,k}$. That is, each hierarchical level is explored only if no previous one contains a fractional variable. Variables related to topological rules are never rounded explicitly. At the end of the rounding process, in case of `SUCCESS`, a MILP problem remains to fix them, involving a small number of variables, which can be easily optimized by general purpose ILP solvers. Nevertheless, we often observed that network topology variables take integer values directly after rounding: in these cases we skip this last MILP optimization process. In case of `FAIL`, instead, the solution produced in step (a) is considered. That is, in any case our static planning algorithm produces a feasible solution, unless the instance itself admits no feasible one.

We remark that, according to this hierarchy, a `FAIL` status can only be triggered in step (iv), when paths variables $r_p^{s,k}$ are considered for rounding. In our instances, where we consider sets of candidates nodes to be equal to the set of APs (i.e. $B = I = J = K$), the rounding process never reach the `FAIL` status, given that is always possible to assigning an AP to itself, elected as a MEC facility.

3.4.5 Local search

Given an integer feasible solution $\bar{S}$, we tried to improve it with an ILP-based very large scale neighborhood search strategy, exploring a *κ-OPT neighborhood*.

We propose a compact version of s-MNDP model which does not consider each path but each possible link of the network. Let us introduce set $H = \{1, \ldots, \bar{H}\}$ representing the hierarchical level of the network. We replace variables r_p^{sk} with variables $r_{i,j}^{s,h} \in \{0,1\},\ \forall (i,j) \in E, \forall s \in B, \forall h \in H$ with value 1 if a path with source in node s use link (i,j) at level h, 0 otherwise.

Links of a path connecting an AP to a MEC facility are represented in the following way:

- a link between an AP to an aggregation node is modeled through variables $t_{s,i}$ and is considered at hierarchical level zero
- a link between an aggregation node to a core node is modeled through variables $r_{(i,j)}^{h,1}$ at first hierarchical level
- a link between two cores is at second hierarchical level and modeled with variables $r_{(i,j)}^{h,2}$

- a link between a node (aggregation or core) and a MEC facility is at third hierarchical level and modeled with variables $r^{h,3}_{(i,j)}$

The corresponding compact model is the following:

min (3.1)
s.t. (3.2) − (3.16)

$$\sum_{\substack{i\in I \mid \\ (s,i)\in E}} t_{s,i} = 1 \qquad \forall s \in B \quad (3.48)$$

$$t_{s,i} \leq x_i \qquad \forall (s,i) \in E \quad (3.49)$$

$$r^{s,h}_{i,j} \leq \begin{cases} y_j \text{ if } h \in \{1,\ldots,(\bar{H}-1)\} \\ y_i \text{ if } h \in \{2,\ldots,\bar{H}\} \\ z_j \text{ if } h = \bar{H} \end{cases} \qquad \forall (i,j) \in E, \forall s \in B, \forall h \in H \quad (3.50)$$

$$\sum_{s\in B} n_s \sum_{\substack{i\in B \mid \\ (i,k)\in E}} r^{s,\bar{H}}_{i,k} \leq C z_k \qquad \forall k \in K \quad (3.51)$$

$$\sum_{s\in B} b_h \sum_{h\in H} r^{s,h}_{m,n} \leq u_{m,n} U (w_{m,n} + o_{m,n}) \qquad \forall (m,n) \in E \mid m \neq n \quad (3.52)$$

$$t_{s,i} = \sum_{(i,k)\in E} (r^{s,1}_{i,k} + r^{s,\bar{H}}_{i,k}) \qquad \forall s \in B, \forall i \in I \quad (3.53)$$

$$\sum_{\substack{i\in I \mid \\ (i,j)\in E}} r^{s,h}_{i,j} \leq \sum_{h'\in\{h+1,\bar{H}\}} \sum_{\substack{k\in K \mid \\ (j,k)\in E}} r^{s,h'}_{j,k} \qquad \forall h \in \{1..(\bar{H}-1)\}, \substack{\forall s\in B \\ \forall j\in B} \quad (3.54)$$

$$r^{s,1}_{m,n} \leq w_{m,n} \qquad \forall (m,n) \in E, \forall s \in B \quad (3.55)$$

$$r^{s,h}_{m,n} \leq o_{m,n} \qquad \substack{\forall (m,n)\in E \\ \forall s\in B}, \ \forall h \in \{2..(\bar{H}-1)\} \quad (3.56)$$

$$\sum_{\substack{j\in I \mid \\ (s,j)\in E}} d_{s,j} t_{s,j} + \sum_{(j,k)\in E} d_{j,k} \sum_{h\in H} r^{s,h}_{j,k} \leq \bar{D} \qquad \forall s \in B \quad (3.57)$$

Constraints (3.48), (3.49) enforce that each AP is connected to one and only one aggregation node. Constraints (3.50) link variables $r^{s,h}_{i,j}$ to location variables y_i and z_i: (i) node j of the link (i,j) is a core from the first to the last but one hierarchical level; (ii) node i of the link (i,j) is a core from the second to the last hierarchical level; (iii) last hierarchical level activates a MEC facility at node j of the link (i,j).

Constraints (3.51), (3.52) require compliance respectively for the VMs capacity of a MEC facility and for the bandwidth capacity of each link. Constraints (3.53)-(3.54) model the flow conservation from an AP to a MEC facility: (i) in (3.53) a flow from an AP to an aggregation node must either generate a flow from and aggregation to a core or to elect the aggregation node to a MEC facility; (ii) (3.54) impose that a flow from a hierarchical level goes either to its successive level or to the last level, i.e. to the MEC facility. Constraints (3.55), (3.56) link flow variables $r^{s,h}_{(i,j)}$ to topology variables $w_{i,j}, o_{i,j}$. Finally constraints (3.57) requires compliance with the maximum length of a path.

We devised the following local search strategy:

- we allow the solution to use only the paths created by the column generation process and the successive hierarchical rounding and pricing; that is, we fix $r^{s,h}_{i,j}$ to value 0 if no path created in the pricing process for the AP s uses the link (i, j) for hierarchy h;
- we add to the model the soft-fixing constraint (3.58), presented in [71] where is called *k-OPT neighborhood*:

$$\sum_{k \in K | \bar{z}_k = 1} (1 - z_k) + \sum_{k \in K | \bar{z}_k = 0} z_k \leq \lceil \kappa \cdot \sum_{k \in K} \bar{z}_k \rceil \tag{3.58}$$

  where parameters $\tilde{z}_k$ represent the values of the variables z_k in $\bar{S}$, and parameter κ represents the fraction of z_k variables whose values are allowed to flip with respect to the current solution.
- we solve this restricted model with a general purpose ILP Solver, setting a limit τ on the execution time.

We run only one iteration of this local search strategy.

3.4.6 Clustering Update Restart Strategy

At the end of the resolution process of the s-MNDP with the restricted network resulting from the initial clustering, as a restart strategy we update the clustering using the fractional solution given by the column generation as a guide:

- when the AP h is associated with MEC facility k through path p at integrality (i.e. $r^{s,k}_p = 1$), the two clusters represented by h and k are joined, a new representative is found and their demands are summed; for example in Figure

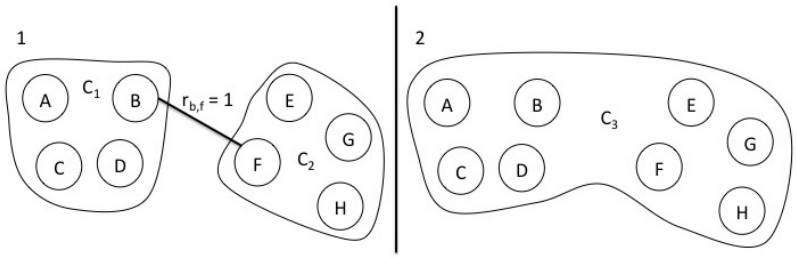


Figure 3.8: Join Clusters

3.8 clusters C_1 is associated to cluster C_2 at integrality: they can be joined in a new cluster C_3 and a new representative has to be chosen.

- if AP h is associated fractionally through multiple paths, the cluster that it represents is split in multiple clusters ignoring the information given by the fractional variables, but rather trying to improve a measure of *connectivity* on the restricted network.

With *connectivity* we mean the measure of the number of pairs of APs that can be reached in a single hop in the restricted network. In order to update the clustering using the connectivity measure, we follow the following steps: (i) compute for all nodes of the original network the individual connectivity, that is the number of APs they can reach within the actual restricted network (ii) sort all nodes by descending connectivity (iii) first N sorted nodes become a cluster composed by a single element (i.e. N separated singleton clusters), where N is the difference between the maximum number of clusters $\lceil |B| \cdot \alpha \rceil$ and the current number of clusters. These nodes can be separated from the cluster to which they belong only if they're not the only eligible representative of the cluster and if their connectivity is greater than zero.

For example, in Figure 3.9 cluster C_1 is assigned fractionally to both cluster C_2 and C_3 and has to be split: C_1 is composed by four APs (from A to D) for each of which the connectivity is measured. B and D can reach both clusters C_2 and C_3, hence their connectivity is equal to value 2; node C can only reach cluster C_2, hence its connectivity is equal to value 1; node A can not reach any cluster, so its connectivity is equal to 0. Nodes are sorted by descending connectivity, and for each node in this order we check if it can be released forming a new singleton cluster without exceed the maximum number of clusters $\lceil |B| \cdot \alpha \rceil$ (in this example equal to 5) and if they're not the only eligible representative of their former cluster. In this example B and D, having higher connectivity, are released first without

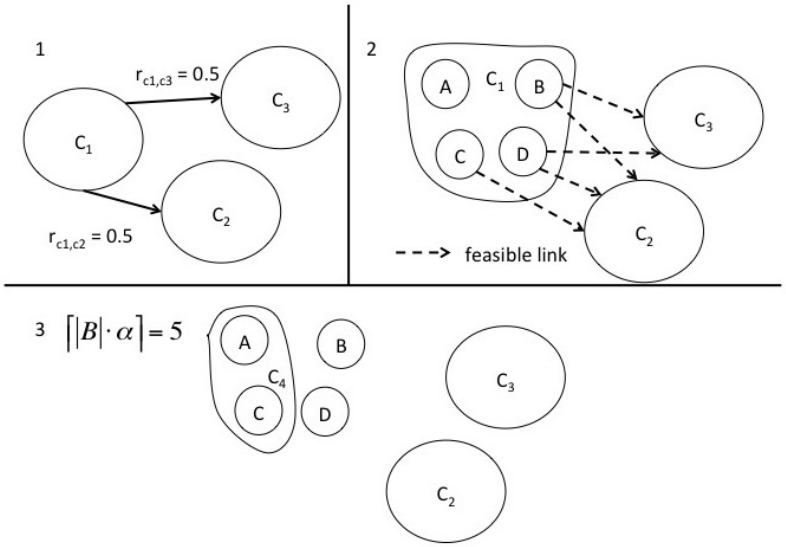


Figure 3.9: Split Clusters Using *Connectivity* Measure

violating these two conditions; on the contrary, C can not be released because we have reached the maximum number of clusters. At the end of the splitting process we have 5 clusters: C_2, C_3, the singletons B and D and the cluster C_4 composed by nodes A and C.

3.5 Dynamic Planning Formulation

In the second variant of the MNDP, we consider the dynamic status of the network. As users move during the planning horizon, they connect to different APs, changing the network load distribution, with the necessity to re-plan the network to re-balance the system. Moreover as they move they may distance themselves from their VM, worsening their QoEs and violating their SLA. In order to re-balance the system and to enforce SLA we introduce VMs mobility in our model.

3.5.1 Time Planning Horizon Discretization

We partition the planning horizon in periods called *time-frames*, identified by set T. To consider the changing in the network load distribution, let n_s^t and b_s^t be the (average) number of users connected to AP $s \in B$ and their overall bandwidth

consumption during time-frame $t \in T$. We consider the user mobility during the overall given horizon without making assumptions on the users positions in a specific point in time, yet we assume that in a single time-frame a user can connect to a single AP; in particular, let $f_{s's''}$ be the number of users moving from AP $s' \in B$ to AP $s'' \in B$ during time horizon T. We allow routing decisions to be changed dynamically, i.e. we allow an AP $s \in B$ to be assigned to different MEC facilities $k \in K$ in different time-frames $t \in T$, replacing the variable $r_p^{s,k}$ with a set of variables $r_p^{s,k,t}$ for each $t \in T$. Due to the features of our application location decisions, instead, are kept fixed. Constraints (3.18) – (3.23) of s-MDNP are extended as the following dynamic planning variant:

$$-\sum_{k\in K}\sum_{p\in S^{sk}|i\in p} r_p^{s,k,t} \geq -x_i \qquad \forall s \in B, \forall i \in I, \forall t \in T \quad (3.59)$$

$$-\sum_{k\in K}\sum_{p\in S^{sk}|j\in p} r_p^{s,k,t} \geq -y_j \qquad \forall s \in B, \forall j \in J, \forall t \in T \quad (3.60)$$

$$-\sum_{p\in S^{sk}} r_p^{s,k,t} \geq -z_k \qquad \forall s \in B, \forall k \in K, \forall t \in T \quad (3.61)$$

$$\sum_{k\in K}\sum_{p\in S^{s,k}} r_p^{s,k,t} \geq 1 \qquad \forall s \in B,\ \forall t \in T \quad (3.62)$$

$$-\sum_{s\in B}\sum_{p\in S^{s,k}} n_s^t r_p^{s,k,t} \geq -Cy_k \qquad \forall k \in K,\ \forall t \in T \quad (3.63)$$

$$-\sum_{s\in B}\sum_{k\in K}\sum_{\substack{p\in S^{s,k}\\|(i,j)\in p}} b_s^t r_p^{s,k,t} \geq -u_{i,j}U(w_{i,j}+o_{i,j}+t_{i,j}) \qquad \forall (i,j) \in E, \forall t \in T \quad (3.64)$$

These are composed by single instances of location variables and $|T|$ copies of each path variable and constraints (3.18) – (3.23) of s-MNDP. However, the instances of location variables are not independent one another, being linked by constraints (3.59), (3.60) and (3.61).

3.5.2 Modelling User Mobility

To include in dynamic planning model the user mobility, let variables $g_{s's''}^{k'k''} \in \mathbb{Z}_+$ represent the amount of users connecting through the planning horizon to APs $s' \in B$ and $s'' \in B$ served by MEC facilities in sites $k' \in K$ and $k'' \in K$, respectively. Let

also binary variables v_{sk} take value 1 if AP $s \in B$ is assigned to a MEC facility in $k \in K$ in at least one time-frame. Following constraints are needed to enforce coherence among these additional variables:

$$-\sum_{p \in S^{sk}} r_p^{s,k,t} \geq -v_{sk} \qquad \forall s \in B, \forall k \in K,\ \forall t \in T \tag{3.65}$$

$$g_{s's''}^{k'k''} \geq (v_{s'k'} + v_{s''k''} - 1) f_{s's''} \qquad \forall s', s'' \in B,\ \forall k', k'' \in K \tag{3.66}$$

We remind that $f_{s's''}$ models the number of users moving from the region covered by AP s' to the region covered by AP s'': every MEC facilities to which this pair of APs connect during the planning horizon must be synchronized. If an AP s is connected to MEC facility k in any time-slot, constraints (3.65) enforce v_{sk} to take value 1. Constraints (3.66) have no effect unless connections occur both from s' to k' and from s'' to k'' independently from the time-slot. In the first case, it is feasible to set $g_{s',s''}^{k',k''}$ to value 0; otherwise constraints (3.66) enforce $g_{s',s''}^{k',k''}$ to take the value $f_{s',s''}$ counting the number of users moving from the region covered by AP s' to the region covered by AP s''. That is, constraints (3.65) and (3.66) allow to link the number of users moving from s' to s'' (which is assumed to be known as input) to the number of users whose VMs need to be synchronized between MEC facilities k' and k'' which are instead a result of the decision process described by our models.

In the following we define the set of constraints modelling the three VM mobility technologies presented in Chapter 1.3.

3.5.3 VM replication

We model the VM replication option including explicitly in our model the routing and congestion assessment arising from MEC facility to MEC facility synchronization traffic. Let $\bar{Q}^{k'k''}$ be the set of paths connecting MEC facility facilities installed in $k', k'' \in K$, through which to route the synchronization traffic of copies of a VM deployed in the two MEC facilities. We refer to these as *synchronization paths*. Let $\bar{D}^Q$ and $\bar{L}^Q$ be the counterpart of $\bar{D}$ and $\bar{H}$ for synchronization paths, and let:

$$Q^{k'k''} = \{p \in \bar{Q}^{k'k''} : \sum_{(i,j) \in p} d_{(i,j)} \leq \bar{D}^Q \wedge\ |p| \leq \bar{H}^Q \wedge\ d_{(i,j)} \leq \bar{d}\ \forall (i,j) \in p\} \tag{3.67}$$

represent the set of *feasible* synchronization paths between k' and k''. Then, let continuous variables $q_p^{k'k''t} \in \mathbb{R}_+$ represent the amount of synchronization traffic

between MEC facility facilities in $k' \in K$ and $k'' \in K$ routed along path $p \in Q^{k'k''}$ during time-frame $t \in T$. A path $p \in Q^{k'k''}$ can traverse multiple sites and with $j \in p$ we denote that site j is traversed by path p. The following constraints enforce coherence among these additional variables:

$$\sum_{p \in Q^{k'k''}} q_p^{k'k''t} \geq \sum_{\substack{s',s'' \in B \\ |s' \neq s''}} \Phi(g_{s',s''}^{k',k''}) \qquad \forall k', k'' \in K | k' \neq k'', \forall t \in T \tag{3.68}$$

$$-\sum_{\substack{p \in Q^{k'k''} \\ |i \in p}} q_p^{k'k''t} \geq -x_i \sum_{s',s'' \in B} \Phi(f_{s',s''}) \qquad \forall i \in I, \forall k', k'' \in K, \forall t \in T \tag{3.69}$$

$$-\sum_{\substack{p \in Q^{k'k''} \\ |j \in p}} q_p^{k'k''t} \geq -y_j \sum_{s',s'' \in B} \Phi(f_{s',s''}) \qquad \forall j \in J, \forall k', k'' \in K, \forall t \in T \tag{3.70}$$

and link utilization constraints (3.64) become:

$$-\sum_{\substack{(s,k) \in \\ B \times K}} \sum_{\substack{p \in S^{s,k} \\ |(i,j) \in p}} b_s^t r_p^{s,k,t} - \sum_{\substack{k',k'' \in K \\ |k' \neq k''}} \sum_{\substack{p \in Q^{k'k''} \\ |(i,j) \in p}} q_p^{k',k'',t} \geq -u_{i,j} U(w_{i,j} + o_{i,j} + t_{i,j}) \; {}^{\forall (i,j) \in E}_{\forall t \in T} \tag{3.71}$$

Function $\Phi : \mathbb{Z}_+ \to \mathbb{R}$ maps the number of moving users $g_{s',s''}^{k',k''}$ to the amount of synchronization traffic they induce among MEC facilities. A possible implementation of function Φ will be presented in Section 3.7.1. The dynamic planning VM replication variant of the MNDP, to which we will refer to with r-MNDP, is therefore obtained by applying (3.1)-(3.16), (3.59)-(3.63), (3.65)-(3.66), (3.68)-(3.71).

3.5.4 Bulk and Live VM Migration

The dynamic association of users to a nearer MEC facility allows an improvement in their QoE, with a possible worsening of the status of the network. Hence the expected number of user migrations given by variables $g_{s's''}^{k'k''}$ has to be limited by the number of migrations that the network infrastructure can handle in an amount of time such that the migration ends before the user moves further, which we will refer to as *useful* migrations. Given the parameters:

- T_w: the temporal window during which the migration of the VM is useful. This values is strictly related to the user's sojourn time in an area T_s, and usually $T_w \ll T_s$;
- V: the size of the VM file to migrate;

the number of migrations that a link can manage is given by:

$$\frac{(1-U)\cdot u_{i,j}\cdot T_w}{V} \tag{3.72}$$

We therefore limit the number of VMs migrations that a single link can handle, with the following constraints:

$$-\sum_{\substack{(k',k'')\\ \in K\times K}}\sum_{\substack{p\in Q^{k',k''}|\\ (i,j)\in p}} \Phi^{-1}\big(q_p^{k',k'',t}\big) \geq -\frac{(1-U)\cdot u_{i,j}\cdot T_w}{V} \quad \forall (i,j)\in E, \forall t\in T \tag{3.73}$$

where Φ^{-1} is the inverse of function Φ found in (3.68), retrieving the number of migration routed through link (i,j).

Dynamic planning Bulk VM Migration variant of the MNDP, to which we will refer to with b-MNDP, is therefore obtained by the set of equations (3.1)-(3.16), (3.59)-(3.66), (3.68)-(3.70) and (3.73),

Dynamic planning Live VM Migration variant of the MNDP, to which we will refer to with l-MNDP, is obtained by the set of equations (3.1)-(3.16), (3.59)-(3.63), (3.65)-(3.66), (3.68)-(3.71) and (3.73). This latter case can be seen as the union of the VM replication and the Bulk Migration, given that it considers both the synchronization traffic (in constraints (3.71)) and the limit on the number of VMs migrations (in constraints (3.73)).

A Notation table for the dynamic planning variants of the MNDP is presented in Table 3.3.

3.6 l-MNDP Matheuristic

We propose a matheuristic algorithm for the Dynamic Planning Live VM Migration variant of the MNDP (l-MNDP) presented previously in Section 3.5.4, that includes

Sets	
T	set of time-frames, partitioning of planning horizon
$Q^{k'k''}$	set of paths connecting two MEC facilities $k', k'' \in K$
Data	
n_s^t	number of users connected to AP $s \in B$ in time-frame $t \in T$
b_s^t	bandwidth consumption in AP $s \in B$ in time-frame $t \in T$
$f_{s',s''}$	number of users moving from AP $s' \in B$ to AP $s'' \in B$
$\bar{D}^Q$	maximum sum of links' length in a path $q \in Q^{k'k''}$
$\bar{H}^Q$	maximum number of hops in a path $q \in Q^{k'k''}$
T_w	temporal window during which the migration of the VM is useful
V	size of the VM file to migrate
$\Phi : \mathbb{Z}_+ \to \mathbb{R}_+$	function mapping number of users to synchronization traffic
Variables	
$r_p^{s,k,t} \in \mathbb{B}$	take value 1 if AP $s \in B$ is served by MEC facility in $k \in K$ through path p in time-frame $t \in T$
$v_{s,k} \in \mathbb{B}$	take value 1 if AP s is served by MEC facility k in any time-frame
$q_p^{k',k'',t} \in \mathbb{R}_+$	amount of synchronization traffic between MEC facilities $k', k'' \in K$ through path p in time-frame t
$g_{s',s''}^{k',k''} \in \mathbb{Z}_+$	Number of users connecting to APs s' and s'' served by MEC facilities k' and k'', resp., through the planning horizon

Table 3.3: MNDP - Dynamic Planning Notation Table

all the constraints of two other VM mobility technology variants, i.e. the VM Replication and the Bulk VM Migration.

Optimizing the l-MNDP is even more involved than the optimization of s-MNDP. First, a copy of each association path $r_p^{s,k,t}$ needs to be considered for each time-frame t. Second, the set of sync-paths variables $q_p^{k',k'',t}$ may grow combinatorially as well. Third, the number of variables $g_{s',s''}^{k',k''}$ used in constraints (3.66) and (3.68) is polynomial, but too large to be explicitly considered in practice: by considering them, we were not able to run our algorithm.

In order to deal with the latter issue, we relax constraints (3.66) and (3.68) with the following constraints:

$$\sum_{q \in Q^{k',k''}} q_p^{k',k''t} \geq \sum_{\substack{s' \in B \\ s'' \in B}} \Phi(f_{s',s''}) \cdot (v_{s'k'} + v_{s''k''} - 1) \; \forall k', k'' \in K | k' \neq k'', \forall t \in T \tag{3.74}$$

When integrality conditions are enforced, (3.74) are equivalent to (3.66), (3.68). Unfortunately, this is not always true when the continuous relaxations are considered during rounding; we therefore strengthened them with the following inequalities:

$$v_{s',k'} + v_{s'',k''} \leq 1 \qquad \forall s',s'' \in B, k',k'' \in K \mid \begin{array}{l} f_{s',s''} > 0 \\ \exists\ p \in S^{s,k} \\ \exists\ p \in S^{s',k'} \\ \nexists\ p \in Q^{k',k''} \end{array} \forall t \in T \tag{3.75}$$

$$v_{s',k'} - \sum_{\substack{k'' \in K \mid \\ \exists p \in S^{s'',k''} \\ \exists p \in Q^{k',k''}}} v_{s'',k''} \leq 0 \qquad \forall s',s'' \in B \mid \begin{array}{l} f_{s',s''} > 0 \ \wedge \\ \exists p \in S^{s',k'} \end{array} \forall k' \in K \tag{3.76}$$

$$\sum_{t \in T} \sum_{p \in S^{s,k}} r_p^{s,k,t} \geq v_{sk} \qquad \forall s \in B, \forall k \in K \tag{3.77}$$

where (3.75) forbid the simultaneous choice of AP-MEC facility associations that does not allow to establish a feasible synchronization path, (3.76) states that, for each pair of APs with expected users migration, at least a pair of AP-MEC facility associations having a feasible synchronization path has to be activated, and (3.77) ensure that AP-MEC facility association variable $v_{s,k}$ is activated only if a related path variable $r_p^{s,k,t}$ is activated in any time-frame t.

An initial solution is not easily available for the l-MNDP, as opposed to the s-MNDP; hence, the initial restricted master problem does not contain any path variables, either $r_p^{s,k,t}$ or $q_p^{k',k'',t}$, and two dummy variables have to be added to the model. To enforce feasibility to constraint constraints (3.74) a dummy variable has to be added to its left hand side (LHS) with a big-M coefficient (for example $\sum_{\substack{s' \in B \\ s'' \in B}} \Phi(f_{s',s''})$). To enforce feasibility to constraints (3.62), a dummy variable has to be added to its LHS with coefficient 1. Both dummy variables have to be added to the objective function (3.1) with a high penalty.

The relaxed l-MNDP model including constraints (3.1) – (3.16), (3.59) – (3.63), (3.65), (3.69) – (3.71) and (3.73) – (3.77) is therefore used in the l-MNDP matheuristic, whose structure is depicted in Fig. 3.10. Steps (a) and (b) of Fig. 3.10 are analogous to steps (c) and (d) of Fig. 3.4, but we perform the dynamic generation of both feasible associations paths $r_p^{s,k,t}$ and synchronization paths $q_p^{k',k'',t}$. In both cases we formulated the pricing problem as a constrained shortest path problem, and we designed ad-hoc dynamic programming algorithms to solve it. At the end of the column generation, a fractional solution is available, and we resort to the hierarchical rounding to obtain an integer one. The order of rounding is the same as that used for s-MNDP. In fact, new continuous variables $q_p^{k',k''t}$ do not need to be rounded; the new binary variables $v_{s,k}$ are not rounded explicitly but are fixed by rounding propagation: when a z_k variable is fixed to zero, related $v_{s,k}$ variables are fixed to zero as well; when an association path variable $r_p^{s,k,t}$ is fixed to one, the related $v_{s,k}$ variable is fixed to one as well.

At the end of the rounding process, we check the compliance with the relaxed constraints on synchronization traffic (3.66) and (3.68) (Fig. 3.10-(c)). Given the AP-MEC facility associations, defined by variables $v_{s,k}$ with value 1, the computation of the related amount of user migration and hence the amount of synchronization traffic to route is straight. If the synchronization paths created until this step are enough to route this amount of synchronization traffic, a feasible solution is found and the optimization process ends successfully.

If not, further synchronization paths need to be created (Fig. 3.10-(d)): we fix MEC facilities locations variables z_k and AP-MEC facility association variables $v_{s,k}$, and the related amount of synchronization traffic is replaced in the right hand sides of constraints (3.74). All other variables ($r_p^{s,k,t}$, y_j and x_i) are unfixed and a new process of dynamic generation of path variables is executed (Fig. 3.10-(e)): differently from the previous step, we look only for AP-MEC facility association paths related to variables $v_{s,k}$ whose value was fixed to one. At the end of the iterative hierarchical rounding and column generation process (Fig. 3.10-(f)), if an

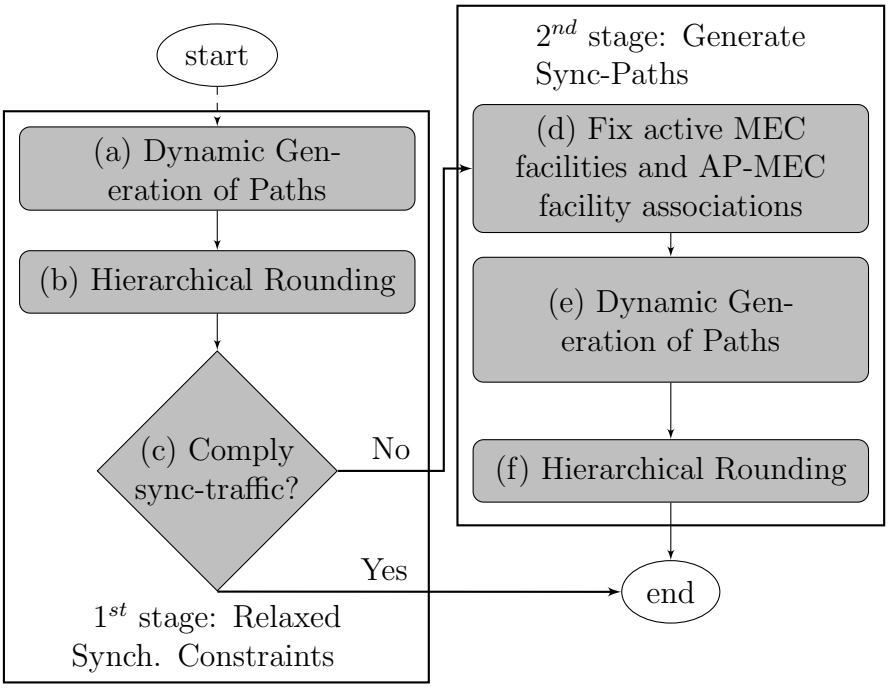


Figure 3.10: Overall Structure of l-MNDP Matheuristic

integer solution is found, then it is also feasible for the original problem; if not, our algorithm stops in a `FAIL` status. No very large scale local search is performed.

Dynamic generation of paths The pricing problem of dynamic generation of paths variables described for the s-MNDP matheuristic in Section 3.4 is now executed for each pair AP-time frame (s,t), together with the following dual variables: ν^t_{is} from (3.59); μ^t_{js} from (3.60); $\gamma^t_{k,s}$ from (3.61); ω^t_t from (3.62); π^t_k from (3.63); ρ^t_{mn} from (3.71); ξ^t_{sk} from (3.65); ϕ_{sk} from (3.77). The objective function (3.33) changes to:

$$\begin{aligned} \min_{s\in B, t\in T} \quad & -\omega_s + \sum_{i\in I} \bar{l}_{s,i}\nu^t_{i,s} + \sum_{j\in J} \bar{a}_j \mu^t_{j,s} \\ & + \sum_{k\in K} \bar{b}_k (n_h \pi^t_k + \gamma^t_{k,s} + \phi_{s,k} + \xi^t_{s,k}) + \\ & + \sum_{(m,n)\in E} \bar{l}_{m,n} \rho^t_{m,n} b_s \end{aligned} \tag{3.78}$$

while all constraints remain unchanged.

Moreover a dynamic generation of sync-path variables $q^{k',k'',t}_p$ is required. A pricing for each triplet of initial MEC facility, final MEC facility and time-frame (k',k'',t) is optimized. The initial model does not contain any sync-path variable and hence a dummy variable have to be added to the constraints (3.74) with a high penalty in objective function (3.1).

We consider that, given the MEC Network defined in Section 1.3, a synchronization path can traverse only core nodes, while the MEC facilities can be either core nodes or aggregation nodes. Hence to model synchronization paths we introduce the following pricing variables: $\bar{l}_{m,n} \in \{0,1\}\ \forall (m,n) \in E$ has value 1 if link (m,n) is in used in the synchronization path; $\bar{a}_j \in \{0,1\}\ \forall j \in N$ has value 1 if core facility j is in used in the synchronization path. Given duals: $\eta^t_{j,k',k''}$ from constraints (3.70); $\rho^t_{m,n}$ from constraints (3.71); $\theta^t_{k',k'}$ from constraints (3.74); and $\chi^t_{i,j}$ from constraints (3.73) the corresponding pricing model is defined as follows:

$$\min_{\substack{k',k''\in K|k'\neq k'' \\ t\in T}} \; -\theta^t_{k',k''} + \sum_{(m,n)\in E} \bar{l}_{m,n}(\rho^t_{m,n} + \chi^t_{m,n}) + \sum_{j\in J} \bar{a}_j \eta^t_{j,k',k''} \tag{3.79}$$

$$\text{s.t.} \sum_{(n,k')\in E|n\neq k'} \bar{l}_{k',n} = \sum_{(n,k'')\in E|n\neq k''} \bar{l}_{n,k''} = 1 \tag{3.80}$$

$$\sum_{(m,j)\in E} \bar{l}_{m,j} = \sum_{(j,m)\in E} \bar{l}_{j,m} \qquad \forall j \in J \mid j \neq k' \wedge j \neq k'' \tag{3.81}$$

$$\sum_{(m,n)\in E} \bar{l}_{m,n} \leq \bar{H}^Q \tag{3.82}$$

$$\sum_{(m,n)\in E} d_{m,n}\bar{l}_{m,n} \leq \bar{D}^Q \tag{3.83}$$

$$\sum_{(n,k')\in E} \bar{l}_{n,k'} = \sum_{(k'',n)\in E} \bar{l}_{k'',n} = 0 \tag{3.84}$$

$$\bar{l}_{m,n} \leq a_m \qquad \forall (m,n) \in E \mid m \neq k' \tag{3.85}$$

$$\bar{l}_{m,n} \leq a_n \qquad \forall (m,n) \in E \mid n \neq k'' \tag{3.86}$$

Pricing model (3.79) – (3.86) corresponds to a resource constrained shortest path problem: (3.79) defines the objective function to minimize; (3.80) enforce that a flow of one unit exit the source MEC facility k' and enter the target MEC facility k'', respectively; (3.81) enforce flow conservation in consecutive links; (3.82) and (3.83) enforce the limit on the number of hops and the total length of the synchronization path, respectively; (3.84) forbids that units of flows enters the source MEC facility k' and exits the target MEC facility k''; finally (3.85) and (3.86) links variables l_{mn} to variables a_j activating core nodes.

We solve this pricing problem with a dynamic programming approach, in the same fashion we solved the pricing model for the AP-MEC facility assignment path. We build a directed layered graph $G(N, A)$ with a layer for each possible hop H^Q. Let $\mathcal{L} = \left\{1, ..., \bar{H}^Q\right\}$ be the set of layers; the first and the last layer represent the initial and final MEC facility to connect with the synchronization path, while layers between the second and the $(H^Q - 1)$-th represent core nodes in between MEC facilities to connect. G is directed and layered, there is no possibility for cycles and all paths are elementary. We assign to each node i in layer $l \in \mathcal{L}$ of graph G states

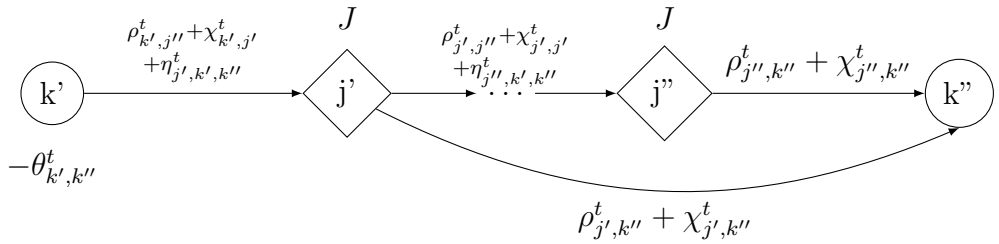


Figure 3.11: Layer Structure of the Dynamic Programming Algorithm for Synchronization Path Variables $q_p^{k',k'',t}$.

of the dynamic programming execution, while arcs in G represents state transitions. Different states, associated with the same node (i,l) correspond to different feasible synchronization paths p reaching i at layer l. They differ by: the subset of nodes S traversed by the current path and the length of the path d. States can be represented by a label $((i,l),S,d)$; to every label a cost is associated $c((i,l),S,d)$. Arcs in the graph, representing state transition, reflect feasible hops of a synchronization path in the network topology: from first layer (initial MEC facility) arcs connect to the second layer only (core nodes); from the second layer to the (H^Q-1)-th nodes are connected both to the next layer and to the last layer H^Q (the final MEC facility). Moreover, each arc has an associated cost, given by the network link costs and the device activation costs defined in the objective function of the pricer (3.79). The layer structure of the dynamic programming algorithm is sketched in Figure 3.11.

The main steps of our algorithm are the following:

Initialization The first layer is composed by a single node, that is the initial MEC facility k', its label is $((k',1),\{k'\},0)$ and its cost is initialized with the dual variable value $-\theta^t_{k',k''}$.

Propagation Nodes in intermediate layers from the second to the (H^Q-1)-th are connected only the nodes in the previous layer if and only if a feasible network links exists between the nodes they represent, that is a state with label $((i,l),S,d)$ is connected to a state in previous layer with label $((j,l-1),S,d)$ if and only if $d_{i,j}\leq\bar{d}$ Costs of these labels are extended according to the following rules:

$$c((j,l+1),S,d)=\min_{i\in S\setminus\{j\}|d_{i,j}<\bar{d}}\{c((i,l),S\setminus\{j\},d-d_{i,j})+\rho^t_{i,j}+\chi^t_{i,j}\}+\eta^t_{j,k',k''}$$
$$\forall j\in J,\forall l\in\{2..(\bar{H}^Q-2)\}\mid d<\bar{D}^Q\wedge j\neq k'\neq k'' \tag{3.87}$$

Last layer is composed by a single node, that is the target MEC facility k'' and its connected to every other nodes in the graph that can be reached by a feasible network link, and its cost is computed according to the following rule:

$$c((k'', \bar{H}^Q), S, d) = \min_{\substack{l \in \{1..(\bar{H}^Q - 1)\} \\ i \in S \setminus \{k''\} | d_{i,j} < \bar{d}}} \{c((i,l), S \setminus \{k''\}, d - d_{i,k''}) + \rho^t_{i,k''} + \chi^t_{i,k''}\} |\ d < \bar{D}^Q \tag{3.88}$$

Stopping The minimum cost path is given by:

$$\begin{aligned} &\min_S\ c((k'', \bar{H}^Q), S, d) \\ &\text{s.t. } d < \bar{D}^Q \end{aligned}$$

3.7 Computational Results

We implemented our algorithms in C++, using IBM ILOG CPLEX 12.6 [72] to solve MILP problems and their continuous relaxations. Our experiments ran on an Intel i7 4GHz workstation equipped with 32 GB of RAM. In order to test our algorithms we considered a synthetic datasets, drawn starting from the capacitated p-median test problems in [73]. The choice of the p-median is driven by the similarities with our problem: both are single source facility location problems, with a set of candidates locations equal to a subset of the clients, these latter having a demand to be satisfied. The parameter p of the number of facilities to open is just not considered in our problem, where a limit on the number of facilities to install is given by the fixed costs associated to the activation of a MEC facility in the objective function.

3.7.1 Synthetic Dataset

The synthetic dataset of the capacitated p-median test problem in [73] consists of 20 instances: 10 instances with 50 nodes and 10 instances with 100 nodes. Every problem instance contains the following data:

- a pair of coordinates for each node, that we used to compute distance parameters $d_{i,j}$ as euclidean distances;
- a demand for each node that we considered as VM demand parameter n_i;

- a single capacity value for facilities, that is our parameter C.

As first missing parameter we set $u_{i,j}$ to value 10000, that for sake of interpretability corresponds to a link with 1Gbps bandwidth. To be consistent with this value, we multiplied by 100 both the demand values n_i found in the p-median instance and the MEC capacity C.

Bandwidth demands b_i are computed as a percentage of the VM demand, generated as a uniform distributed random value between 95% and 105%, independently for every node. There is no connection between this computation and real correlation between number of users and bandwidth demand: we will perform extensive evaluation of real-world scenario in Chapter 4 using real number of users and the corresponding bandwidth demand. Parameter U, which has to be seen as a percentage, is set to 0.3: after preliminary analysis, higher values of U lead to loose link capacity constraints and low variability in the results. Further parametric analysis on the value of U in performed in Chapter 4. Following [74] and [75], we fix $l_i = 0.01$, $m_j = 0.1$, and $c_k = 1$. These costs can be seen as percentage costs, and the network costs can be estimated as about 10% of the overall MEC facility costs as suggested in [74]. Coefficients $\bar{D}$ and $\bar{D}^Q$ are computed respectively as the 75% and 85% of the radius of the considered area; to compute this latter, in turn, we consider the diameter of the considered area as the maximum distance between the nodes. $\bar{d}$ is set to be the 50% of $\bar{D}$. The number of hops $\bar{H}$ and $\bar{H}^Q$ are set to four and three, respectively: accordingly to [35], a wireless EC facilities approach outperforms a cloud-based approach when up to a maximum of four hops are considered. We set parameters of constraints (3.24)–(3.26), representing a lower bound on the number of facilities to activate, as follows:

- the minimum number of MEC facilities is computed as the minimum number of facilities needed to satisfy all VMs demands: $\underline{Z} = \lceil \sum_s n_s / C \rceil$;
- to set the minimum number of aggregation nodes $\underline{X}$ we solve a vertex covering

problem through the following MIP

$$\min \underline{X} = \sum_{i \in I} x_i \tag{3.89}$$

$$\text{s.t.} \sum_{i \in I} t_{s,i} \geq 1 \qquad \forall s \in B \tag{3.90}$$

$$t_{s,i} \leq x_i \qquad \forall s \in B, \forall i \in I \tag{3.91}$$

$$t_{s,i} = 0 \qquad \forall s \in B, \forall i \in I : d_{s,i} > \bar{d} \tag{3.92}$$

$$t_{s,i} \in \{0, 1\} \qquad \forall s \in B, \forall i \in I \tag{3.93}$$

$$x_i \in \{0, 1\} \qquad \forall s \in B \tag{3.94}$$

- the minimum number of cores $\underline{Y}$ is finally set equal to the minimum number of aggregation nodes $\underline{X}$;

After preliminary tests, including these constraints helped in strengthening the lower bound of the CG process, and lowering the execution time for the CG process; with respect to the final solution value their inclusion has not shown either particular worsening or improvement.

Rounding step threshold parameter ϵ is set to 10^{-3}. For the local search step, time limit parameter τ is set to 300 seconds and κ parameter is set to 30%.

For what concerns dynamic planning parameters, two time-frames are considered. The first time frame is composed by data directly taken from the dataset, and used also in the static planning scenario, while data for the second time frame are computed with the following process:

(i) the overall number of active users is set as a random fraction of the users in the first time frame, that in turn is computed as the sum of demand values $\sum_{i \in B} n_{i,1}$; this percentage is generated uniformly in the range $[97\%, 103\%]$;

(ii) user mobility is modelled through a $B \times B$ matrix: a cell (i, j) of the mobility matrix represents the amount of users moving from AP i to AP j in between the two time frames. A synthetic user mobility matrix is generated by computing for every cell of the matrix a value drawn from an exponential distribution with parameter $\lambda = 0.1$;

(iii) the mobility matrix is normalized twice: first normalization imposes that the percentage of users that don't move (i.e. the diagonal of the mobility matrix) is near the 20% of the total mobility, matching real world mobility that we present in Chapter 4.1, and the second normalization forces the overall number of users in the system to be the value computed at stage (i);

(iv) a data binning process is successively executed on user mobility, setting to 0 all values below the 50th percentile; that is, we only consider high mobility;

(v) the VM demands in the second time frame $n_{i,2}$ are given by the sum of the VM demands in first time frame and the mobility matrix.

Bandwidth demands $b_{i,2}$ of the second time frame are computed as in the static planning scenario. For simplicity, function $\Phi(x)$ is set as a linear function that consider the synchronization traffic to be the 30% of the average traffic generated by users:

$$\Phi(x) = x \cdot \frac{\sum_{i \in I} \sum_{t \in T} n_i^t}{\sum_{i \in I} \sum_{t \in T} b_i^t} \cdot 0.3 \tag{3.95}$$

The values of parameters $\bar{D}$ and $\bar{D}^Q$ are set for each instance independently. These parameters represent a percentage of the radius of the considered area, which in turn is computed as half of the maximum distance between nodes: starting from a percentage of 70% for both parameters, both values are increased by one percentage points until a feasible solution for the first phase of l-MNDP matheuristic is found.

VM file size (parameter G) and users' sojourn time (parameter T_w) used in constraints (3.73) were set respectively to 1GB and 5 hours.

Extensive analyses and motivation on the setting of these last two parameters are discussed for real world instances in Chapter 4, together with the parametric analysis of parameters $\bar{D}$, U and C.

3.7.2 s-MNDP Computational Results

In order to assess computational viability and effectiveness of the s-MNDP Matheuristic, we present three execution variants:

(i) **s-MNDP-noclust**: we execute the s-MNDP Matheuristic considering the complete set of nodes B, that is without clustering the network;

(ii) **s-MNDP-2layers**: we execute the s-MNDP Matheuristic considering the complete set of nodes $|B|$ but forcing each of them to be aggregation nodes, that is fixing all variables x_i to value 1 and generating only paths that consider a fixed assignment of an AP to itself as an aggregation node; in this way we want to assess the usefulness of considering a three layers (aggregation nodes, core nodes and MEC facilities) hierarchical network instead of a two layers (core nodes and MEC facilities) network;

(iii) **s-MNDP**: we execute the complete s-MNDP Matheuristic described in Section 3.4, including the initial clustering and the clustering update; we consider two initializations strategies, and two values for the α parameter.

s-MNDP-noclust

Initialized with CVC-round As first analysis we run the s-MNDP-noclust variant, that is s-MNDP without the clustering step (step (a) in Figure 3.4): we directly consider the complete network executing steps (b)–(e) only once. We initialize the restricted master problem with the solution given by our CVC-round.

In Table 3.4 we include details for every instance of our dataset (row-wise). For every instance we include the total time required by the execution of the algorithm in seconds (column 'total time') and we report several information for every stage of the matheuristic, respectively the column generation ('CG'), the hierarchical rounding and pricing process ('R&P') and the local search. For the column generation process we report the final lower bound ('LB'), the number of iterations ('no. iter'), the sum of the times spent in every master LP execution in seconds ('mast. time') and the sum of the times spent in every pricing subproblem execution in seconds ('pric. time'). For the hierarchical rounding and pricing we report: the number of CG iterations executed throughout the process ('no. iter.'); the sum of the times spent in every master LP ('mast. time') and in every pricing ('pric. time'); the time spent in final MIP execution in seconds ('MIP time'), to round the remaining free variables; and, finally, its final feasible integer solution value ('MIP sol.'). For the local search process we report the time spent by the generic MIP solver in seconds ('time') (we remind that we fix a maximum execution time of $\tau = 300$ seconds), the final feasible integer solution ('sol.') and percentage gap between the final solution and the lower bound found at the end of the CG ('gap'): this latter is computed as $(z^* - lb)/lb$ where z^* is the final solution value and lb is the lower bound value. At last we report the number of activated MEC facilities ('$|K|$'), core nodes ('$|J|$') and aggregations nodes ('$|I|$') of the final solution.

We first notice that computing time grows quickly as the size of the instance increases.

The CG has good convergence behaviour and no significant differences can be noticed in the number of iterations required to complete CG execution moving from 50 nodes to 100 nodes. The time spent for the execution of the master LPs is slightly higher than the time required for the pricing subproblems. In just one instance over

twenty the iterations needed to complete the computation are significantly higher and in this case the time required for the pricing subproblems computation is much higher than the time spent for the master LPs.

The hierarchical rounding and pricing process shows a higher variance in the number of iterations required to complete execution and in the final execution times. The time spent for the pricing subproblems is always higher than the time spent in the master LPs. While these latter are easier to solve as the rounding process goes on fixing more variables, master resolution process does not simplify. The final MIP execution to fix the last remaining free variables is always solved in few seconds. For instances with 100 nodes, the majority of the execution time in fact is spent in this rounding and pricing step.

The local search process, the execution of which we limit to a maximum of 300 seconds, on average improves the feasible solution found by the rounding and pricing process of about 14%.

The gap of the solution given by the local search and the lower bound found though column generation is on average about 25% for the 50 nodes cardinality instances and 8% for the 100 nodes cardinality instances. Since we set the facility activation costs so that a MEC facility has cost 1, a core nodes has cost 0.1 and an aggregation node has cost 0.01, by looking at the difference in absolute value between the final solution and the column generation lower bound we can measure how many MEC facilities can be saved in the most optimistic scenario. In seven cases, the solutions difference is less than 1 and hence our solutions are proved to use a globally minimum number of MEC facilities. In nine cases the difference is between 1 and 2, and hence no solution exists that reduce the number of MEC facilities by more than one unit. In the remaining four cases the difference is between 2 and 3, and hence optimality guarantees as less strong.

Initialized with CPLEX We run the s-MNDP-noclust variant initializing the restricted master problem with the solution of the simplified version of the problem (3.27)–(3.32) solved to optimality with the ILP generic solver of CPLEX instead of our CVC-round. We present the corresponding results in Table 3.5.

For every instance we include: the total time required by the execution of our algorithm in seconds (column 'total time'); the execution time in seconds required by CPLEX to solve (3.27)–(3.32) to optimality ('CPX init. time'); for the CG process we provide its final lower bound ('LB'), the number of CG iterations ('no iter.') and the total time in seconds ('total time'); for the rounding and pricing process we pro-

vide the number of CG iterations ('no iter.'), the total time in seconds ('total time') and the final feasible solution ('MIP sol.'); for the local search step we provide its execution time ('time'), the final best solution value ('sol'), the percentage gap between the final solution and the lower bound of the CG, and the corresponding number of MEC facilities, core nodes and aggregation nodes ('$|K|$', '$|J|$' and '$|I|$').

Finally, we provide the percentage gap between the final solution values and final execution times given by this scenario and those given by the s-MNDP-noclust, which is initialized with our CVC heuristic (in Table 3.4), respectively in column 'Δ^{pb}' and 'Δ^{t}'. Given s_1 the solution (resp. time) provided by this scenario and with s_2 the solution (resp. time) given in Table 3.4, the gap is computed as $(s_1 - s_2)/s_2$: hence, a positive gap is related to a higher final solution value (resp. execution time) for this scenario, while a negative gap is related to a lower value given by this scenario.

With respect to these percentage gaps we can notice that: the initialization with an optimality solution retrieves a better final solution 9 times over 20 and a worse solution 7 times over 20. However in 3 cases it allows to save a MEC facility activation, that represent the most expensive contribution to the final solution value. However, the initialization with an optimality solution leads to a longer execution time, with an average increase of about 20%. This increase is not related to the initial ILP run of CPLEX, which requires at maximum few seconds, while it seems to lie in a longer execution required by the rounding process.

We can state that an optimal initial solution of the simplified version of the s-MNDP can lead to improvements in the s-MNDP final solution with limited drawbacks in terms of higher execution times. However for bigger size networks, the initialization with an ILP execution of CPLEX may require substantial time.

s-MNDP-2layers

In Table 3.6 we include details of the s-MNDP-2layers. In this variant we fix every AP to be also an aggregation node, that is we fix the corresponding variables x_i and $t_{i,i}$ to value 1, and we force all generated paths to use the AP itself as aggregation node. The cost contribution of the aggregation nodes is not removed from the objective function. Similarly to the case of the s-MNDP-noclust variant, we also do not consider clustering step (step (a) in Figure 3.4), directly considering the complete network executing steps (b)–(e) only once without further changes. Given that the aggregation nodes are the least expensive facilities, we trade a possibly (not much) higher final cost for a lower execution time. The rationale is to evaluate the usefulness

given from considering a complete three-layer hierarchical network (as in s-MNDP-noclust variant) with respect to an easier two-layer network simply obtained through preprocessing.

The format is the same of Table 3.4, with two more columns: column Δ^{pb} reports the percentage gap between the final solution found by this variant with the final solution found by the s-MNDP-noclust variant (presented in Table 3.4); column Δ^{db} contains the percentage gap between the final solution found by this variant and the lower bound found by the CG in the s-MNDP-noclust variant (still presented in Table 3.4). Both gaps are computed as $(z^* - x)/x$ where z^* is the solution found by this variant and x is the solution (resp. lower bound of the CG) found by s-MNDP-noclust. The measure of Δ^{db} represents the optimality gap given by s-MNDP-2layers with respect to the original problem, while the value presented in column 'gap' is the optimality gap with respect to this simplified variant of the problem.

Restricting the model to a two-layer network decreases the required execution time on average of almost a factor 7 for the 50 nodes instances and 3 for the 100 nodes instances. In particular the pricing subproblems require few seconds overall, both for the column generation and for the rounding and pricing process, while considering a three-layer network the pricing execution is the most time-consuming process.

The lower execution times come at the price of a worsening of the final solution of about 11% for the 50 nodes instances and 12% for the 100 nodes instances, with respect to s-MNDP-noclust solutions. However, if we focus on the number of activated MEC facilities (column '$|K|$'), s-MNDP-2layers solutions activate the same number of MEC facilities of the s-MNDP-noclust solutions in fourteen cases, saving 2 MEC facilities in one case, 1 MEC facility in four cases, and activating one extra MEC facility in just one case. If we consider only MEC facilities costs and core nodes costs, s-MNDP-2layers retrieves a solution that is worse than s-MNDP-noclust solution on average of about 4%, but that is better in four cases.

s-MNDP Clustering Effects

As third analysis we consider the effects of the use of clustering of the complete s-MNDP matheuristic with respect to the use of the complete network (as in s-MNDP-noclust variant). In Table 3.7 we include details of the s-MNDP Matheuristic for every instance of out dataset (row-wise), considering:

- two different values for parameter α, that identifies the maximum number of clusters to consider (row-wise): in particular for the 50 nodes instances we

consider at maximum half of the original nodes as clusters, while for the 100 nodes instances we consider both one third and half of the original nodes.

- two different initialization methods, that is the Partitioning Around Medoids algorithm ('PAM Init.' columns) and the heuristic algorithm CVC-round ('CVC Init.' columns).

For every instance we limit the number of clustering update iterations to fifteen.

For each initialization variant we report the solution given by the heuristic algorithm CVC-round ('CVC sol.'), the required execution time in seconds ('time'), the iteration in which the best solution was found ('best sol. iter.'), the value of the first solution found ('first sol.'), the value of the best solution found ('best sol.'), the percentage difference between the first solution and the best solution ('Δ^s first sol.'), the percentage difference between the best solution and the solution given by CVC-round ('Δ^s CVC sol.') and, finally, the percentage difference between the best solution found by the clustered network and the best solution known, either found by s-MNDP-noclust or s-MNDP-2layers ('Δ^s best sol.'). These gaps are computed as $(z^* - x)/x$ where z^* is the solution found by the clustering process while x is either the first solution, or the CVC-round solution or the best known solution for the instance.

We can notice that the choice of the initial solution is crucial: the best solution retrieved by the variant initialized with the PAM Algorithm is worse than the solution given by the rounding heuristic in all cases but one, and it is worse than the best known solution on average of about 64%. On the contrary, by initializing the clustering with the rounding heuristic, s-MNDP retrieves a solution that on average is worse than the s-MNDP-noclust solution of about 16% and that can never be worse than the initial simplified solution. Actually in this last case the clustering variant improves the heuristic solution on average by about 9%, which means that it can be used as a means to improve an initial simplified solution.

Clustering allows to save execution times: on average the required time for the variant initialized with the rounding heuristic is twelve times lower than its s-MNDP-noclust variant counterpart; however the execution times was slightly higher in four cases for the 100 nodes instances clustered with half of the original nodes (i.e. $\alpha = 1/2$).

The percentage difference between the best solution and the first solution found is a measure of effectiveness for our clustering update strategy. For the variant initialized with CVC-round, in fifteen cases over thirty the best solution was the initial solution and hence the clustering update strategy was ineffective, while in

nine cases the best solution was given by the first clustering updating. Considering only the strictly positive improvements, the average improvement was of about 8%, but focused in the 50 nodes instances. For the solely 100 nodes instances the average improvement was on average about 3%.

For what concerns the choice of the α parameter (i.e. the choice of the maximum number of clusters to consider), we can notice that there are no striking differences in the quality of the best solution found: for the 100 nodes instances, in the variant initialized with CVC-round, using one third of the nodes retrieves a solution worse than the best solution on average of about 11% while using one half of the nodes, i.e. a bigger clustered network, retrieves a solution that is worse than the best solution on average of about 12%. The main difference lies in the required execution times: using one half of the original nodes requires on average thirty times more computation time than using one third of nodes.

3.7.3 l-MNDP Computational Results

We present computational results analysis of the l-MNDP Matheuristic and, successively, we compare solutions given by the l-MNDP model with those given by the s-MNDP-noclust, considering a measure of users' SLA violation. We present results of the l-MNDP Matheuristic for instances with 50 nodes, as we were not able to find solutions for the instances with 100 nodes.

In Table 3.8 we report details of the l-MNDP Matheuristic for the ten instances with 50 nodes of our dataset (row-wise). The first three columns identify the instance: in particular we report the number of nodes ('$|B|$'), the maximum length of an AP-MEC facility path and the maximum length of a synchronization path (columns '$\bar{D}$' and '$\bar{D}^Q$', respectively), defined as percentage of the radius of the considered area. Then we report the required execution time in seconds (column 'total time') and several information regarding the first and the second stage of our Matheuristic. For the first stage we report the lower bound given by the column generation (column 'LB CG'), the number of column generation iterations of column generation (column 'no. iter.'), the lower bound at the end of the hierarchical rounding and pricing process (column 'LB R&P'), the time in seconds required by all master LPs executions (column 'master time'), the time in seconds required by the pricing subproblems separately for the AP-MEC facility assignment path variables (column 'pricing time r_p^{skt}') and for the synchronization path variables (column 'pricing time $q_p^{k'k''t}$'); finally we report the final solution of the first stage (column 'sol.'). For the second stage we report the time in seconds required by the master LPs and the two pricing sub-

problems, together with the final feasible solution (column 'sol.') and the number of activated MEC facility, core nodes and aggregation nodes (columns '$|K|$', '$|J|$' and '$|I|$').

We were not able to find a feasible solution to the third instance, that we label with **NF** (i.e. not found).

We can notice that the execution time rises from the few minutes required for the s-MNDP variant to a maximum of ten hours. In particular we can notice that the majority of time is spent in the first phase of the algorithm, and in particular in the pricing subproblems for the AP-MEC facility assignment path variables and, to a lesser extent, for the master LPs. The pricing subproblems for the synchronization path variables, on the contrary, requires negligible time, as well as the overall second stage. The number of CG iteration required in the first stage is similar to the number of iteration required by the s-MNDP Matheuristic CG, and hence the complexity lies in the rounding and pricing process.

The gap of the final solution with the CG lower bound of the first stage is on average very large. A possible explanation is highlighted by the following experiment.

Static Planning vs Dynamic Planning

In Table 3.9 we report the solution of the s-MNDP-noclust variant for the ten instances with 50 nodes using the same parameter $\bar{D}$ that we use for the l-MNDP variant, in order for a fair comparison of the resulting solutions. The format of the table is the same of the previously presented Table 3.4, with an additional column representing the number of users experiencing a violation of their SLA (column '% violated SLA').

In order to compare Dynamic Planning Variant with Static Planning Variant from the users perspective, we introduce a further fitness measure, that is the percentage of users with violated SLA after migration. We remark that l-MNDP guarantees by design 0% of users with violated SLA, while, on the contrary, s-MNDP does not give any a-priori guarantee. In order to compute the users migrations from a solution given by the Static Planning s-MNDP Matheuristic, a post-processing phase is needed. Given a solution $\bar{S}$ resulting from s-MNDP, we know by the parameter $f_{s',s''}$ that users migrate in the planning horizon between APs s' and s''; at the same time, we know, by values of variables $r_p^{s,k}$ in $\bar{S}$, which are those MEC facilities k' and k'' servicing s' and s'', respectively. If it is possible to construct, after the optimization

process, a feasible synchronization path between k' and k'' respecting constraints (3.67), then we say that the SLA of those $f_{s',s''}$ users are respected; otherwise we say that they are violated. Indeed, if a feasible synchronization path cannot be established, a user may perceive a latency during migrations that exceeds his SLA.

We can notice that the s-MNDP always leads to impressive savings in the execution time: few seconds are required with this relaxed instances instead of several hours of computation. Moreover also the final cost is always better than the cost given by the l-MNDP variant, and is very close to the CG lower bound. l-MNDP solutions requires at maximum two extra MEC facilities to be enabled with respect to the s-MNDP solution, but many extra core nodes and aggregation nodes that are used for the synchronizations path to be established.

However considering the Dynamic Planning version is crucial to comply with migrating users' SLA: in s-MNDP solutions, on average, about the 40% of users experience violation of their SLA. For the fourth instance no user has violated SLA, but this outcome is likely to be fortuitous.

Finally, if we compare the lower bounds of the CG and of the rounding process of the s-MNDP and those given by the first stage of the l-MNDP in Table 3.8, we can notice little difference in values while we could expect a substantial difference given that l-MNDP is more constrained. Hence we can assume that the high gap of the final feasible solution of l-MNDP with respect to its CG lower bound is given by the weakness of the latter.

The introduction of strengthening inequalities might be pertinent in applications requiring accurate quality estimates.

3.8 Conclusions

In this chapter a new strategic network design NP-Hard problem, the MEC Network Design Problem (MNDP), has been presented. We formally defined the problem, including two planning model variations: (i) considering a static status of the network, unaware of variations during the planning horizon, and (ii) considering a dynamic network, including load variations and mobility of users and virtual machines, encoding three different virtual machine mobility technologies.

We provide link-path mixed integer linear programming formulations including a polynomial number of variables to represent location and design decisions, and an exponential number of them to encode routing ones. In order to achieve high

quality solutions in reasonable time, we provide mathematical programming based matheuristics for the two variants of MNDP, combining column generation, iterative rounding, very large scale neighborhood local search and problem reduction.

We experimented on a dataset of twenty small size instances involving 50 and 100 APs adapted from the facility location literature, and drawing insights to use our approach as a tool for a predictive analysis.

In case it is crucial to comply with users' SLA also considering users' migration, the l-MNDP variant guarantees such a compliance and has to be used. As a drawback, l-MNDP can be used only for small size networks to retrieve solution in reasonable time, and it retrieves higher costs solution than the s-MNDP case. On the contrary, if SLA can be defined as *static* with respect to users' actual location in time, s-MNDP variant can be used.

In case s-MNDP variant is used, we identified two discriminants for its use: the size of the network and the availability of time. s-MNDP can tackle in reasonable time instances with up to 100 nodes; best results in terms of minimum cost solution are given by the s-MNDP-noclust variant which requires up to few hours of execution times. To save execution time, the s-MNDP-2layers variant can be used, requiring up to few minutes of execution: the worsening in terms of solution value is reasonable (up to 11%) but focusing on the solely MEC facility and core node locations the worsening is further limited (up to 4%).

For bigger size networks, the most appealing option seems to be the complete s-MNDP variant, which includes the clustering of the initial network. In this case is crucial the choice of the initial clustering: if a good heuristic solution is provided, the s-MNDP can be used to improve it. If a good initial solution is not available, the clustered network should be has big as possible with respect to the original size to have good final solution. The time required by s-MNDP is comparable to the time required by s-MNDP-2layers with a network of the same size of the clustered network.

As to future work from the algorithmic point of view: with respect to the static planning variant, it could be interesting to improve the restart strategy for the clustering approach to improve the effectiveness of the corresponding algorithm variant; with respect to the dynamic planning variant, a different approach to restore integrality on the fractional solution to replace the rounding process could be devised to improve the final solution value and the lower bound.

			CG				R&P					local search					
$\|B\|$	inst. ID	total time	LB	no. iter.	mast. time	pric. time	no. iter.	mast. time	pric. time	MIP time	MIP sol.	time	sol.	gap	$\|K\|$	$\|J\|$	$\|I\|$
50	1	152	5.388	66	6	10	117	3	19	0	6.60	114	6.60	22.49%	6	5	10
	2	279	5.536	53	30	9	482	31	78	1	7.94	126	6.62	19.58%	6	5	12
	3	170	6.233	62	13	12	314	10	57	0	7.74	78	6.75	8.29%	6	6	15
	4	417	8.509	69	18	11	422	20	64	0	10.03	300	9.91	16.46%	9	8	11
	5	179	5.389	43	10	9	319	23	41	0	8.91	95	7.52	39.54%	7	4	12
	6	364	5.474	55	10	10	237	8	34	0	8.78	300	7.75	41.58%	7	6	15
	7	405	5.391	42	10	11	331	33	50	1	8.05	300	6.61	22.61%	6	5	11
	8	86	5.716	39	13	7	230	20	29	0	8.63	16	6.62	15.82%	6	5	12
	9	440	5.375	53	17	7	307	52	54	0	8.04	300	6.60	22.79%	6	5	10
	10	389	6.858	57	15	7	353	14	51	1	11.38	300	9.73	41.88%	9	6	13
100	11	4366	9.393	66	175	156	1040	1621	2011	5	12.03	301	10.61	12.96%	10	5	11
	12	4001	9.393	180	256	347	1080	690	2389	1	11.72	300	10.61	12.96%	10	5	11
	13	2228	9.490	78	123	153	629	480	1161	1	12.83	300	9.61	1.26%	9	5	11
	14	2553	9.390	33	118	82	721	626	1411	1	12.94	300	9.51	1.28%	9	4	11
	15	4161	9.490	42	118	75	1762	516	3127	1	13.13	300	10.51	10.75%	10	4	11
	16	5376	9.391	85	238	171	1322	1990	2662	1	11.81	300	9.62	2.44%	9	5	12
	17	7032	9.398	55	194	134	1498	2039	2815	2	12.00	300	11.73	24.81%	11	6	13
	18	4857	9.500	20	99	46	1192	1709	2681	1	13.06	300	10.72	12.84%	10	6	12
	19	3157	10.400	23	58	48	1292	288	2444	1	11.92	300	10.63	2.21%	10	5	13
	20	1351	10.400	14	13	28	521	57	944	0	11.94	301	10.62	2.12%	10	5	12

Table 3.4: s-MNDP-noclust - Computational Results

				CG			R&P			local search						w.r.t. Table 3.4	
$\|B\|$	inst. ID	total time	CPX init. time	LB	no. iter.	total time	no. iter.	total time	MIP sol.	time	sol.	gap	$\|K\|$	$\|J\|$	$\|I\|$	Δ^{pb}	Δ^{t}
50	1	154	0	5.388	66	16	117	139	6.60	114	6.60	22.50%	6	5	10	0.00%	1.32%
	2	389	0	5.536	54	39	462	591	8.03	233	6.61	19.41%	6	5	11	-0.15%	39.43%
	3	211	0	6.233	59	25	267	381	7.75	127	6.75	8.29%	6	6	15	0.00%	24.12%
	4	432	0	8.509	69	29	422	506	10.03	300	9.91	16.47%	9	8	11	0.00%	3.60%
	5	318	0	5.388	47	19	378	383	8.82	221	7.6	41.06%	7	5	10	1.06%	77.65%
	6	363	0	5.474	53	20	244	279	9.09	300	7.74	41.40%	7	6	14	-0.13%	-0.27%
	7	391	0	5.391	36	21	354	414	7.81	300	6.63	22.98%	6	5	13	0.30%	-3.46%
	8	154	0	5.716	34	20	211	279	8.05	96	6.63	15.99%	6	5	13	0.15%	79.07%
	9	445	0	5.375	53	24	307	413	8.13	300	6.52	21.31%	6	4	12	-1.21%	1.14%
	10	393	0	6.858	59	22	361	418	11.96	300	8.85	29.05%	8	7	15	-9.04%	1.03%
100	11	5543	2	9.393	67	331	2017	4672	12.25	300	10.73	14.23%	10	6	13	1.13%	26.96%
	12	3584	0	9.394	73	603	1258	4159	10.73	300	10.61	12.95%	10	5	11	0.00%	-10.42%
	13	2292	1	9.490	75	276	788	2270	9.59	300	9.59	1.05%	9	5	9	-0.21%	2.87%
	14	5468	0	9.390	43	200	1600	2758	11.04	300	9.6	2.24%	9	5	10	0.95%	114.18%
	15	3462	3	9.491	79	193	1423	5405	11.84	300	9.77	2.94%	9	6	17	-7.04%	-16.80%
	16	10120	1	9.391	79	409	1537	5974	10.7	300	9.72	3.51%	9	6	12	1.04%	88.24%
	17	6961	4	9.398	45	328	1217	6352	12.36	301	10.73	14.17%	10	6	13	-8.53%	-1.01%
	18	3142	0	9.500	20	145	1166	5582	12.14	300	10.73	12.95%	10	6	13	0.09%	-35.31%
	19	3492	2	10.400	42	106	1400	4024	12.96	300	10.62	2.12%	10	5	12	-0.09%	10.61%
	20	1363	2	10.400	13	41	486	1522	11.81	300	10.52	1.15%	10	4	12	-0.94%	0.89%

Table 3.5: s-MNDP-noclust - Init. with CPLEX - Computational Results

			CG				R&P					local search					w.r.t. Table 3.4	
$\|B\|$	inst. ID	total time	LB	no. iter.	mast. time	pric. time	no. iter.	mast. time	pric. time	MIP time	MIP sol.	time	sol.	gap	$\|K\|$	$\|J\|$	Δ^{pb}	Δ^{db}
50	1	10	6.15	51	2	0	134	2	2	0	7.8	3	7.3	18.78%	6	8	10.61%	35.49%
	2	61	6.31	48	8	1	275	14	2	0	8	33	7.6	20.41%	6	11	14.80%	37.28%
	3	43	7.11	44	5	0	205	8	1	1	10.4	26	8.8	23.84%	7	13	30.37%	41.18%
	4	118	9.25	51	5	0	271	10	4	0	11.9	96	10.1	9.15%	8	16	1.92%	18.70%
	5	19	6.16	38	4	0	158	2	2	1	8.1	7	6.9	12.09%	5	14	-8.24%	28.04%
	6	15	6.52	45	3	0	158	2	1	1	9.4	5	8.1	24.29%	6	16	4.52%	47.97%
	7	141	6.18	43	5	0	185	9	1	1	8.2	121	7.6	23.06%	6	11	14.98%	40.98%
	8	34	6.59	47	5	0	149	6	0	1	9.1	20	8.3	26.04%	6	18	25.38%	45.21%
	9	37	5.98	43	7	1	184	8	3	1	8.1	14	7.3	22.07%	6	8	10.61%	35.81%
	10	146	7.68	40	5	0	137	6	0	0	10.7	133	10.5	36.75%	9	10	7.91%	53.11%
100	11	879	10.50	101	182	1	632	349	8	2	12.6	300	12	14.31%	10	10	13.10%	27.75%
	12	924	10.48	171	273	1	823	304	22	2	12.5	300	11.2	6.85%	9	12	5.56%	19.24%
	13	852	10.46	79	175	1	799	331	11	1	12.2	300	11.1	6.09%	9	11	15.50%	16.97%
	14	773	10.47	108	210	5	660	216	12	1	12.1	301	11	5.08%	9	10	15.67%	17.15%
	15	847	10.48	103	228	3	733	277	8	5	12.8	300	12.1	15.50%	10	11	15.13%	27.50%
	16	1051	10.49	169	289	4	772	406	16	2	12.3	300	11.1	5.84%	9	11	15.38%	18.20%
	17	989	10.50	100	243	0	491	392	8	1	13	301	12.4	18.11%	10	14	5.71%	31.94%
	18	898	10.52	101	235	2	588	312	9	2	12.4	300	12.2	16.02%	10	12	13.81%	28.42%
	19	881	11.47	113	200	3	985	326	16	1	12.3	300	12.2	6.37%	10	12	14.77%	17.31%
	20	638	11.48	117	126	2	686	169	10	1	13.4	301	12.1	5.44%	10	11	13.94%	16.35%

Table 3.6: s-MNDP-2layers - Computational Results

				PAM Init.							CVC Init.						
$\|B\|$	α	inst. ID	CVC sol.	time	best sol. iter.	first sol.	best sol.	Δ^s first sol.	Δ^s CVC sol.	Δ^s best sol.	time	best sol. iter.	first sol.	best sol.	Δ^s first sol.	Δ^s CVC sol.	Δ^s best sol.
50	$\frac{1}{2}$	1	8.88	10	6	10.05	9.15	-8.96%	3.04%	38.64%	7	1	8.88	6.95	-21.73%	-21.73%	5.30%
		2	9.99	13	3	11.25	11.25	0.00%	12.61%	70.20%	34	1	9.99	7.95	-20.42%	-20.42%	20.27%
		3	11.1	9	6	12.25	10.15	-17.14%	-8.56%	50.37%	19	4	10	9.25	-7.50%	-16.67%	37.04%
		4	11.1	21	7	13.15	12.25	-6.84%	10.36%	23.61%	25	1	11.1	10.25	-7.66%	-7.66%	3.43%
		5	9.99	12	9	11.25	10.25	-8.89%	2.60%	48.55%	32	9	9.89	9.15	-7.48%	-8.41%	32.61%
		6	12.21	6	1	14.35	14.25	-0.70%	16.71%	84.11%	5	12	12.21	10.25	-16.05%	-16.05%	32.43%
		7	9.99	11	4	12.15	11.15	-8.23%	11.61%	68.68%	53	1	9.89	9.05	-8.49%	-9.41%	36.91%
		8	9.99	9	0	11.25	11.25	0.00%	12.61%	69.94%	26	0	9.79	9.79	0.00%	-2.00%	47.89%
		9	7.77	16	9	10.15	8.05	-20.69%	3.60%	23.47%	53	0	7.67	7.67	0.00%	-1.29%	17.64%
		10	9.99	9	7	12.25	12.25	0.00%	22.62%	38.42%	17	0	9.99	9.99	0.00%	0.00%	12.88%
100	$\frac{1}{3}$	11	13.32	44	1	20.33	20.23	-0.49%	51.88%	90.67%	200	1	11.82	11.13	-5.84%	-16.44%	4.90%
		12	11.1	47	7	17.33	16.33	-5.77%	47.12%	53.91%	90	0	12.72	12.72	0.00%	14.59%	19.89%
		13	12.21	51	1	16.13	16.13	0.00%	32.10%	68.20%	294	0	10.81	10.81	0.00%	-11.47%	12.72%
		14	12.21	38	0	19.13	19.13	0.00%	56.67%	101.16%	203	0	10.71	10.71	0.00%	-12.29%	12.62%
		15	13.32	47	5	21.13	20.13	-4.73%	51.13%	106.04%	237	1	11.03	11.03	0.00%	-17.19%	12.90%
		16	11.1	39	12	21.33	19.33	-9.38%	74.14%	100.94%	1758	1	10.23	10.23	0.00%	-7.84%	6.34%
		17	13.32	54	0	21.03	21.03	0.00%	57.88%	95.99%	188	0	11.92	11.92	0.00%	-10.51%	11.09%
		18	13.32	35	3	19.33	18.33	-5.17%	37.61%	70.99%	151	2	12.23	12.23	0.00%	-8.18%	14.09%
		19	13.32	52	0	21.13	21.13	0.00%	58.63%	98.96%	118	10	12.13	11.23	-7.42%	-15.69%	5.74%
		20	12.21	47	0	22.13	22.13	0.00%	81.24%	110.36%	62	1	12.03	12.03	0.00%	-1.47%	14.35%
	$\frac{1}{2}$	11	13.32	652	5	15.6	13.7	-12.18%	2.85%	29.12%	2680	1	11.82	11.5	-2.71%	-13.66%	8.39%
		12	11.1	545	14	13	12.7	-2.31%	14.41%	19.70%	2180	10	12.72	12.4	-2.52%	11.71%	16.87%
		13	12.21	499	2	15.5	15.5	0.00%	26.95%	61.63%	3770	13	10.81	10.6	-1.94%	-13.19%	10.53%
		14	12.21	1334	0	14.7	14.7	0.00%	20.39%	54.57%	3624	0	10.7	10.7	0.00%	-12.37%	12.51%
		15	13.32	389	13	14.7	14.4	-2.04%	8.11%	47.39%	3674	2	10.9	10.8	-0.92%	-18.92%	10.54%
		16	11.1	421	1	17.5	17.5	0.00%	57.66%	81.91%	3807	8	10.9	10.7	-1.83%	-3.60%	11.23%
		17	13.32	582	10	16.9	14.7	-13.02%	10.36%	37.00%	3747	0	11.92	11.92	0.00%	-10.51%	11.09%
		18	13.32	485	0	14.9	14.9	0.00%	11.86%	38.99%	4272	1	12.4	12.4	0.00%	-6.91%	15.67%
		19	13.32	1109	4	16.5	16.5	0.00%	23.87%	55.37%	4177	14	11.5	11.4	-0.87%	-14.41%	7.34%
		20	12.21	345	9	19.5	17.7	-9.23%	44.96%	68.25%	1665	1	12.21	12.21	0.00%	0.00%	16.06%

Table 3.7: s-MNDP - Clustered Network Computational Results

					First Stage							Second Stage						
$\|B\|$	inst. ID	$\bar{D}$	$\bar{D}^Q$	total time	LB CG	no. iter.	LB R&P	mast. time	pric. time r_p^{skt}	pric. time $q_p^{k'k''t}$	sol.	mast. time	pric. time r_p^{skt}	pric. time $q_p^{k'k''t}$	sol.	$\|K\|$	$\|J\|$	$\|I\|$
50	1	0.8	0.9	656	5.395	119	5.650	115	377	80	6.15	11	6	1	6.39	5	12	19
	2	0.8	0.9	11398	5.265	64	7.390	5855	5212	111	7.59	8	27	3	8.23	7	11	13
	3	0.9	1	20102	5.266	66	8.630	5561	14073	173	**NF**	-	-	-	**NF**	-	-	-
	4	0.9	1	4088	5.257	27	5.360	1743	2198	41	5.46	9	20	2	6.11	5	10	11
	5	0.9	1	2777	5.265	60	6.390	1133	1386	72	6.69	12	17	1	7.21	6	11	11
	6	0.8	0.9	3801	5.385	93	6.740	2963	543	54	7.14	10	8	0	7.37	6	12	17
	7	0.9	1	37148	5.258	24	6.433	18612	17908	170	6.79	19	51	0	7.33	6	12	13
	8	0.8	0.9	21414	5.262	93	7.490	8796	12231	224	7.79	8	23	1	8.33	7	12	13
	9	0.8	0.9	2882	5.271	64	6.390	1120	1625	49	6.49	12	14	1	7.66	6	15	16
	10	0.8	0.9	1163	5.292	111	6.510	671	346	43	6.71	18	5	3	7.34	6	12	14

Table 3.8: l-MNDP - Computational Results

			CG				R&P						local search						
$\|B\|$	inst. ID	total time	LB	no. iter.	mast. time	pric. time	LB	no. iter.	mast. time	pric. time	MIP time	MIP sol.	gap	time	sol.	$\|K\|$	$\|J\|$	$\|I\|$	% violated SLA
50	1	1	5.35	3	0	0	5.350	11	0	0	0	5.35	0%	0	5.35	5	3	5	62.38%
	2	6	5.25	12	1	0	5.350	49	0	2	0	5.35	0%	2	5.35	5	3	5	75.21%
	3	7	5.25	5	1	0	5.350	94	1	2	0	5.35	0%	1	5.35	5	3	5	7.01%
	4	7	5.25	3	0	0	5.350	109	0	6	0	5.35	0%	0	5.25	5	2	5	0.00%
	5	11	5.25	13	0	1	6.732	121	1	5	1	7.37	0%	3	5.25	5	2	5	41.32%
	6	20	5.35	19	1	0	6.470	349	5	10	0	6.47	0%	3	5.36	5	3	6	82.55%
	7	24	5.25	12	1	1	5.350	252	6	7	0	5.35	0%	9	5.25	5	2	5	9.83%
	8	16	5.25	19	1	1	6.057	226	4	7	1	6.36	0%	2	5.35	5	3	5	53.90%
	9	2	5.25	6	0	0	5.250	11	0	1	0	5.25	0%	1	5.25	5	2	5	25.71%
	10	63	5.25	22	4	0	6.470	271	17	9	0	6.47	0%	32	5.47	5	4	7	59.70%

Table 3.9: s-MNDP-noclust – l-MNDP Comparison

Chapter 4

MEC Network Design Optimization: Data Analytics and Best Practices

In this Chapter we tackle the full MNDP problem from the application point of view. That is, we discuss and motivate the application details, and we perform data analytics on real 4G cellular network data-sets from the Île-de-France region, provided by Orange Mobile. Optimization algorithms presented in Chapter 3 are used as tools for these analyses. We draw best practices, comparing static and dynamic models, performing sensitivity analyses on several model parameters and evaluating different virtual machine orchestration and mobility policies.

We bring novel and original insights on the planning of MEC Networks. By performing extensive simulations, we show the trade-off that can be achieved by means of the two design cases and the impact of user mobility on the MEC network: as few as 13 to 26 MEC Facilities can be planned for 180 thousands of users while requiring tight delay guarantees. We show that there is a sensible gain in the number of users with respected SLA, up to 20%, by including user and VM mobility in the network planning. We do also qualify the eligibility of two different VM mobility strategies, namely VM Bulk and Live Migrations, for two reference MEC services differing in the level of required latency and memory characteristics: augmented-reality and remote desktop. We report empirical distributions of the dataset features in order to allow the reproducibility of our results.

Main results were presented in [10] and published in [11].

4.1 Real-World Dataset

In order to ground simulations of our MEC Network Design Problem on real data, we used a dataset collected by Orange Mobile in Île-de-France region, in the frame of the ABCD project [76]. The dataset comes from network management tickets, containing UE data exchange information aggregated in 6 minutes periods. User session is assigned to the cell identifier of the last used antenna. Data are recorded on a per-user basis and cover a large metropolitan area network, including urban, peri-urban and rural areas. We had access to data of a single 24-hour period, originated by 606 LTE 4G APs in an area of 931 km^2, with a density of about 0.65 APs per km^2. The number of users served by the considered APs is $\sim$ 180 thousands, generating an overall daily traffic of $11TB$.

In Table 4.1 we present a partial schema of the dataset from which the data were retrieved: a record in the dataset contains the amount of bytes and packets exchanged by a user with an access point in a six-minute period; bytes are classified by the operator depending by the type of application to which they belong. We took into consideration only information coming from 4G data exchange; moreover, for every period we summed demands of all users and of all different application types, as single-user and single-service demands are not used in our model. No further manipulation were needed to retrieve the data.

The only available information about the network topology is the location of the access points, in terms of latitude and longitude: no information about the back-hauling network were available to us, as well as no information about link capacity were available.

With respect to MDNP model presented in Chapter 3, in the following we present the estimation of its parameters starting from this real-world dataset.

4.1.1 Estimation of Model Parameters

The number of users connected to every APs $s \in B$ (coefficient n_s) and the corresponding demand in terms of bandwidth (b_s) are drawn by direct queries from the dataset: given all six-minutes time-slots of the chosen planning horizon and an AP, the number of users connected is retrieved counting the number of unique users in all records associated to the AP in the time-slots, while the corresponding demand is given by the sum of demands of all applications in all records associated to the AP in the time-slot.

Field Name	Format	Unit	Description
Period	Integer		Period number indicating the 6 minutes period in the 24 hours timeslot of the file
IdAgent	String		Agent Identifier (anonymized)
AccessType	Integer		Radio Access Type for MOBILE configuration (0 = 4G, 1 = 3G, 2 = 2G)
nByte	Integer	Bytes	Total amount of data for downlink OR uplink
Byte_App	Integer	Bytes	Amount of data for an application category
Pack_App	Integer		Number of packets for an application category
LocInfo	String		User Location Information (encoding longitude and latitude of AP)
Timestamp	String		Timestamp Format 'YYYY-MM-DD hh:mm:ss' (extended information of period)

Table 4.1: Mobile Operator Dataset Schema

Following [74] and [75] we set the activation costs of aggregation node (l_i) to value 0.01, those of core nodes (m_j) to value 0.1, and those of MEC facilities (c_k) to value 1.0, which can be seen as percentages. That is, we give maximum priority to the minimization of MEC facility costs, assuming to be the most relevant, and, as suggested in [74], we estimate the network costs to be as about 10% of the overall cloud data center costs.

With respect to the operator physical topology, we had access only to the position of the APs, while the underlying backhauling physical topology were not available to us, as well as the actual link capacity. However, the knowledge of real world information about these parameter is not critical for our experiments, given that our model aims exactly in designing a new MEC network from scratch, and provides a complete MEC network topology as a by-product. We recall that the topological rules with which we model a MEC network are described in Chapter 1.2 and encoded in set of constraints (3.2) – (3.16). Therefor, as values for the distances between two nodes ($d_{i,j}$) we take the euclidean distances between each pair of APs $i, j \in B$.

In the same way, network link capacities that we could have found in the dataset are designed for the current 4G mobile paradigm, and they are not suited for our application. Hence, we assume a fixed bandwidth capacities $u_{i,j}$ of each link $(i, j) \in E$ of $10Gbps$ in both hierarchical levels, that represents a plausible value in the current convergence towards packet-switching carrier-grade solutions, where bit-rates for pseudo-cables links is set to giga-Ethernet granularities (typically 1 or 10 Gbps).

We limit the paths to four hops ($\bar{H} = 4$), accordingly to [35], where wireless EC facilities approach are shown to outperform cloud-based approach when up to a maximum of four hops are considered.

Instead of choosing a particular setting for the capacity of MEC facility C, the maximum link capacity usage U and the maximum length of a AP-MEC facility path $\bar{D}$, we perform a parametric analysis on them. Moreover, we set the maximum link length $\bar{d}$, that is required for the feasibility of the assignment paths of AP to MEC facilities defined in (3.17), by observing the position of the APs and their distances.

These analysis are presented in following Sections 4.2 and 4.3.

User Mobility Patterns Individual user mobility patterns cannot be obtained for confidentiality reasons. Furthermore, allowing migrations even when an AP is visited infrequently would have a strong negative impact on the overall network load, without significantly improving user experience. Trying to cope with this issue we perform binning on data: for each user we consider his two more frequently visited APs during the planning horizon. We restrict to consider possible migrations only between these two locations representing, for instance, home and work places of users, which, following [77, 78], dominate human mobility. Technically, this data is obtained by creating groups of users and obfuscating individual identifiers. Other options may be considered, in absence of such data, to estimate mostly visited places [79].

Summarizing, for each pair of APs s' and s'' let $f_{s',s''}$ be the number of users having s' and s'' as the most frequently visited APs; this parameter is general and can be used with any number of frequently visited locations other than two, without changes. In order to further characterize such user mobility patterns, and to allow third parties to reproduce adequately our findings, we report in Fig. 4.1 the cumulative distribution function of the distances traveled by users while migrating. We observe that about 20% of users do not move at all during the day and that almost all users move less than the radius of the considered region (i.e. $15km$). Moreover, in Fig. 4.2, we present an histogram reporting on the x axis ranges for number of users. For each range $[x', x'']$ on the x axis, a bar represents the number of pairs of APs s' and s'' having $f_{s',s''} \in [x', x'']$. We can conclude that: (i) the majority of paths are covered by a small number of users, and (ii) about 72% of the possible pairs of APs never appear as most frequent for any user. That is, the mobility is concentrated along a few frequently chosen paths, matching our intuition.

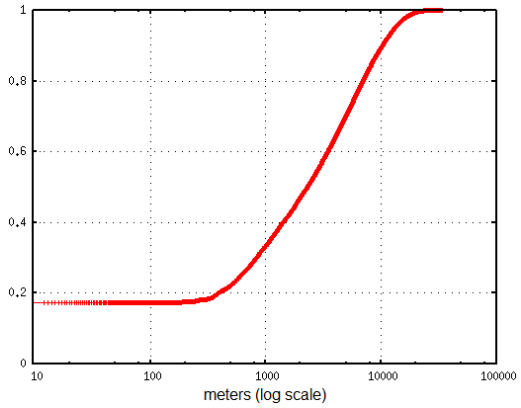


Figure 4.1: Cumulative Distribution Function of traveled distances of user flights.

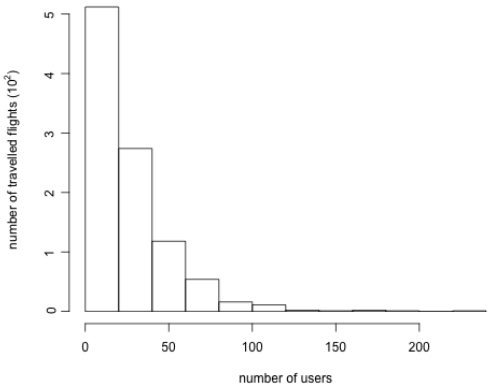


Figure 4.2: Histogram of no. of users covering same flight.

4.2 Experimental Setup

As reported in Chapter 3, we found adaptations of heuristics from the literature to be unable to produce accurate results; this was one of the main motivations to build our models. Still, our path-based formulations offer great modeling flexibility and present computational challenges at once. In particular, the number of feasible paths in sets S^{sk} (set of AP-MEC facility paths) and $Q^{k'k''}$ (set of synchronization paths between MEC facilities) grows very fast with the network size. In order to obtain good feasible solutions in limited computing time, we implemented two ILP-based heuristics presented in Chapter 3 and whose flowcharts are presented in Figures 3.4 and 3.10, respectively for s-MNDP and l-MNDP variants.

In order to run our algorithm with the large size network given by the Real World Dataset, we have to further reduce to size of the network in a pre-processing step that is added at the beginning of both s-MNDP and l-MNDP variant:

1. we fix the location of aggregation nodes, and the assignment of APs to them, creating clusters of APs of limited size and minimum worst-case latency through the following heuristic: (i) fix a number F of aggregation nodes to be installed; (ii) fix a maximum number G of APs connected to each aggregation node; (iii) run a Partitioning Around Medoids algorithm [68, 69] on the set of APs to choose F baricentric ones; (iv) use such a solution as initialization for a G-capacitated F-center alternating heuristic. This alternating heuristic, in turn,

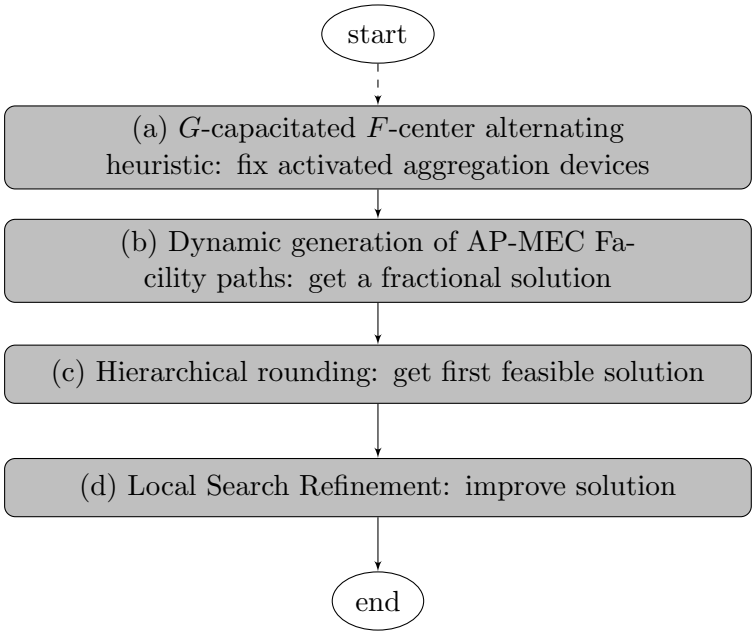


Figure 4.3: Structure of the s-MNDP for Real World Dataset

works as follows: (i) fix the locations of aggregation nodes, and solve an ILP for assigning the APs to aggregation nodes, forming clusters where at most G APs are connected, and minimizing the maximum distance between an AP and the center of its cluster; (ii) choose as new center for each cluster the AP minimizing the maximum distance between all other APs in the cluster; then iterate from (i), until no more changes in the solution are observed.

2. we fix the x_i variables in our models according to the G-capacitated F-center solution obtained as above, we fix $J = K = \{i \in I : x_i = 1\}$, and we remove from each S^{sk} set all paths in which the AP s is not assigned to the aggregation device of its cluster. After preliminary experiments, we fixed $F = 50$, $G = 1.3 \cdot (|B|/F)$.

The s-MNDP Matheuristic for the real-world dataset is slightly modified, and its structure is presented in Figure 4.3: the clustering step is not executed, as well as the clustering update iteration, and its replaced by the G-capacitated F-center alternating heuristic to reduce the network size. The l-MNDP Matheuristic does not change, the G-capacitated F-center alternating heuristic is added as first step.

As for the computational results assessments presented in Chapter 3.7, we implemented our algorithms in C++, using IBM ILOG CPLEX 12.6 [72] to solve

both LP and MILP problems. Our experiments ran on an Intel i7 4GHz workstation equipped with 32 GB of RAM.

We experimented on the real-world dataset considering three MEC facility size cases: tiny MEC facility of $C = 1.5$ racks, car parking MEC Facilities of $C = 2$ racks and $C = 2.5$ racks, and a 2-4 DC-room MEC facility with $C = 40$ racks. Using values from [80], we assume one rack to host up to 2500 VMs. The case with a single rack proved to be infeasible for our instances.

Considering bottleneck-free back-hauling networks ($U \leq 1$), where latency is approximately directly proportional to the euclidean distances among nodes, we consider three latency bounds $\bar{D}$: 'loose', 'mid-level', and 'strict'bounds, corresponding respectively to roughly the urban area radius (that is, half of the maximum distance between a pair of APs, 15 km), 4/5 of it, and 2/3 of it. These three levels of MEC facility access latency are chosen to correspond to three reference MEC services: delay-tolerant *storage box services* for the loose case, delay-sensitive *remote desktop services* for the mid-level case, and delay-critical *augmented-reality support services* for the strict case. We express these bounds as relative numbers, since there is no available public information on absolute MEC facility network latency requirements, despite partial valuable information can be found at [34, 81]. At last, we have to define the value for maximum length of a single link (parameter $\bar{d}$), that is needed for the definition of a feasible AP-MEC facility assignment path (defined in Chapter 3 by the equation (3.33)). We consider four different values: the three latency bounds $\bar{D}$ (the radius ($15km$), 4/5 of the radius ($12km$) and 2/3 of the radius ($10km$), resp.), and a stricter value of 1/2 of the radius ($7.5km$). It is not trivial to forecast the behavior of our multi layer model just looking to the single link maximum length: counting all possible feasible paths in the network is computationally impossible on networks of realistic size.

In Figure 4.4 we present the Complementary Cumulative Distribution Function (CCDF) of the percentage of APs that an AP can be connected to with a single link, for each possible value of $\bar{d}$. We called this number 'One-Hop Neighbors'. We can notice that: if we consider a link with a length equal to the radius of the area ($15km$) all APs are connected to at least 40% of all other APs with a single link; considering 4/5 and 2/3 of the radius, almost half of the APs can reach each at least half of all other APs; finally considering 1/2 of the radius ($7.5km$), at maximum a node can reach less than 40% of all other nodes. After preliminary tests, we choose to set $\bar{d}$ to the stricter value of $7.5km$, as higher values would lead to trivial solutions (in two hops all nodes would be connected), while lower values would lead no feasible solution.

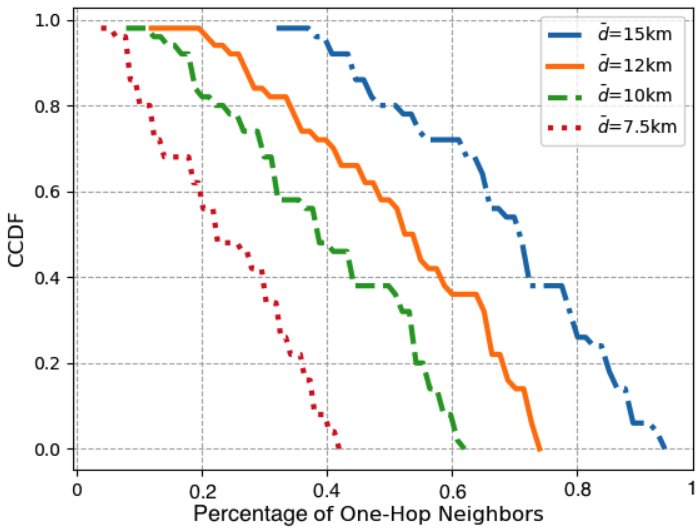


Figure 4.4: CCDF Number of Neighbors Given $\bar{d}$ (% on total number of APs)

As already described, the maximum link utilization (percentage) U needs to be kept as low as possible in order to better master the congestion risk and guarantee the QoE for real-time and interactive services. We evaluate three levels for the maximum link utilization bounds: 'loose', 'mid-level', and 'strict', corresponding to the 10%, 20% and 30%, respectively, of the link capacity. The stricter they are, the better interactive service support is expected to be, such as for remote desktop and augmented reality. Storage box (TCP-based) services are fault tolerant, given the bulk transfer nature of its data.

In the following, we report extensive results for the s-MNDP, then we investigate the parametric scenarii for the l-MNDP variant, finally comparing the approaches in terms of virtual resource migration volume with VM bulk migration. In the plots, we label every parametric scenario with a pair of letters representing respectively the maximum link utilization percentage level U and the MEC facility access latency level $\bar{D}$, as in Table 4.2. Moreover, a summary of the parameter setting for the s-MNDP algorithm is presented in Table 4.3.

		MEC Facility Access Delay Bound		
		Strict	Mid-Level	Loose
Maximum Link Utilization Bound	Strict	S-S	S-M	S-L
	Mid-Level	M-S	M-M	M-L
	Loose	L-S	L-M	L-L
Related Reference MEC Services		*Augmented Reality Supp.*	*Remote Desktop*	*Storage Box*

Table 4.2: Labelling of parametric scenarii

Parameter	Value	Origin
$\|B\|$	50	Set after preliminary analysis
n_i	-	Query on Operator Dataset
b_i	-	Query on Operator Dataset
$d_{i,j}$	-	Euclidean Distance given coordinates of APs from Operator Dataset
$u_{i,j}$	$10Gbps$	Assumed to giga-Ethernet granularities
$\bar{d}$	$7.5km$	Set after preliminary analysis
$\bar{H}$	4	Assumed accordingly to [35]
C	[3750, 5000, 100000]	Parametric Analysis
$\bar{D}$	[$15km$,$12km$,$10km$]	Parametric Analysis
U	[0.1, 0.2, 0.3]	Parametric Analysis

Table 4.3: Real World Scenario - s-MNDP Parameters Setting Summary

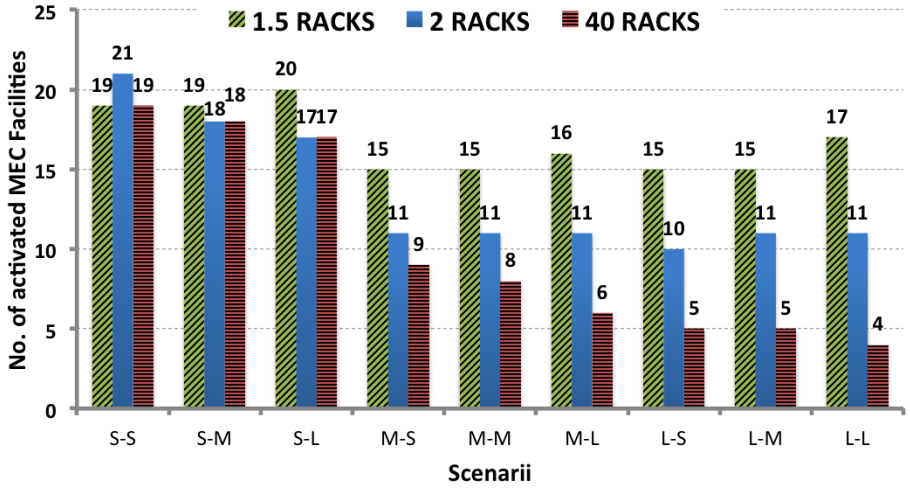


Figure 4.5: s-MNDP - Number of enabled MEC Facilities.

4.3 Experimental Results

4.3.1 s-MNDP Results

For the s-MNDP variant we consider the full day average behavior, by averaging the traffic and number of users at each AP over the full day. Combining in every possible way capacity, delay and link utilization bound settings, we get $3 \cdot 3 \cdot 3 = 27$ scenarii.

Number Of Installed MEC Facilities As first fitness measure we consider the number of installed MEC Facilities, as reported in Figure 4.5. We can observe that:

- w.r.t. MEC Facility capacity C, trivially the lowest rack capacity leads to the largest number of installed MEC Facilities (i.e. between 15 and 20 over 50 nodes), with no relevant changes by strengthening delay and utilization bound. The difference in number of facilities between the 2-rack and 40-rack cases does not justify the huge capacity gap: this effect is due to the delay constraints requiring a minimum level of geo-distribution. Overall, intermediate size facilities (2 racks) appear as the most appealing option: smaller ones require to install on average one MEC Facility every two aggregation nodes, which appears as too much, and larger ones do not reduce the number of required

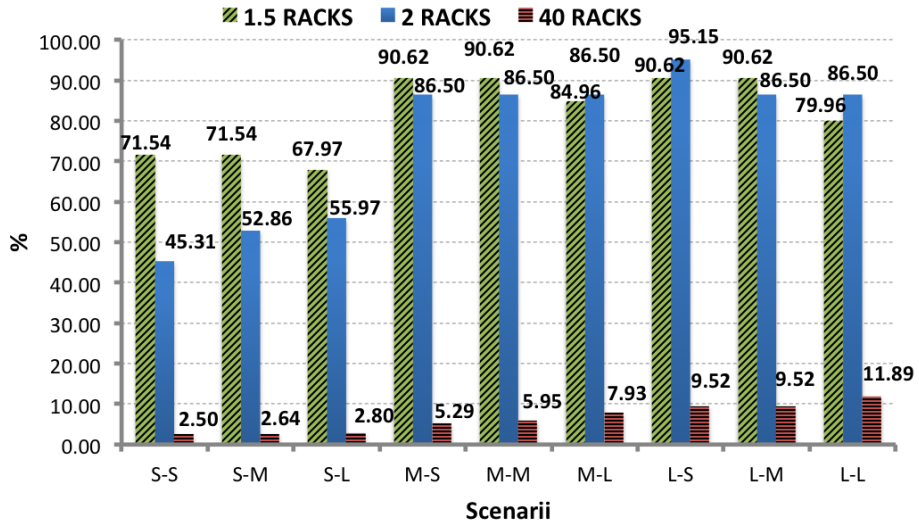


Figure 4.6: s-MNDP - Average usage of MEC Facilities (%).

facilities significantly, leading to resource and space waste.

- w.r.t maximum link utilization, the number of required MEC Facilities rapidly grows while moving from mid-level to strict bound, except for the 1.5-rack case, likely due to the lower aggregation of traffic on a more distributed MEC network.
- w.r.t. MEC Facility access latency, we cannot see clear trends. On average, the solutions show little sensitivity on the value of $\bar{D}$, suggesting that, if a decision maker decides to resort to static models unaware of users and VMs mobility, a location planning could be pursued without specifically taking into account different services.

MEC Facilities Usage As second fitness measure, we consider the average usage of the enabled MEC Facilities, whose percentage values are reported in Fig. 4.6. Such a value is trivially related to the number of enabled MEC Facilities. We can however observe that:

- tiny MEC Facilities have always a high average usage, with a slight usage decrease just in the case with strict link utilization, having a higher number of enable MEC Facilities;
- 2-rack MEC Facilities show a behavior similar to tiny ones on mid-level and

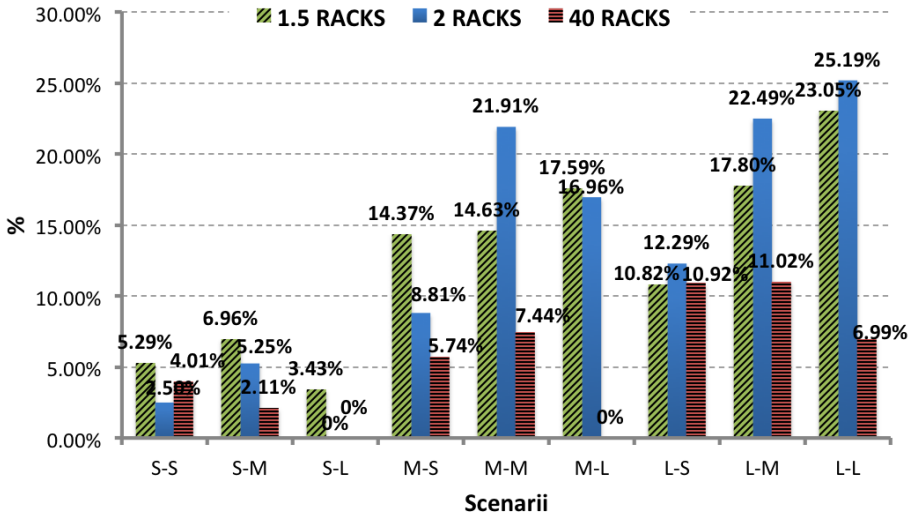


Figure 4.7: s-MNDP - Ratio of users with violated SLA after migration.

loose link utilization (scenarii M-* and L-*); on the other hand, strict constraints on link utilization (scenarii S-*) lead to a remarkable decrease of the usage;

- as expected, very big MEC Facilities always show little average usage, independently of other parameters choice;
- the setting of MEC Facility access latency bounds has very little impact on the average MEC Facility usage.

Users With Violated SLA As third fitness measure, we consider the percentage of users whose SLAs are violated after their migration. In details, given a solution $\bar{S}$ resulting from s-MNDP, we know by the parameter $f_{s',s''}$ that users migrate in the planning horizon between APs s' and s''; at the same time, we know, by values of variables $r_p^{s,k}$ in $\bar{S}$, which are those MEC Facilities k' and k'' servicing s' and s'', respectively. If it is possible to construct, after the optimization process, a feasible synchronization path between k' and k'' respecting constraints (3.67), then we say that the SLA of those $f_{s',s''}$ users are respected; otherwise we say that they are violated. Indeed, if a feasible synchronization path cannot be established, a user may perceive a latency during migrations that exceeds his SLA. Our results are presented in Fig. 4.7, where we notice that:

- enabling a high number of MEC Facilities leads to low percentage of users with

violated SLA: this is the case when the constraint on maximum link utilization is strict. For the scenario *S-L* we have no unsatisfied user for neither the 2-rack nor the 40-rack case;

- conversely, enabling a low number of MEC Facilities, the percentage of unsatisfied users increases up to 25%.

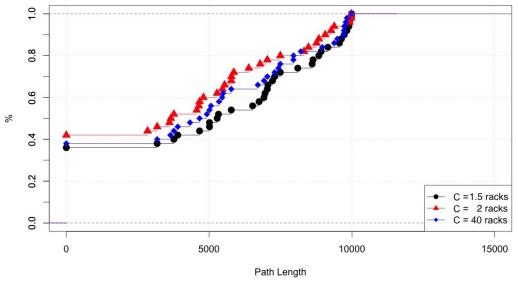
We argue the reason of this behavior to be the following: when the number of enabled MEC Facilities is high, it is possible to create a higher number of feasible synchronization paths by taking advantage of the higher number of direct links between core nodes. Indeed, the lack of control on the number of unsatisfied users in s-MNDP models is the main motivation to consider dynamic planning ones, which instead allow to explicitly enforce SLA to be never violated.

MEC Facility Access Path Lengths As fourth fitness measure, we consider the cumulative distribution function of the MEC Facility access path length, as reported in Figure 4.8. We can notice that:

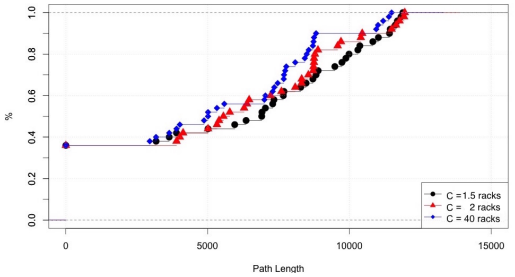
- w.r.t. MEC Facility capacity C, no clear trend is found; however we can notice that usually tiny MEC Facilities need longer paths, while high capacity MEC Facilities need shorter ones: this may be due to the fact that by providing very high capacity, each AP can connect to the nearest MEC Facility, while with lower capacity the nearest MEC Facility may be congested, and therefore a farther one has to be used.
- w.r.t. access delay bound, trivially, with looser access delay bounds the average path length increases; however, the distribution of path lengths does not show striking differences.
- mid-level and loose bounds on the maximum link utilization yield very similar path length distributions. Instead, for strict bounds, more paths are short ones: very few aggregation nodes can route traffic on the same links to the MEC Facilities; this requires to activate many facilities, that in turn allow to assign aggregation nodes to near MEC Facilities.

As a general remark, we note that a small percentage of paths have length equal to the threshold $\bar{D}$, and this is a sign that our final solutions still have room to improve MEC Facility access latency for many users.

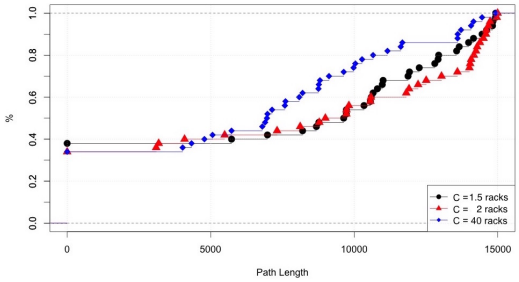
Computational Efficiency Finally, on the computational efficiency side, all s-MNDP instances had a running time of few minutes, with an average time of 360



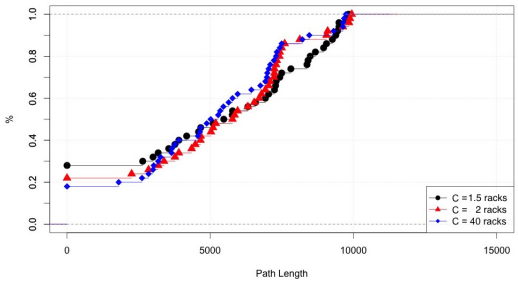
(a) S-S



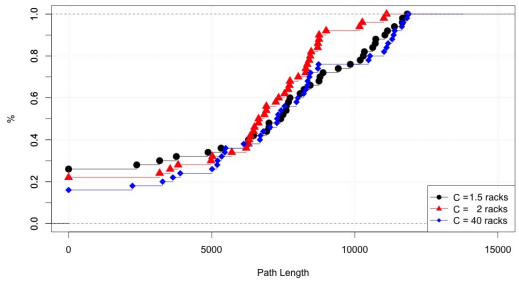
(b) S-M



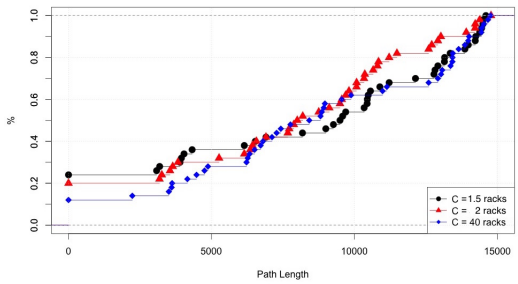
(c) S-L



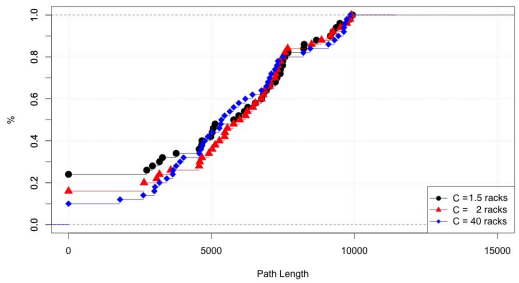
(d) M-S



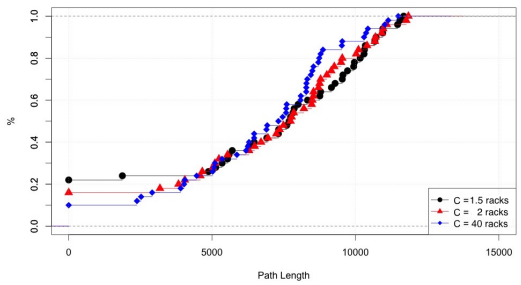
(e) M-M



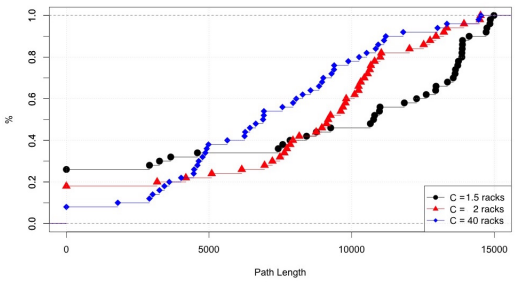
(f) M-L



(g) L-S



(h) L-M



(i) L-L

Figure 4.8: s-MNDP - Cumulative Distribution Function of access paths length

sec., a minimum of 56 sec. and a maximum of 821 sec., with no evident differences depending on parameters. In Table 4.4 we include details for every scenario of our dataset (row-wise), identified by the values of parameters C, U and $\bar{D}$. For every instance we include the total time required by the execution of the algorithm in seconds (column 'total time') and we report several information for every stage of the matheuristic, respectively the column generation ('CG'), the hierarchical rounding and pricing process ('R&P') and the local search. For the column generation process we report the final lower bound ('LB'), the number of iterations ('no. iter'), the time spent in every master LP execution in seconds ('master time') and the time spent in every pricing subproblem execution in seconds ('pricing time'). For the hierarchical rounding and pricing we report the same information given for the column generation together with the feasible integer solution given by the final MIP execution, used to round the remaining free variables ('MIP sol.'). For the local search process we report the time spent by the generic MIP solver in seconds ('time') (we remark that we fix a maximum execution time of 300 seconds) and the final feasible integer solution ('sol.') together with the number of activated MEC facilities ('$|K|$') and of core nodes ('$|J|$') of the final solution. We remark that the number of aggregation nodes is fixed in the pre-processing step of the G-capacitated F-centers heuristic to the value of 50. Finally, we report the percentage of users with violated SLA over the total number of users ('% violated SLA'), also reported in Figure 4.7.

In order to measure the effectiveness of our algorithm, we can look at the gap between the final solution found and the lower bound of the column generation, on which the rounding is executed. Given z^* the final solution and with $\underline{z}$ the lower bound given by the column generation, we compute the percentage gap as $(z^* - \underline{z})/\underline{z}$. A high gap given with respect to the continuous relaxation of a problem usually lead to a poor effectiveness of a rounding process. We can notice that:

- w.r.t. MEC facility capacity C, with the 40-rack case the gap is very high, that is the final solution is in the worst case five times the CG lower bound; with loose maximum link utilization the gap improves but it is at best 22%. With 1.5-rack and 2-rack the gap is influenced more by the maximum link utilization;
- w.r.t the maximum link utilization, in the case of a strict constraint ($U = 0.1$) the gap is high for every MEC facility capacity; while in the case of mid-level constraint ($U = 0.2$) the gap is reasonable for the 1.5-rack and 2-rack cases, while it is high for the 40-rack case;

No clear trend rises from the comparison of different Access Delay Bounds ($\bar{D}$).

				CG				R&P					local search				
C	U	$\bar{D}$ (10^3)	total time	LB	no. iter.	mast. time	pric. time	LB	no. iter.	mast. time	pric. time	MIP sol.	time	sol.	$\|K\|$	$\|J\|$	% violated SLA
3750	0.1	10	128	14.900	23	18	0	16.393	165	59	1	27.4	32	20.4	19	9	5.29%
		12	180	14.800	22	21	0	15.276	200	90	1	29.2	44	20	19	5	6.96%
		15	435	14.700	38	44	0	15.174	311	250	3	30.4	59	20.9	20	4	3.43%
	0.2	10	146	10.924	49	27	1	13.685	150	45	0	27.9	27	22.2	21	7	14.37%
		12	190	10.800	29	36	1	12.124	176	97	0	25.5	35	19.1	18	6	14.63%
		15	763	10.701	91	153	3	11.327	275	422	2	26.1	97	18	17	5	17.59%
	0.3	10	372	10.908	59	25	0	11.973	145	29	0	15.9	11	12.3	11	8	10.82%
		12	430	10.800	28	27	0	11.402	254	59	2	17.4	35	12.3	11	8	17.80%
		15	568	10.700	44	42	0	11.173	244	149	3	17.2	58	12	11	5	23.05%
5000	0.1	10	146	10.924	49	27	1	13.685	150	45	0	27.9	27	22.2	21	7	2.50%
		12	190	10.800	29	36	1	12.124	176	97	0	25.5	35	19.1	18	6	5.25%
		15	763	10.701	91	153	3	11.327	275	422	2	26.1	97	18	17	5	0%
	0.2	10	372	10.908	59	25	0	11.973	145	29	0	15.9	11	12.3	11	8	8.81%
		12	430	10.800	28	27	0	11.402	254	59	2	17.4	35	12.3	11	8	21.91%
		15	568	10.700	44	42	0	11.173	244	149	3	17.2	58	12	11	5	16.96%
	0.3	10	361	10.906	45	21	0	11.544	134	22	2	13.6	10	11.7	10	12	12.29%
		12	390	10.800	26	23	2	11.326	211	43	0	13.5	14	12	11	5	22.49%
		15	461	10.700	41	39	0	12.100	263	89	3	14.2	19	12	11	5	25.19%
10^5	0.1	10	103	5.165	76	33	0	7.730	133	37	2	21.7	17	20.3	19	8	4.01%
		12	242	4.012	121	74	1	4.613	155	53	0	20.4	97	19	18	5	2.11%
		15	396	2.829	205	218	3	3.956	138	110	0	18.9	33	17.9	17	4	0%
	0.2	10	64	5.128	95	29	0	5.504	149	12	0	10.5	12	10.3	9	8	5.74%
		12	114	4.000	111	41	1	4.209	203	20	3	10.2	24	9.4	8	9	7.44%
		15	649	2.819	379	251	2	3.039	187	35	4	8.9	300	7.1	6	6	0%
	0.3	10	56	5.125	125	34	0	5.424	172	11	2	6.5	2	6.3	5	8	10.92%
		12	472	4.000	104	51	0	4.204	211	14	1	6.1	300	6.1	5	6	11.02%
		15	821	2.817	356	301	10	3.000	165	22	4	4.9	300	4.9	4	4	6.99%

Table 4.4: s-MNDP - Real World Dataset - Mean Daily Activity - Computational Analysis

4.3.2 Dynamic Planning Results

In a second round of experiments, we tested the behavior of the Dynamic Planning models (see Chapter 3.5) in the case of two time-frames: from 7 am to 8 pm, and from 8 pm to 7 am. These approximately represent working and resting hours. We compared through simulations the Bulk and Live VM Migration cases. As VM replication mobility technology can be seen as a special case of VM Live Migration with infinitesimal amount of memory to transfer, we have not considered experiments using this technology; however results on VM Live Migration are valid also for VM replication scenario. Moreover we included in our tests also an s-MNDP model as reference, using data from the working-hours time-frame only (i.e. from 7am to 8pm). This may be seen as a 'worst case' planning option, since it is considering the bottleneck time-frame only; we remark that still no guarantee is obtained on SLA satisfaction, even resorting to such a conservative static option.

We restrict the simulations to the six most interesting scenarii, looking at the s-MNDP results. That is, we discard the 1.5-rack scenarii and the loose MEC Facility access latency bound scenarii: the first proved to yield infeasible instances when the demand of the sole working-hours is considered; the second provided less interesting insights in previous analysis.

We set the width of the time window suitable to perform VM orchestration T_w to 5 hours, which is less than a half of our time-frames. For the storage synchronization path maximum length, we set $\bar{D}^Q = 12.75\ km$ (4/5 of the urban area radius), i.e. we consider the synchronization as a service that requires a mid-level latency bound. The size of the disk for *augmented-reality support* VMs, requiring strict latency bounds, is 20GB, reasonably lower than the one for *remote desktop* VMs, that is 60GB, requiring mid-level latency bounds. Conversely, the size of the memory is higher for augmented-reality (8GB) than for remote desktop (4GB). To preserve tractability we defined the mapping function Φ of (3.68) as the following linear function that considers the synchronization traffic generated by any user as a percentage ϕ of the average traffic generated by all users:

$$\Phi(x) = x \cdot \frac{\sum_{i \in I} \sum_{t \in T} n_i^t}{\sum_{i \in I} \sum_{t \in T} b_i^t} \cdot \phi$$

The percentage ϕ is characterized by the type of MEC service: considering remote desktop VMs, only part of the disk is expected to be modified upon user actions; so ϕ is set to 70%. Instead, for augmented-reality support VMs disks are expected to be smaller and consequently only small volume need to be synchronized; so ϕ is set

		MEC Service Properties			
		Access Delay Bound	Memory Size (GB)	Disk Size (GB)	ϕ^*
Reference MEC Services	Augmented Reality Support	Strict	8	20	30%
	Remote Desktop	Mid-Level	4	60	70%

*percentage of users' traffic that induce synchronization traffic

Table 4.5: Reference MEC Services Parameters.

to 30%. A summary of the parametrization of the Reference MEC services used in our experiments can be found in Table 4.5. Moreover, a summary of the parameter setting for the dynamic planning scenario is presented in Table 4.6.

Dynamic Planning with Live Migration.

At first, we experimented on l-MNDP variant with VM Live Migration policy, with our l-MNDP Matheuristic. Our algorithms could find feasible solutions for 9 over 12 of these instances. In the results provided hereafter, the missing solutions are marked with the notation **NF**, meaning *Not Found*. In fact, as our heuristic builds the solution by rounding one variable at a time with a two-stage process, at every step there is a chance to perform a rounding that eventually leads to infeasibility. To improve this behavior a diversification mechanism could be implemented. However, since these missing results do not affect the overall understanding of our experiments, we did not further investigate in that direction.

Number of Installed MEC Facilities As first fitness measure we consider the number of enabled MEC Facilities, reported in Fig. 4.9. We note that, while in the 2.5-rack case l-MNDP enables a slightly higher number of MEC Facilities w.r.t. s-MNDP, the models behave similarly in the 40-rack case.

A fitness measure related to the number of activated MEC facilities, we consider

Parameter	Value	Origin
$\|T\|$	2	Assumed to have two time-slots: working hours (7 am to 8 pm) and resting hours
n_s^t	-	Query on Operator Dataset
b_s^t	-	Query on Operator Dataset
$f_{s',s''}$	-	Estimated from Operator Dataset
$\bar{H}^Q$	3	Assumed accordingly to [35]
$\bar{D}^Q$	$12.75km$	Assumed to have VM orchestration as delay-sensitive service
T_w	5	Largest meaningful value for our time-slots settings
V	-	Parametric Analysis (details in Table 4.5)

Table 4.6: Real World Scenario - l-MNDP Parameters Setting Summary

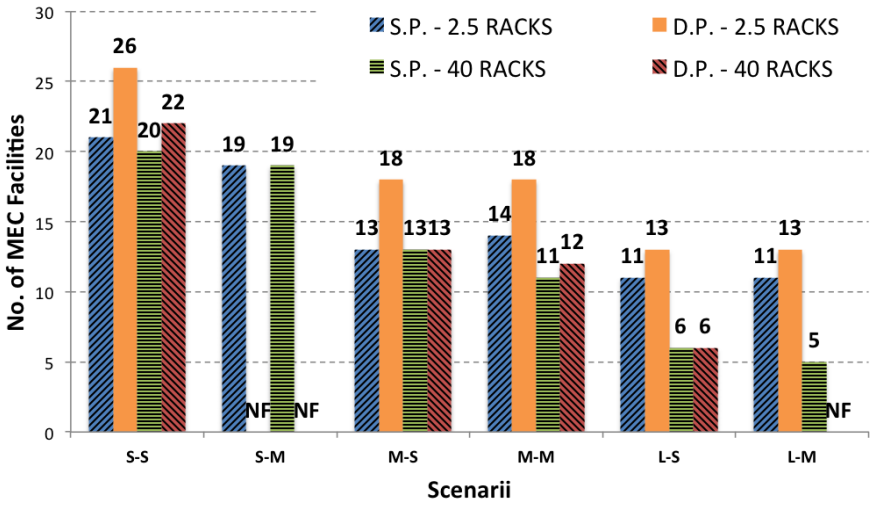


Figure 4.9: l-MNDP - Real World Dataset - Number of enabled MEC Facilities.

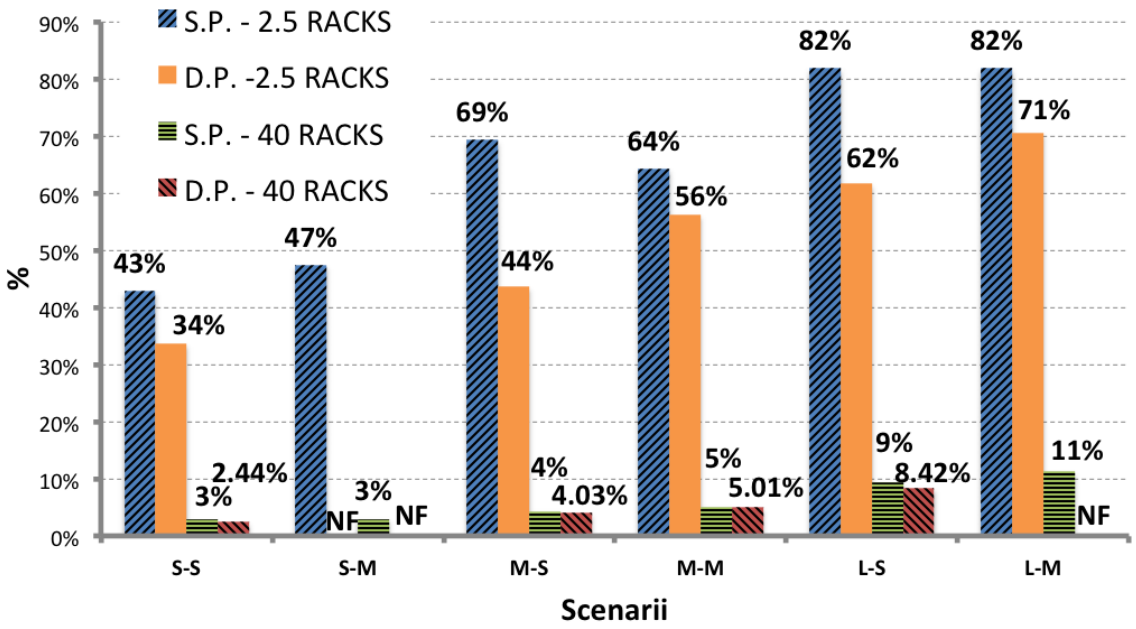
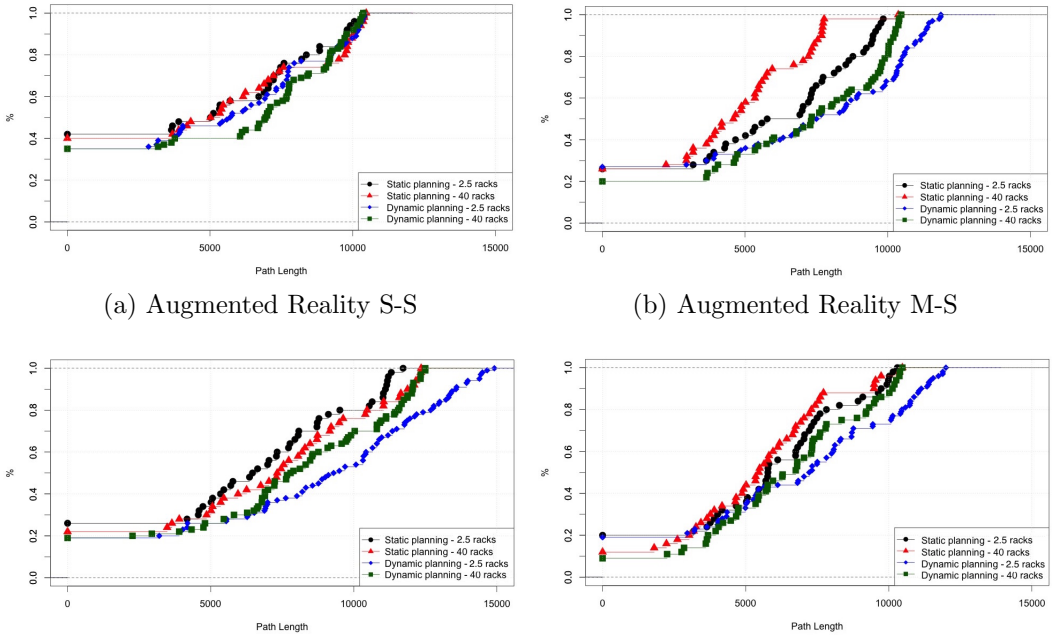


Figure 4.10: l-MNDP - Real-World Dataset - MEC Facilities Percentage Usage.

the average usage of MEC Facilities (Figure 4.10). By enabling a higher number of MEC Facilities, l-MNDP yields a lower average usage; large size MEC Facilities provide a very low usage in any case. For 2.5-rack scenarii, the average usage is around half of the total capacity. Instead, it is non-trivial to notice that in many scenarii the decrease in percentage of use remains limited while moving from the static to the dynamic planning, highlighting the fundamental role of our optimization.

MEC Facility Access Path Lengths As second fitness measure, we consider the MEC facility access path lengths: in Figure 4.11 we consider the cumulative distribution function of the MEC Facility access path lengths. We can note that:

- s-MNDP uses shorter paths than l-MNDP, showing once again how the latter is more accurate in detecting that more resources are needed to produce solutions fulfilling SLA;
- the paths used in 2.5-rack scenarii tend to be longer than those in 40-rack scenarii, for both s-MNDP and l-MNDP, still matching the intuition that exploiting tight MEC Facility capacities requires the design of more involved clusters, hence longer association paths;



(a) Augmented Reality S-S

(b) Augmented Reality M-S

(c) Remote Desktop M-M

(d) Augmented Reality L-S

Figure 4.11: l-MNDP - Cumulative Distribution Function of access paths length

- a very small percentage of paths uses the maximum delay $\bar{D}$, showing that in general the resource is not scarce and do not complicate the resolution of either model.

Number of VM Migrations As third fitness measure we consider the expected number of VM migrations generated by the different planning models; this can be seen as a measure of expected incremental point traffic on the network. Such a value can be computed from the values of variables $g_{s's''}^{k'k''}$ that encode the number of users moving from AP s' to AP s'' associated to MEC Facilities k' and k'', resp.. We remark that in l-MNDP $g_{s's''}^{k'k''}$ terms are explicitly included as variables in the models, while in s-MNDP they can be computed in a post-processing phase, once optimization is over. In order to obtain normalized fitness values, we compute the following upper bound on the number of possible VM migrations:

$$\gamma = \frac{|T| \cdot (|T| - 1)}{2} \sum_{s' \in B} f_{s',s'} + |T|^2 \cdot \sum_{s',s'' \in B | s' \neq s''} f_{s',s''}$$

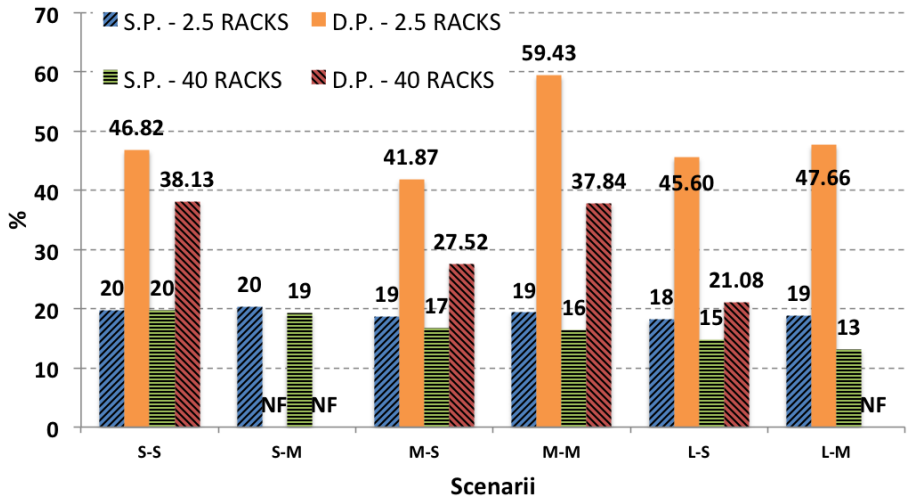


Figure 4.12: l-MNDP - Real World Dataset - Expected percentage of VMs to migrate.

which represents the number of migrations needed if all users are assigned to a different MEC Facility in each time slot, and we measure fitness as $(\sum_{k',k''\in B,s',s''\in K} g_{s's''}^{k'k''})/\gamma$. Our results are reported in Fig. 4.12 (as percentage points).

The fraction of migrated VMs for the l-MNDP is always higher than that for the s-MNDP, without striking differences among scenarii. It is crucial to consider, however, that only l-MNDP has an explicit control on the feasibility of these orchestrations. Therefore, as third fitness measure we consider the percentage of users with violated SLA after migration (Fig. 4.13). l-MNDP guarantees by design 0% of users with violated SLA. On the contrary, s-MNDP, which does not give any a-priori guarantee, shows an experimental behavior similar to that presented in Fig. 4.7: tighter constraints on link utilization lead to a higher number of enabled MEC Facilities, increasing the possibility to create synchronization paths through a higher number of direct links between core nodes, and hence yielding a low fraction of unsatisfied users. A remarkable scenario is the L-S with 2.5-rack MEC Facilities: s-MNDP asks to enable 11 MEC Facilities, requiring ∼18% of all possible VMs migrations, but leaving ∼14% of users unsatisfied; l-MNDP asks to enable, during the working-hours time-frame, 2 more MEC Facilities, requires ∼45% of all possible VMs migrations, but without violating any SLA.

To give an insight on the reason for SLA violations in s-MNDP, we show in Fig. 4.14 the clusters of APs associated to the same MEC Facility by: (a) s-MNDP during working-hours time-frame; (c) l-MNDP during working-hours time-frame,

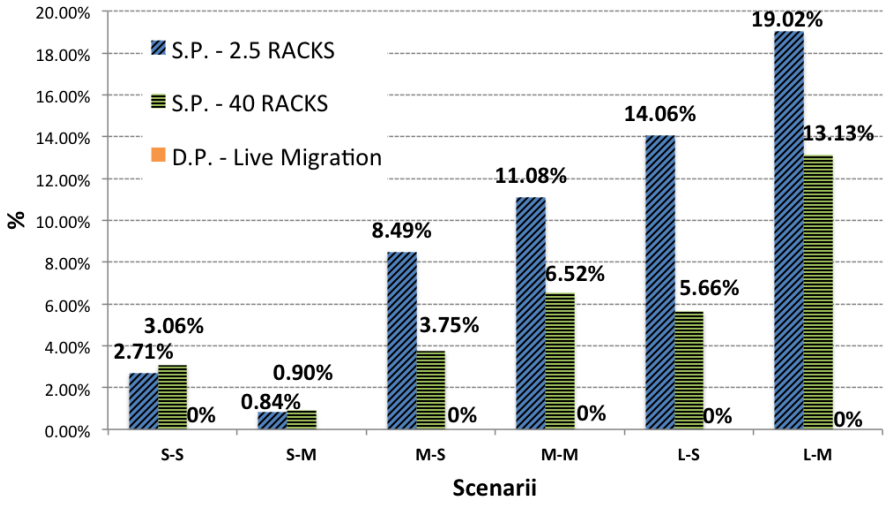
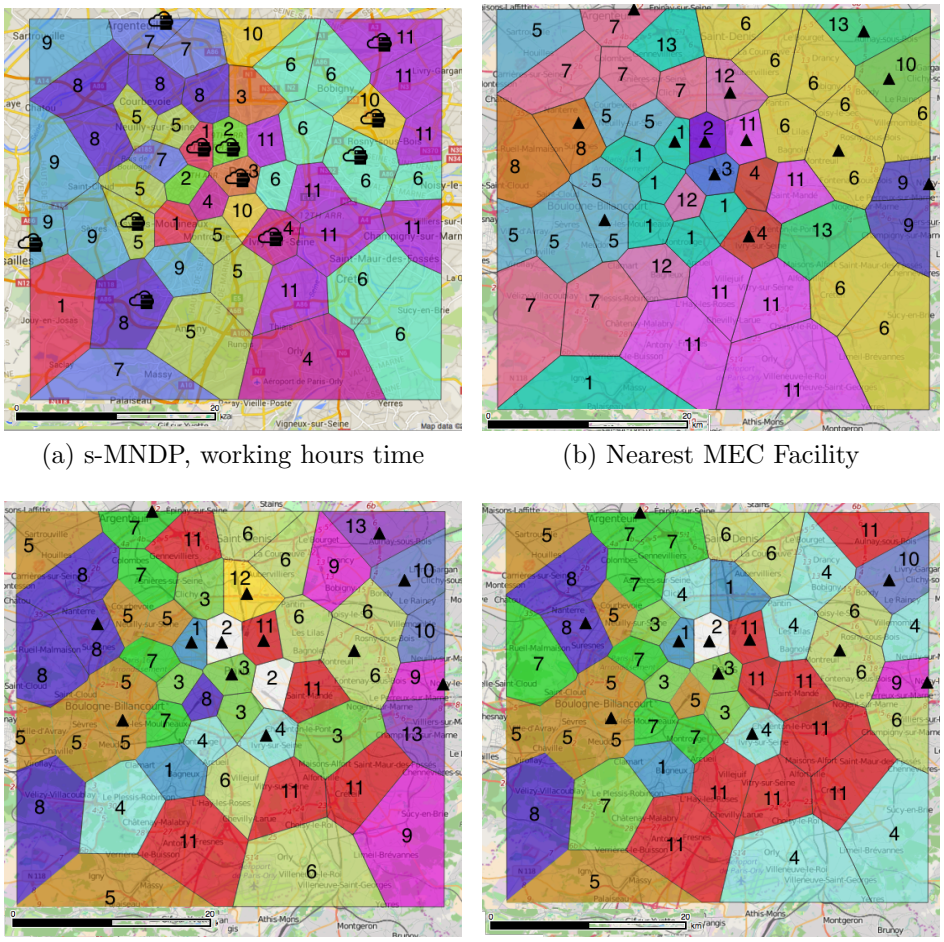


Figure 4.13: l-MNDP - Real World Dataset - Percentage of users with violated SLA.

and (d) l-MNDP during during night-time time-frame. MEC Facility locations are identified by a triangle icon and clusters are identified by different numbers and colors. First, we observe that s-MNDP spreads MEC Facilities more uniformly in the region, while l-MNDP locates the MEC Facilities in a smaller sub-area near the center of the territory, limiting the maximum distance between two MEC Facilities to satisfy SLA latency bound. Second, we observe that clusters are not necessarily compact; in fact, capacity restrictions may forbid an area to be associated to its nearest MEC Facility. l-MNDP tends to create a more involved clustering structure, especially during working-hours (Figures 4.14c and 4.14d): in these two figures major changes are observed in clusters 4 (light blue) and 7 (purple) and 9 (pink), while the remaining tend to keep the same structure over the two time frames.

Computational Efficiency On the computational efficiency side, while s-MNDP instances have execution times in the scale of few minutes, l-MNDP instances have execution times that ranges from few hours to several days. In particular, while s-MNDP cases have an average execution time of 4 min., with a minimum of 74 sec. and a maximum of 17 min., l-MNDP cases have and average execution time of $\sim$ 22 hours, with a minimum of 75 min. and a maximum of about 6 days. Since our model is designed for medium and long term planning, none of them appears to be critical. In Tables 4.7 and 4.8 further computational assessment measures are presented for the s-MNDP instances and for the l-MNDP instances, respectively.



(a) s-MNDP, working hours time

(b) Nearest MEC Facility

(c) l-MNDP, working hours time

(d) l-MNDP, resting hours time

Figure 4.14: Clustering produced by AP-MEC Facility associations in L-M scenario with 2.5-rack MEC Facilities.

The structure of Table 4.7 is similar to the structure of the previously presented Table 4.4: we report the computational solutions of s-MNDP instance identified by the first three columns by MEC facility capacity C, the link maximum utilization percentage U and the MEC Facility Access Delay Bound $\bar{D}$.

In Table 4.8 we report details of the l-MNDP Matheuristic for all scenarii (row-wise), identified by the MEC reference service ('MEC Ref. Service'), the capacity of MEC facilities ('C') and the maximum percentage link utilization ('U'). Then we report the required execution time in seconds (column 'total time') and several information regarding the first and the second stage of our Matheuristic. For the first stage we report the lower bound given by the column generation (column 'LB CG'), the number of column generation iterations of column generation (column 'no. iter.'), the lower bound at the end of the hierarchical rounding and pricing process (column 'LB R&P'), the time in seconds required by all master LPs executions (column 'master time'), the time in seconds required by the pricing subproblems separately for the AP-MEC facility assignment path variables (column 'pricing time r_p^{skt}') and for the synchronization path variables (column 'pricing time $q_p^{k'k''t}$'); finally we report the final infeasible solution (column 'sol.'). For the second stage we report the time in seconds required by the master LPs and the two pricing subproblems, together with the final feasible solution (column 'sol.') and the number of activated MEC facility, core nodes and aggregation nodes (columns '$|K|$', '$|J|$' and '$|I|$').

We were not able to find a feasible solution for three instances, all related to the Remote Desktop MEC Reference Service, that we label with **NF** (i.e. not found).

				CG				R&P					local search					
C	U	$\bar{D}$ (10^2)	total time	LB	no. iter.	master time	pric. time	LB	no. iter.	master time	pric. time	MIP sol.	time	sol.	$\|K\|$	$\|J\|$	$\|I\|$	% violated SLA
6500	0.1	105	133	10.90	25	23	0	12.401	167	72	0	26.3	18	22.4	21	9	50	2.71%
		125	268	10.80	30	41	0	11.803	205	168	0	28.5	33	20	19	5	50	0.84%
	0.2	105	99	10.90	23	20	0	13.334	172	49	2	19.4	17	14.1	13	6	50	8.49%
		125	183	10.80	27	27	0	13.416	242	86	3	22.7	38	15	14	5	50	11.08%
	0.3	105	364	10.90	22	17	0	11.445	186	36	1	13.6	6	12.2	11	7	50	14.06%
		125	982	10.80	26	32	4	11.291	351	84	296	14.5	241	12.3	11	8	50	19.02%
10^5	0.1	105	153	5.077	87	48	0	7.483	159	64	0	23.3	16	21.4	20	9	50	3.06%
		125	329	3.903	120	161	1	4.822	186	125	2	21.1	19	20	19	5	50	0.90%
	0.2	105	109	5.018	132	36	2	6.400	204	36	1	15.4	23	14.1	13	6	50	3.75%
		125	184	3.886	169	91	4	5.157	251	46	0	13	22	12.3	11	8	50	6.52%
	0.3	105	74	5.018	139	38	1	5.784	134	12	1	8.2	7	7.3	6	8	50	5.66%
		125	175	3.883	187	72	1	4.101	247	29	2	7.1	48	6.1	5	6	50	13.13%

Table 4.7: s-MNDP - Real World Dataset - Mean Work-Time Activity - Computational Analysis

				First Stage							Second Stage						
MEC Ref. Service	C	U	total time	LB CG	no. iter.	LB R&P	master time	pricing time r_p^{skt}	pricing time $q_p^{k'k''t}$	sol.	master time	pricing time r_p^{skt}	pricing time $q_p^{k'k''t}$	sol.	$\|K\|$	$\|J\|$	$\|I\|$
A.R.	6250	0.1	14116	11.031	66	27.300	10442	2049	799	28	59	2	72	30	26	35	50
		0.2	38836	10.869	63	19.101	25355	11392	1523	20	31	6	28	22.1	18	36	50
		0.3	17913	10.863	65	13.225	9939	6187	1266	15.1	32	4	10	16.7	13	32	50
	10^5	0.1	18983	5.235	212	23.000	13536	3669	959	24.7	58	4	37	25.8	22	33	50
		0.2	6838	5.194	214	14.700	4135	1553	728	15.4	42	3	10	16.4	13	29	50
		0.3	4142	5.190	223	7.300	3112	479	359	7.8	18	3	1	8.4	6	19	50
R.D.	6250	0.1	90176	10.890	66	27.018	31214	54163	3383	28.5	76	8	42	**NF**	-	-	-
		0.2	524839	10.806	77	19.300	42693	473636	7105	21	67	75	22	21.7	18	32	50
		0.3	79475	10.791	75	13.200	26754	49881	2034	15.5	21	13	6	16.1	13	26	50
	10^5	0.1	95829	3.982	626	20.900	35940	56180	2685	21.6	30	12	36	**NF**	-	-	-
		0.2	50798	3.937	220	13.100	22383	25850	2020	14.4	28	14	8	15	12	25	50
		0.3	20608	3.923	265	6.001	8225	10679	1396	7.1	18	2	0	**NF**	-	-	-

Table 4.8: l-MNDP - Real World Dataset - Computational Analysis

4.3.3 Nearest MEC Facility Association

In order to further assess the need for considering the association between APs and MEC Facilities directly within the planning model, we propose the following experiment: given the network resulting by our model we disregard the association of AP to MEC Facility, and instead we associate APs to the nearest MEC Facility in terms of number of hops. For example, in Fig. 4.14b we can see the cluster of APs associated to the nearest MEC Facility using the same network used in Fig. 4.14c and 4.14d.

As a first comparison, in Fig. 4.15 we show the percentage of users with violated SLA after a nearest MEC Facility association. As compared with our approach in Fig. 4.13, we remark that: (i) nearest MEC Facility association produces SLA violations even with a network produced by the l-MNDP model, while our approach guarantees no violations; (ii) for almost all static planning scenarii, violations are worse with nearest MEC Facility than with our approach.

As a second comparison, we compute the MEC Facility overuse, i.e. the excess over VM capacity C. In Fig. 4.16 we present the overuse amount, noting that with 40-rack MEC Facilities there is no overuse, and with tiny 2.5-rack MEC Facilities there is a high overuse for several instances of both s-MNDP and l-MNDP approaches.

4.3.4 Bulk VM Migration Results

Our initial attempts to optimize Dynamic Planning models with VM Bulk migration produced no feasible solutions on any instance of the dataset. Indeed, bulk migration policies clash with the ambition of producing ahead a careful service and synchronization plan; in other terms, bulk migrations can be seen as the result of an unexpected need of synchronization, a human-ordered point operation, rather than a consolidated and automated operation.

Nevertheless, in order to analyze the impact of Bulk Migrations, we proceeded as follows. We produced solutions of l-MNDP, and we computed the maximum size of a VM file that the network could manage to transfer without violating user SLA. Such a value is unfortunately not directly available after optimizing l-MNDP models; on the contrary, the problem of finding it can be proved to be NP-Hard. Therefore, we performed the following simplifying assumptions: (i) MEC Facilities located in aggregation nodes are moved to the corresponding core nodes; (ii) synchronization

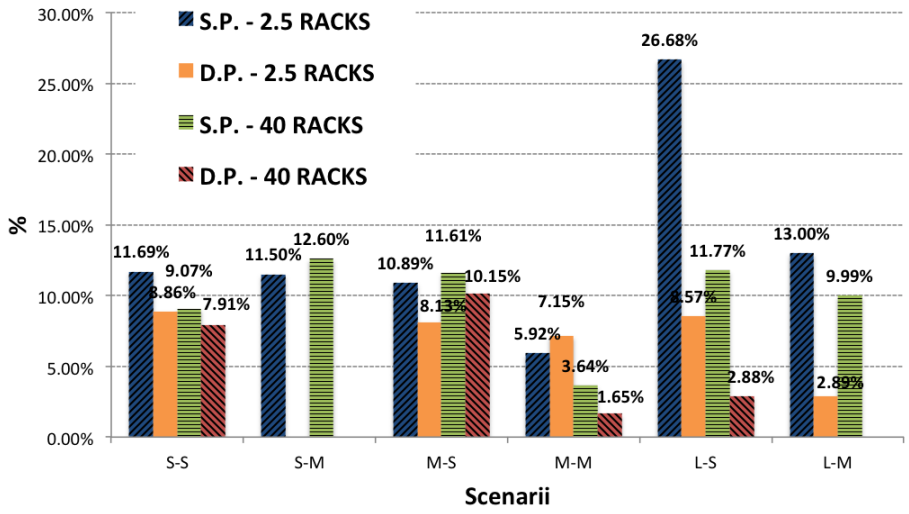


Figure 4.15: SLA violation (% users) with nearest MEC Facility association

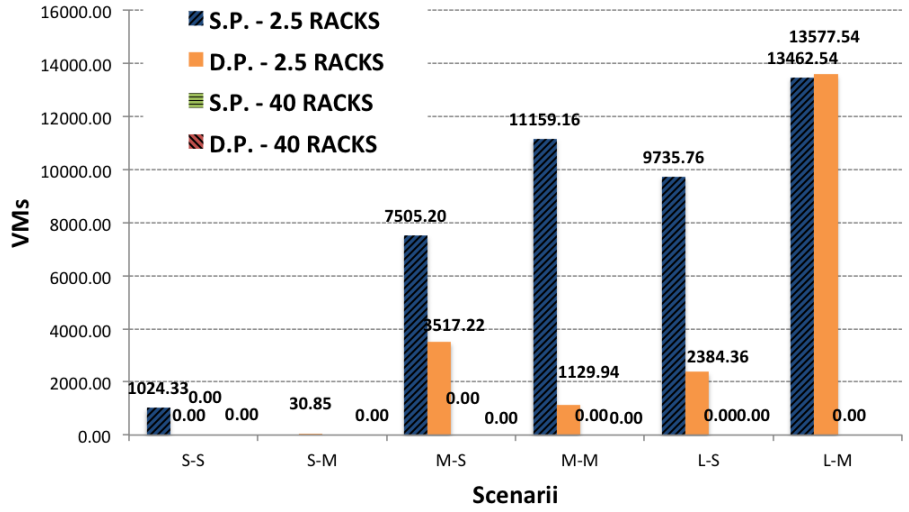


Figure 4.16: MEC Facility overuse with nearest MEC Facility association

paths are allowed for an arbitrary number of hops and arbitrary length - that is, synchronization is performed among core nodes only, and only maximum single-link length constraints and single-link latency bounds are kept. Assumption (i) is particularly mild, as any MEC Facility placed at aggregation level would represent a bottleneck of the whole network.

The problem of finding the largest file size Γ turns out to be a multicommodity-flow problem modeled as follows:

$$\max \Gamma \tag{4.1}$$

$$\text{s.t.} \sum_{k',k'' \in K} f_{i,j}^{k',k''} \leq (1-U) u T_w \qquad \forall (i,j) \in E^J \tag{4.2}$$

$$\sum_{j \in J} f_{i,j}^{k',k''} - \sum_{j \in J} f_{j,i}^{k',k''} = \begin{cases} \bar{f}_{k'k''}\Gamma \text{ if } i = k' \\ 0 \text{ if } i \neq k' \wedge i \neq k'' \\ -\bar{f}_{k'k''}\Gamma \text{ if } i = k'' \end{cases} \qquad \begin{matrix} \forall (k',k'') \in K \\ \forall i \in J \end{matrix} \tag{4.3}$$

$$f_{i,j}^{k',k''} = 0 \qquad \begin{matrix} \forall (i,j) \in E^J \mid \\ d(i.j) \geq \bar{d} \end{matrix} \tag{4.4}$$

$$f_{i,j}^{k',k''} \geq 0 \tag{4.5}$$

Let: E^J be the set of links between core nodes; $\bar{f}_{k'k''}$ be the fixed number of VMs to migrate between MEC Facilities k' and k''; and $f_{i,j}^{k',k''}$ be non-negative continuous variables representing the number of VMs to migrate from k' to k'' and whose migration path traverses link (i,j). Inequalities (4.4) and (4.2) model single-link length and latency bounds, resp.. Inequalities (4.3) are flow conservation constraints. That is, model (4.1)–(4.5) is a LP that can be optimized very efficiently.

For each 2.5-rack MEC Facilities case where we obtained a feasible solution in the VM Live Migration model, we run this model with a parametric analysis on the link length threshold $\bar{d}$: starting from the value used in l-MNDP experiments, we decreased it stepwise, until the problem became infeasible.

Our results are collected in Fig. 4.17. Three different features of each solution are reported: (i) the optimal Γ value, i.e. the maximum VM file size that the network can afford; (ii) the average number of hops of the generated synchronization paths; and (iii) the average length of the generated synchronization paths. Each chart contains a dashed black line, representing the required standard for synchronization paths. These are 3 hops, maximum total length of $12Km$ and either a $28GB$ file ($8GB$ memory and $20GB$ disk) for the augmented reality service or a $64GB$ file

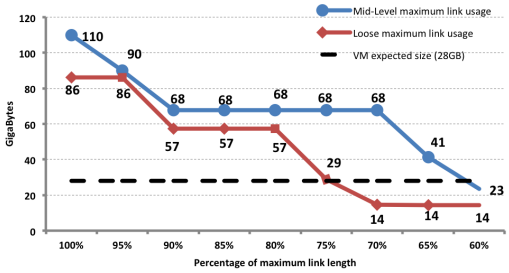
($4GB$ memory and $60GB$ disk) for remote desktop. That is, fully feasible solutions would have values above the dashed line in the leftmost chart, and below it in the central and rightmost ones: it is easy to check that in no case it was possible to find one of them. Matching intuition, using high allowed link length values, one can move very large VM files, at the price of generating highly infeasible synchronization paths. We can further note that:

- for augmented reality reference service, using the total link length we can route very big-size VMs (almost 3 times the desired size, see Fig. 4.17a), but with highly infeasible paths (almost 3 times more hops than expected, see Fig. 4.17b, and 5 times longer paths, see Fig. 4.17c). However a reasonable trade-off can be reached using 75% of the maximum link length: in this case we can route a $29GB$ VM file, with an average number of hops of 5 and an average path length that is 50% above the threshold;
- for remote desktop reference service we were not able to route the expected VM file size (see Fig. 4.17d): a maximum file size of $29GB$ can be routed, and still with violations in terms of average number of hops and average paths length. No improvement is achieved by lowering the allowed link length. Moreover using less than 80% of the link length already leads to infeasibility.

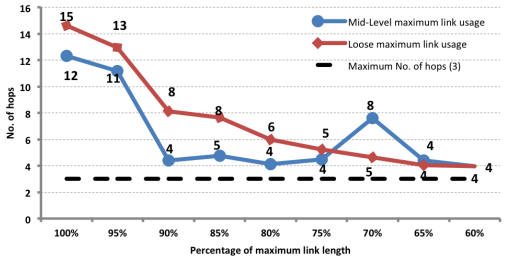
Summarizing, Bulk Migration seems to be a feasible alternative to Live Migration on Augmented Reality reference services, where the size of synchronization files is still limited; in fact solutions can be found, violating latency and maximum hop constraints only slightly. On the contrary, on Remote Desktop reference services, Bulk Migration does not appear as a viable option. In either case, matching Dynamic Planning models with VM Live Migration proves to be the most appealing option.

4.4 Conclusions

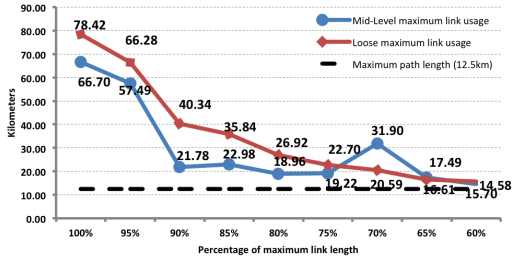
We provided extensive prescriptive analysis on MEC network design for mobile access metropolitan area networks. We compared the different planning options extensively for scenarii built over real cellular network datasets, differentiating between different traffic engineering and performance goals for reference MEC services, analyzing: (i) the use of network facilities resources, i.e. number of enabled MEC Facilities, usage of MEC Facility resources, migrated volume and (ii) the compliance with users' SLA. As conclusion we can state that:



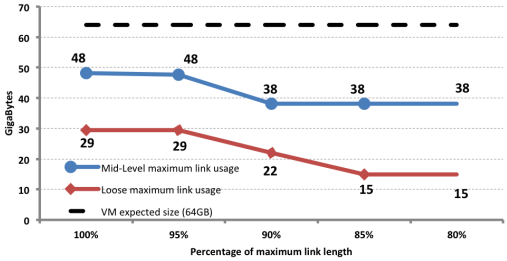
(a) Augmented Reality VM file size



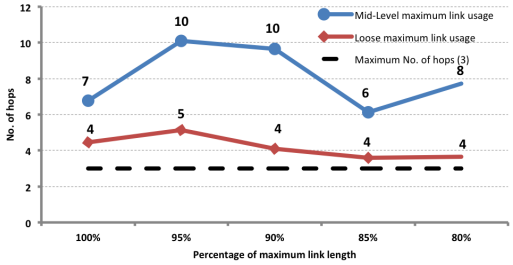
(b) Augmented Reality average No. of hops



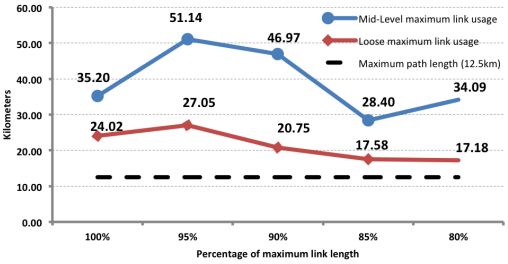
(c) Augmented Reality average paths lengths



(d) Remote Desktop VM file size



(e) Remote Desktop average No. of hops



(f) Remote Desktop average paths lengths

Figure 4.17: Bulk Migration post-processing results using 2.5-racks MEC Facilities.

- while we guarantee full compliance with users' SLA considering users mobility and dynamic variations of the network, their exclusion from the modeling leads to the infringement of SLA for up to 20% of users;
- the increase of use of network resources given by the consideration of users mobility is limited to at most 5 more enabled MEC Facilities for serving 600 APs, for the Paris metropolitan area network use-case (on real traffic logs);
- the simultaneous consideration of the design of the network, the association between APs and MEC Facilities and the routing is needed to keep compliance with the limited resource and users' SLA: decoupling these design decisions using trivial heuristics leads to SLA infringement for up to 27% of users and in MEC Facility capacity over-use;
- comparing VM Live Migration and VM Bulk Migration technologies, the former has proved eligible for the use both with delay-critical and delay-sensitive MEC services, while the latter constantly violates limits on network resources and seems to be a feasible alternative only when the size of VM files to synchronize is small.

Part II

Tactical MEC Network Planning

Chapter 5

Dynamic Mobile Edge Computing Facility Assignment

In this chapter we deal with the tactical side of the MEC network planning: given an existing MEC network including virtualization facilities of limited capacity, and a set of mobile APs 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 name this problem as the Dynamic Assignment and Switching Problem (DASP).

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 the combinatorial problem DASP, which 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.

Results were presented in [20] and appeared as technical report [21], that is now submitted for publication.

5.1 Introduction

In a MEC infrastructure, MEC facilities 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 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 [82, 83]. 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. An example of the application is presented in Figure 5.1: a MEC network with two MEC facilities (`K1` and `K2`) and four APs (`A` to `D`) is considered in three consecutive time-slots (`t`=0, 1 and 2). At time 0 APs `A` and `B` are assigned to MEC facility `K1` while APs `C` and `D` are assigned

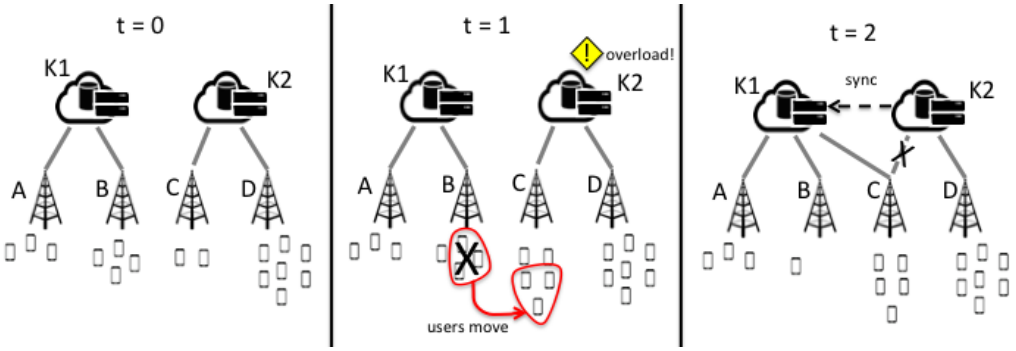


Figure 5.1: APs dynamic assignment to MEC facilities

to K2. At time 1, users move from the region served by AP B to the region served by AP C, hence increasing the load of the MEC facility K2, causing an overload of the facility. As facilities do not move in our scenario, an *orchestrator* decides to change the assignment of AP C to the underused facility K1 at time 2: a synchronization between the two facilities may be necessary. The aim of our tactical model is to provide a plan of assignments of APs to MEC facilities that tries to forecast the change of load of time $t = 1$ and anticipate the change of assignment.

5.2 A data-driven MEC management optimization framework

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 [84]. 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 [85]. 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 [12]. 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 [13, 14]. 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) [86]. We point to [87] 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 [15] 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 [16], in which a dual ascent technique from [88] 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 the strategic planning presented in Chapter 3 and given in input, one may expect features and computational challenges similar to those of multi-period location problems [89]. Recent approaches on that field include [19]: 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 [17] 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 [18], 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.

Summarizing, from a telecommunication research point of view, a need for the integration of data analytics for the autonomic management of telecoms network infrastructure is emerging [12]; however few works have been presented so far [13, 14],

none of which is taking into consideration our application. From a research operation point of view, our techniques lies at the edge of recent approaches for the multi-period extension of the GAP [15, 16] and the multi-period location problems [17, 18, 19].

Contribution summary and chapter 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 this chapter to the technical insights of our optimization core.

The chapter is structured as follows. We first detail our framework (Section 5.3). 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 5.4); 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 5.5). We show first of all that our algorithms are effective from a computational point of view (Section 5.6). 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 5.7). We finally draw some conclusions (Section 5.8).

5.3 A data-driven MEC management optimization framework

The algorithmic core architecture of our data-driven MEC management optimization framework is sketched in Figure 5.2. 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,

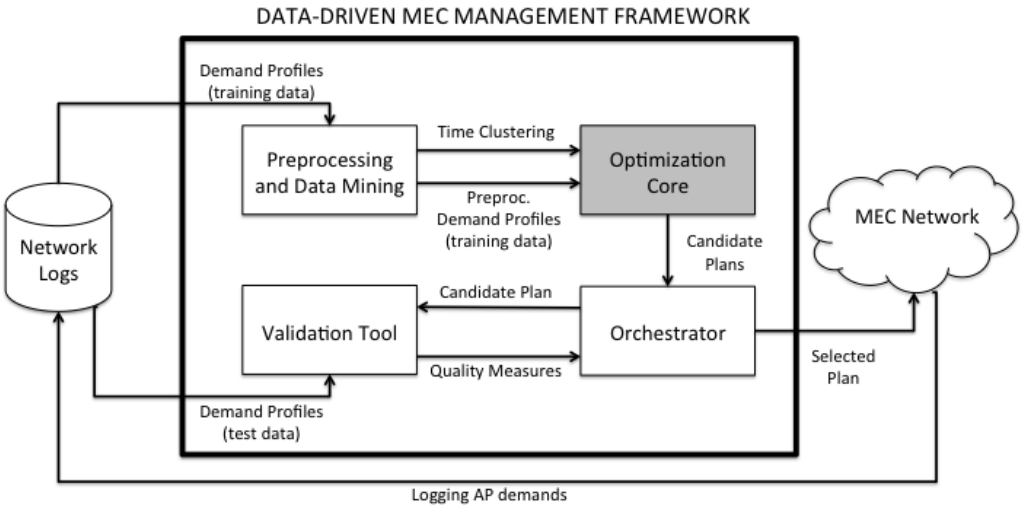


Figure 5.2: A data-driven MEC management optimization framework.

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 [1], in Section 5.7.1.

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 connections of APs to MEC facility (many APs to one 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 with 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 tightly constrained by 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.

5.4 Formulation

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 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 5.5.6.

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 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

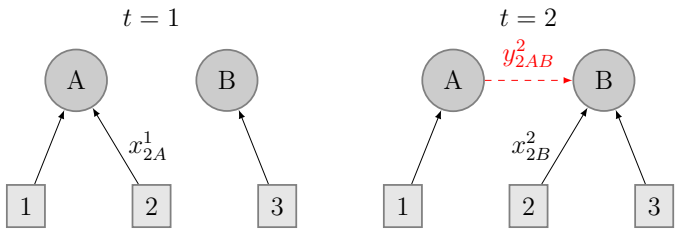


Figure 5.3: DASP - x and y variables

taking value 1 if AP $i \in A$ is associated with 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:

$$\min \alpha \sum_{t\in T} \sum_{i\in A} \sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} y_{ijk}^t + \beta \sum_{t\in T} \sum_{i\in A} \sum_{k\in K} d_i^t m_{ik} x_{ik}^t \tag{5.1}$$

$$\text{s.t.} \sum_{i\in A} d_i^t x_{ik}^t \leq C_k \qquad \forall t \in T, \forall k \in K \tag{5.2}$$

$$\sum_{k\in K} x_{ik}^t = 1 \qquad \forall i \in A, \forall t \in T \tag{5.3}$$

$$x_{ik}^t = \sum_{l\in K} y_{ilk}^t \qquad \forall i \in A, \forall t \in T \setminus \{1\}, \forall k \in K \tag{5.4}$$

$$x_{ik}^t = \sum_{l\in K} y_{ikl}^{t+1} \qquad \forall i \in A, \forall t \in T \setminus \{T\}, \forall k \in K \tag{5.5}$$

$$x_{i,k}^t \in \{0,1\} \qquad \forall i \in A, \forall k \in K, \forall t \in T \tag{5.6}$$

$$y_{i,k',k''}^t \in \{0,1\} \qquad \forall i \in A, \forall k', k'' \in K, \forall t \in T \tag{5.7}$$

the objective (5.1) 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 (5.2) impose that the overall demand assigned to MEC facility k at time t does not exceed its capacity. Constraints (5.3) impose that each AP is connected to a single MEC facility during a time slot. Constraints (5.4) and (5.5) link x and y variables in a flow conservation fashion: when $x^t_{ik} = 0$, 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). No additional restriction are imposed on the assignment of APs to MEC facilities; however, an infeasible association can be modelled fixing the corresponding variable $x^t_{i,k}$ to value 0 and, moreover, distance parameter $m_{i,k}$ is flexible enough to represent values coming from any distance function considering the topology of the network.

A sample instance with three APs (squares), two MEC facilities (circles) and two time-slots (left and right parts) is depicted in Figure 5.3: 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 (5.1) – (5.7) 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* (5.3) *are redundant.*

Indeed, for $t > 1$, they are implied by constraints (5.4) and (5.5) 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 (5.5) 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 (5.4), 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 $t = 1$. However, constraints (5.3) are included in the model for the computational experiments presented in Section 5.6.

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 (5.5) 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 (5.4), that can be feasible only if $x^{t+1}_{ik} = 1$ for more than a single $k \in K$, violating constraints (5.3), yielding infeasibility and thus leading to a contradiction. The converse is trivially implied by both constraints (5.4) and (5.5).

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

5.5 Optimization Algorithm

Unfortunately, when the size of the MEC network is large, even solving the continuous relaxation of model (5.1) – (5.7) turns out to be computationally hard. Therefore, we devise an ad-hoc exact solution approach based on decomposition.

Following the Dantzig-Wolfe reformulation principle [58], let

$$\mathcal{P}^i = \{(x^t_{ik}, y^t_{ikl}) : (5.3), (5.4), (5.5), (5.6), (5.7)\}, \forall i \in A$$

represent the convex hull of the feasible region respect to constraints (5.3) – (5.7). 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 (5.1) – (5.7) as the following *Master Problem* (MP):

$$\min \sum_{i \in A} \sum_{p \in \Omega^i} \left(\alpha \sum_{t \in T} \sum_{\substack{(j,k) \in \\ K \times K}} d^t_i l_{jk} \tilde{y}^{t,p}_{ijk} + \beta \sum_{t \in T} \sum_{k \in K} d^t_i m_{ik} \tilde{x}^{t,p}_{i,k} \right) z^p \tag{5.8}$$

$$\text{s.t.} \quad -\sum_{i \in A} \sum_{p \in \Omega^i} d^t_i \tilde{x}^{t,p}_{ik} z^p \geq -C_k \qquad \forall t \in T, \forall k \in K \tag{5.9}$$

$$\sum_{p \in \Omega^i} z^p = 1 \qquad \forall i \in A \tag{5.10}$$

$$z^p \geq 0 \tag{5.11}$$

We provide further details of this DW decomposition, together with an alternative version, in Appendix 5.A.

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 5.5.1) 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 5.5.2): 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 5.5.3) to obtain a corresponding upper bound. If upper and lower bounds do not match, we first perform probing to potentially fix variables (see Subsection 5.5.4) and then, when needed, we enter a recursive tree search phase (see Subsection 5.5.5). Our algorithms can be easily adapted to the split-assignment and periodic-plan variants (see Subsection 5.5.6).

5.5.1 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 4. In particular, for each time slot, APs are sorted by non-increasing demand and each AP is associated with 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

Algorithm 4 Greedy Binary AP-MEC facility assignment

$\bar{k}_a = \text{none}\ ,\ \forall a \in A$ {MEC facility associated with 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

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 (5.9). This ensures RMP feasibility also after branching.

We also remark that many other heuristics are possible, however considering and independent subproblem for each time-slot is likely to yield very poor solutions. Therefore classical heuristic as dose proposed in [90, 91] are very-hard to adapt.

5.5.2 Pricing algorithms

Let $\lambda_{t,k}$ be the (non-negative) dual variables corresponding to constraints (5.9), and η_i be the (free) dual variables corresponding to constraints (5.10).

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:

$$\min \pi_i = -\eta_{\hat{i}} + \alpha \sum_{t \in T} \sum_{\substack{(j,k) \in \\ K \times K}} d_{\hat{i}}^t l_{jk} y_{\hat{i}jk}^t +$$

$$+ \sum_{t \in T} \sum_{k \in K} \left(\beta d_{\hat{i}}^t m_{ik} + d_{\hat{i}}^t \lambda_{t,k}\right) x_{\hat{i},k}^t \tag{5.12}$$

$$\text{s.t.} \sum_{k \in K} x_{\hat{i}k}^t = 1 \qquad \forall t \in T \tag{5.13}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}jk}^t \qquad \forall t \in T \setminus \{1\}, \forall k \in K \tag{5.14}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}kj}^{t+1} \qquad \forall t \in T \setminus \{T\}, \forall k \in K \tag{5.15}$$

$$\boldsymbol{x} \in \{0,1\}, \boldsymbol{y} \in \{0,1\} \tag{5.16}$$

Proposition 2 *The pricing problem* (5.12) - (5.16) *possesses the integrality property.*

In fact, according to Observation 2, constraints (5.13) can be removed for $t > 1$, and constraints (5.14) used to replace $x_{\hat{i}k}^t$ in (5.15) and then removed. The remaining is basically a network flow matrix, which is known to be totally unimodular [58].

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 (5.1) – (5.7). 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 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 5.4 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.

Algorithm 5 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

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 5.

5.5.3 Rounding Heuristics

In order to find good primal bounds, a simple rounding algorithm (presented in Algorithm 6) 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

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

For each time-slot $t \in T$, for each APs i the highest $\tilde{x}_{ik}^t$ 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.

5.5.4 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 5, 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 $c_k^{|T|} = 0$ and updating each $c_k^t = \min_{k^* \in K} c_{k^*}^{t+1} + d_i^{t+1}(\alpha l_{k,k^*} + \beta m_{i,k^*} + \lambda_{t+1,k^*})$. In this way, the cost χ_k^t 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_k^t.

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 5. We perform the following fixes:

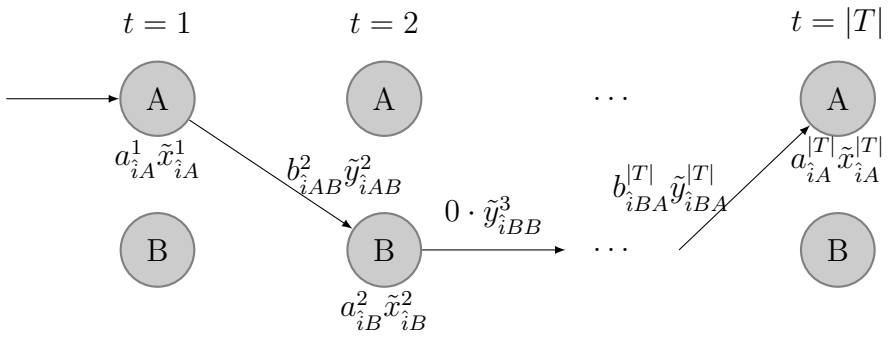


Figure 5.4: DASP - Pricing Problem Structure

Algorithm 6 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 $\hat{x}$}

- 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.

5.5.5 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.

5.5.6 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 (5.1) – (5.5) needs to be enriched, adding constraints

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

to the family (5.4) and constraints

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

to the family (5.5): 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 7. 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.

Algorithm 7 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

5.6 Computational Evaluation

We implemented our algorithms in C++, using CPLEX 12.6 [72] 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 (5.1) – (5.7).

5.6.1 Dataset

We have access to a dataset of real-world mobile traffic demands [92], 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 with 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 [93]. 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.

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.

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 [92]. Demands d_i^t 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 [92], 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 $[-5\%, +5\%]$, to create five perturbed instances from the same single day. While Dataset A aims at stressing our algorithm, Dataset B aims at evaluating its robustness in the presence of noise.

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 [92], 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 5.7.1 (clust) and previously presented in [1]. 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.

In Table 5.1 the instances of our datasets are summarized: for every instance we specify its name (column 'name'), the number of versions ('no. of instances'), the number of APs, MEC facilities and time-slots ('$|A|$', '$|K|$' and '$|T|$', resp.) and the number of variables and constraints of the corresponding model (5.1)–(5.7) ('$|x_{i,k}^t|$', '$|y_{i,k',k''}^t|$' and 'no. of constraints', resp.).

5.6.2 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 5.2a, 5.2b and 5.2c 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 5.6.3). 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 ('# iter') needed to reach convergence and the number

name	no. of instances	$\|A\|$	$\|K\|$	$\|T\|$	$\|x^t_{i,k}\|$	$\|y^t_{i,k',k''}\|$	no. of constraints
1–5	5	1419	10	12	1.70e+05	1.70e+06	3.72e+05
				24	3.41e+05	3.41e+06	7.30e+05
				48	6.81e+05	6.81e+06	1.45e+06
				96	1.36e+06	1.36e+07	2.88e+06

(a) Synthetic Dataset A - Realistic Dataset B

name	no. of instances	$\|A\|$	$\|K\|$	$\|T\|$	$\|x^t_{i,k}\|$	$\|y^t_{i,k',k''}\|$	no. of constraints
clust.	1	1419	10	168	2.38E+06	2.38E+07	5.02E+06
4h				84	1.19e+06	1.19e+07	2.52e+06
3h				56	7.95e+05	7.95e+06	1.68e+06
2h				42	5.96e+05	5.96e+06	1.27e+06
1h				38	5.39e+05	5.39e+06	1.15e+06

(b) Dataset C

Table 5.1: DASP - Dataset Instances Summary

of variables created in the process ('# cols'), the overall CPU time spent for solving the pricing problems ('t^p') and the CPU time required to complete the column generation process ('t'). 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 (column 't').

We set a time limit of two hours to each computation, marking in the tables as 'T.L.' those computations hitting that limit, and marking with 'N.F.' those computations that were not able to retrieve any final solution.

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 $|T| = 48$ and $|T| = 96$ 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. The same solution can be retrieved by the execution of the LP solver of CPLEX for the continuous relaxation of model (5.1)–(5.5), where integrality conditions on variables x and y ((5.6) and (5.7)) are replaced by the conditions $x \in [0, 1]$, $y \in [0, 1]$.

As a further experiment, we solved the continuous relaxation of our model with the LP solver of CPLEX, setting a time limit of two hours, and using the final fractional solution as input for our rounding heuristic to retrieve a feasible solution (Algorithm 6). In Tables 5.3a and 5.3b we include details of these experiments for each instance of datasets A and B, respectively, including the primal and dual bound gaps and the time required to complete the computation: as for the ILP root node, we can notice that the CPU time required by CPLEX LP is up to two orders of magnitude higher than that required by our CG. In more than 25% of the instances, CPLEX is unable to terminate the LP computation within the time limit, while CG always completes the computation.

5.6.3 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 5.4a, 5.4b, 5.4c 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

		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	**N.F.**	**N.F.**	T.L.
	3	0.034%	2.779%	61	43107	9	863	**N.F.**	**N.F.**	T.L.
	4	0.187%	2.528%	61	42692	5	850	**N.F.**	**N.F.**	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	0.539%	0.475%	51	29786	6	130	0%	0.475%	T.L.
	2	0.315%	0.766%	43	28272	3	125	32.042%	0.766%	T.L.
	3	0.249%	0.853%	46	28221	9	129	5.953%	0.853%	T.L.
	4	0.395%	0.692%	49	29320	5	126	0.002%	0.692%	T.L.
	5	0.154%	0.726%	48	29005	4	120	**N.F.**	**N.F.**	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	0.588%	0.297%	50	31706	2	133	0%	0.295%	T.L.
1h	0.097%	0.905%	85	56322	18	953	**N.F.**	**N.F.**	T.L.

(c) Raw Real Demand Dataset C

Table 5.2: DASP Computational Results - CG Root vs. CPLEX Root

$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{DB-z^*}{z^*}$	t
12	1	0%	-1.2%	754
	2	1%	-2.0%	820
	3	1%	-1.4%	589
	4	1%	-1.0%	573
	5	1%	-1.6%	705
24	1	1%	-1.5%	4742
	2	1%	-1.5%	5115
	3	0%	-2.0%	4018
	4	1%	-1.9%	4102
	5	1%	-1.8%	4820
48	1	2%	0.2%[1]	T.L.
	2	-	3.2%[1]	T.L.
	3	4%	2.5%[1]	T.L.
	4	-	-0.1%[1]	T.L.
	5	-	2.4%[1]	T.L.
96	1	25%	21.9%[1]	T.L.
	2	17%	14.3%[1]	T.L.
	3	15%	13.2%[1]	T.L.
	4	17%	14.5%[1]	T.L.
	5	14%	12.2%[1]	T.L.

(a) Synthetic Dataset A

$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{DB-z^*}{z^*}$	t
12	1	4.34%	-0.11%	99
	2	3.67%	-0.11%	89
	3	4.09%	-0.13%	97
	4	4.13%	-0.10%	89
	5	3.85%	-0.10%	89
24	1	0.46%	-0.42%	603
	2	0.50%	-0.27%	613
	3	0.46%	-0.58%	580
	4	0.63%	-0.17%	572
	5	0.58%	-0.21%	598
48	1	0.75%	-0.26%	2225
	2	0.53%	-0.34%	2107
	3	0.73%	-0.36%	2158
	4	0.58%	-0.31%	2250
	5	0.77%	-0.29%	2190
96	1	0.58%	-0.47%	6151
	2	0.15%	-0.76%	5692
	3	-	9.9e07%[1]	T.L.
	4	0.05%	-0.69%	5945
	5	0.01%	-0.73%	6290

(b) Realistic Dataset B

[1] Sub-Optimal LP Solution

Table 5.3: DASP Computational Results - CPLEX LP

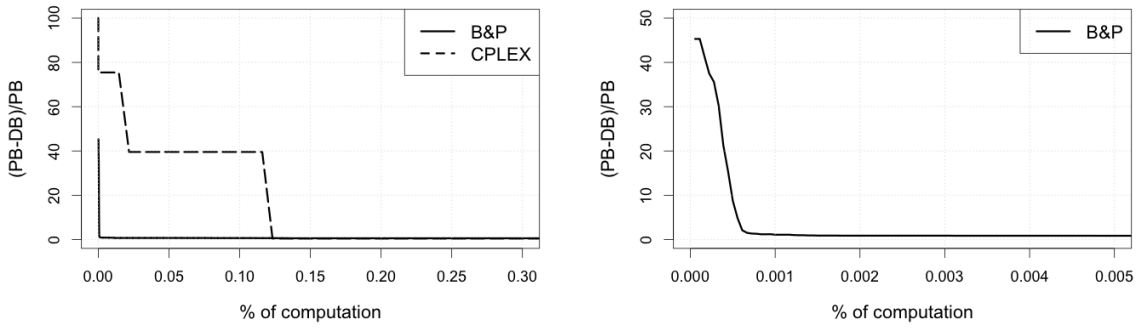
branch-and-bound nodes, for both our BaP and CPX. We report no computing time because, surprisingly, neither BaP nor CPX could bring the duality gap below 0.1% within the time limit, except for instance 4 with $|T| = 12$ of Dataset A, 5 with $|T| = 12$ of Dataset B and 'clust' of Dataset C, that CPX 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 5.5a 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. For instance, corresponding to $x = 0.1$, BaP series reports the primal bound after 10% of the overall number of CG iterations needed to conclude the test, while CPX series reports the primal bound after exploring 10% of the nodes of the full Branch-and-Bound tree needed to conclude the test. 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, which yields negligible difference in the Figure). 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 5.5b: 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.



(a) BaP vs. CPLEX (b) BaP Detail

Figure 5.5: DASP Primal Bound BaP vs. CPLEX

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.

5.7 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 [92], and run actual analytics on it so as to generate the demand profiles. We then feed our optimization models with such profiles, which, ultimately, let us measure the quality of the assignment plans it returns in a practical case.

		BaP			CPLEX		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# 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	**N.F.**	**N.F.**	T.L.
	3	0%	2.779%	168	**N.F.**	**N.F.**	T.L.
	4	0%	2.528%	135	**N.F.**	**N.F.**	T.L.
	5	0%	2.568%	207	272.834%	-	T.L.

(a) Synthetic Dataset A

		BaP			CPLEX		
$\|T\|$	inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# 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	0.462%	0.475%	3133	0%	0.475%	0
	2	0%	0.765%	3193	32.042%	0.766%	0
	3	0%	0.852%	3398	5.953%	0.853%	0
	4	0%	0.691%	3099	0.002%	0.692%	0
	5	0%	0.726%	3299	-	-	T.L.

(b) Realistic Dataset B

	BaP			CPLEX		
inst.	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# nodes	$\frac{PB-z^*}{z^*}$	$\frac{\|DB-z^*\|}{z^*}$	# 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	0.430%	0.297%	2864	0%	0.295%	0
1h	0%	0.905%	370	**N.F.**	**N.F.**	T.L.

(c) Raw Real Demand Dataset C

Table 5.4: DASP Computational Results - BaP vs. CPLEX

5.7.1 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 [94]. 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 $4 \cdot 24 \cdot 7 = 672$ 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 same configuration for longer timeframes, making the optimization possibly oblivious of fast dynamics in the demand.

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 differently in training and in testing:

- in training instances, C_k is set to to $\left(\max_{t\in T}\sum_{i\in A} d_i^t/|K|\right)\cdot 1.05$ in each test. That is, for each instances the capacity is computed separately using its own values of d_i^t and, for example, one-hour time-slot instances are optimized with lower capacity values;
- while testing, to ensure fairness in the comparison, we use a single reference MEC capacity value fixed to $\left(\max_{t\in \tilde{T}}\sum_{i\in A} d_i^t/|K|\right)\cdot 1.10$, considering the 15-minute time-slots demands over the complete dataset.

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:

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

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.

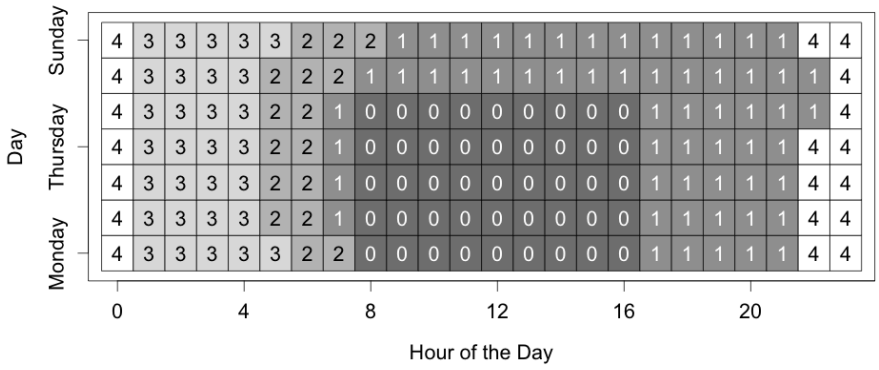


Figure 5.6: Time-Clustering resulting from [1]

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 [1] 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 [1]; *(iii)* we find the intersection of the two cluster sets obtained at the previous point.

Figure 5.6 shows the resulting clustering of one-hour time slots: five typical demand profiles are identified, which can be associated with 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 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.

5.7.2 Experimental evaluation

Following the computational results of Section 5.6, 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. This is the same setting reported and analysed previously in Table 5.2 and labeled as "CG Root". 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 5.7 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 5.5 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.

Finally, we can notice that result of week 6 differs from the other weeks, with higher peak exceeded capacity for every training set. This behavior is associated to the special nature of the week with respect to all others: indeed, week 6 contains data of a national holiday (Easter).

We also experimented on using the full BaP instead of the truncated HBP. Our results are reported in Figure 5.8: no significant change was observed. Similarly, no significant change was observed by using HBP instead of CPLEX on the benchmark clustering.

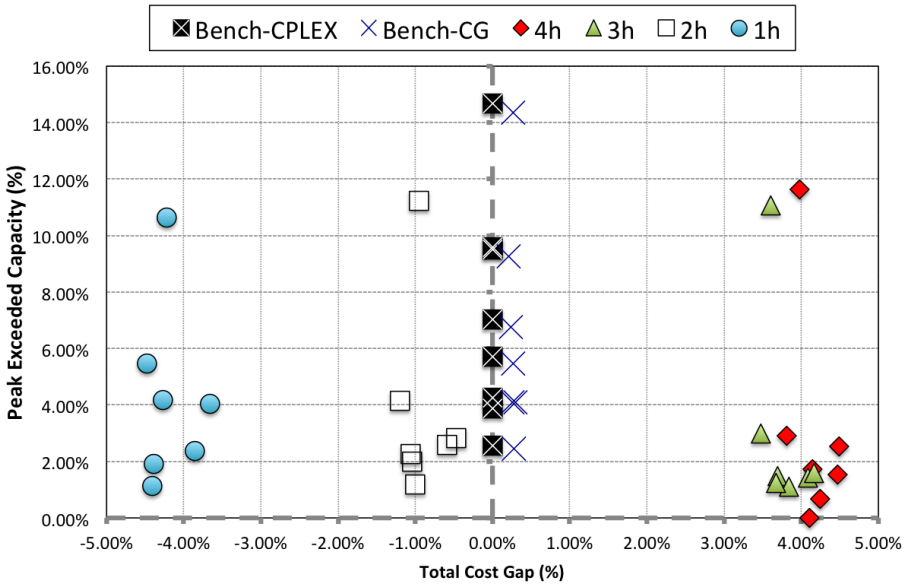
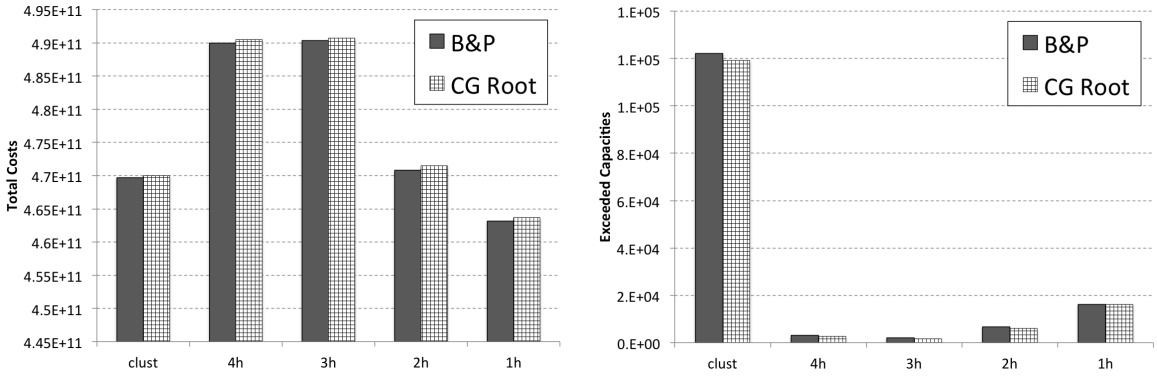


Figure 5.7: DASP 1-Week Plan - Time Aggreg. Comparison

	Week							
inst.	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 5.5: DASP Time-Slot Peak Exceeded Capacity Over Available Capacity



(a) Total Costs (b) Exceeded Capacity

Figure 5.8: Mean Costs And Exceeded Capacity - CG Root vs. BaP

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 5.6: DASP Switching Cost Gap - Acyclic vs. Cyclic

5.7.3 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 5.6 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 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.

5.8 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.

Appendix

5.A Danztig-Wolfe Decomposition of DASP

In Section 5.5 we presented the result of a Dantzig-Wolfe reformulation of DASP model (5.1)–(5.7) presented in Section 5.4. In this Section we give further details on the reformulation, and we present a possible alternative DW reformulation.

With respect to DASP model (5.1)–(5.7) we identify constraints (5.2) as the *complicating* constraints. Let us decompose DASP following DW principles relaxing constraints (5.2). We rewrite DASP as:

$$\min \alpha \sum_{t\in T}\sum_{i\in A}\sum_{\substack{(j,k)\in\\ K\times K}} d_i^t l_{jk} y_{ijk}^t + \beta \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik}^t \tag{5.17}$$

$$\text{s.t.} \sum_{i\in A} d_i^t x_{ik}^t \le C_k \qquad \forall t\in T, \forall k\in K \tag{5.18}$$

$$(x_{ik}^t, y_{ikl}^t) \in \mathcal{P}^i = \{(x_{ik}^t, y_{ikl}^t) : (5.3)-(5.7), {}^{\forall t\in T}_{\forall k\in K}\} \qquad \forall i\in A \tag{5.19}$$

$$x_{i,k}^t \in \{0,1\} \qquad {}^{\forall i\in A,\forall k\in K}_{\forall t\in T} \tag{5.20}$$

$$y_{i,k',k''}^t \in \{0,1\} \qquad {}^{\forall i\in A,\forall t\in T}_{\forall k',k''\in K} \tag{5.21}$$

$\mathcal{P}^i$ represents the feasible region with respect to constraints (5.3)- (5.7). We can replace $\mathcal{P}^i$ with its convex hull $\texttt{conv}(\mathcal{P}^i)$. The relaxation of the integrality conditions (5.20), (5.21) leads to a final bound of (5.17)–(5.19) that is not weaker than the LP relaxation of the original problem (5.1)–(5.7).

Let Ω^i be the set of extreme points of $\texttt{conv}(\mathcal{P}^i)$. Ω^i have a particular combinatorial interpretation: they correspond to all *association* paths p for the AP i, that is the sequence of MEC facilities to which the AP i is assigned in consecutive time-slots and complying with constraints (5.3) – (5.7). For each path $p\in\Omega^i$ let $\tilde{x}_{ik}^{t,p}$ and $\tilde{y}_{ijk}^{t,p}$ be the coefficients encoding the path itself. Each element of $\mathcal{P}^i$ can be represented as a linear combination of points in Ω^i and hence we can replace (5.19) with the following:

$$x_{i,k}^t = \sum_{p\in\Omega^i} \tilde{x}_{ik}\lambda_p \qquad \forall i\in A, \forall t\in T, \forall k\in K \tag{5.22}$$

$$y_{ijk}^t = \sum_{p\in\Omega^i} \tilde{y}_{ijk}^t\lambda_p \qquad \forall i\in A, \forall t\in T, \forall j,k\in K \tag{5.23}$$

with $\lambda_p \ge 0$. Replacing every variable $x_{i,k}^t$ (resp. y_{ijk}^t) with its linear combination

(5.22) (resp. (5.23)) we obtain the master problem (5.8)–(5.10) presented in Section 5.5.

5.A.1 Alternative DW Decomposition of DASP

Several alternative DW decomposition of the DASP model (5.1)–(5.7) are possible, depending on the subset of constraints that are relaxed. In the DW relaxation presented in Section 5.5 we choose constraints (5.2) to be relaxed, which lead to a pricing problem with integrality properties (Proposition 2).

We tested a different variant of DW decomposition of DASP, relaxing constraints (5.4). We rewrite DASP as:

$$\min \alpha \sum_{t\in T}\sum_{i\in A}\sum_{\substack{(j,k)\in\\ K\times K}} d_i^t l_{jk} y_{ijk}^t + \beta \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik}^t \tag{5.24}$$

$$\text{s.t. } x_{ik}^t = \sum_{l\in K} y_{ilk}^t \;\; \forall i \in A, \forall t \in T \setminus \{1\}, \forall k \in K \tag{5.25}$$

$$(x_{ik}^t, y_{ikl}^t) \in \mathcal{P}^t = \{(x_{ik}^t, y_{ikl}^t) : (5.2), (5.3), (5.5), (5.6), (5.7), {}^{\forall i\in A}_{\forall k\in K}\} \;\; \forall t \in T \tag{5.26}$$

$$x_{i,k}^t \in \{0,1\} \;\; \forall i \in A, \forall k \in K, \forall t \in T \tag{5.27}$$

$$y_{i,k',k''}^t \in \{0,1\} \;\; \forall i \in A, \forall k', k'' \in K, \forall t \in T \tag{5.28}$$

$\mathcal{P}^t$ represents the feasible region with respect to constraints (5.2),(5.3), (5.5), (5.6), (5.7). We can replace $\mathcal{P}^t$ with its convex hull $\texttt{conv}(\mathcal{P}^t)$. The relaxation of the integrality conditions (5.27), (5.28) leads to a final bound of (5.24)–(5.26) that is not weaker than the LP relaxation of the original problem (5.1)–(5.7).

Let Ω^t be the set of extreme points of $\texttt{conv}(\mathcal{P}^t)$. Ω^t have a particular combinatorial interpretation: they correspond to a single time assignments of AP to MEC facilities, that can be modelled as a Generalized Assignment Problem with additional constraints. For each set of assignment in time $p \in \Omega^t$ let $\tilde{x}_{ik}^{t,p}$ and $\tilde{y}_{ijk}^{t,t+1,p}$ be the coefficients encoding the assignments itself, and having same behavior of the corresponding variables x_{ik}^t and $y_{ijk}^{t,t+1}$ of the original model. Each element of $\mathcal{P}^t$ can be represented as a linear combination of points in Ω^t and hence we can replace (5.26) with the following:

$$x^t_{i,k} = \sum_{p\in\Omega^t} \tilde{x}_{ik}\lambda_p \qquad \forall i \in A, \forall t \in T, \forall k \in K \tag{5.29}$$

$$y^t_{ijk} = \sum_{p\in\Omega^t} \tilde{y}^t_{ijk}\lambda_p \qquad \forall i \in A, \forall t \in T, \forall j,k \in K \tag{5.30}$$

with $\lambda_p \geq 0$. Replacing every variable $x^t_{i,k}$ (resp. y^t_{ijk}) with its linear combination (5.29) (resp. (5.30)) we obtain the following Master Problem:

$$\min \sum_{t\in T}\sum_{p\in\Omega^t} \beta \sum_{i\in A}\sum_{k\in K} d^t_i m_{ik}\tilde{x}^{tp}_{i,k}\lambda_p + \sum_{\substack{t\in T\\ |t<|T|}}\sum_{p\in\Omega^t} \alpha \sum_{i\in A}\sum_{\substack{(j,k)\in\\ K\times K}} d^t_i l_{jk}\tilde{y}^{t,t+1,p}_{ijk}\lambda_p \tag{5.31}$$

$$\text{s.t.} \sum_{p\in\Omega^t} \tilde{x}^{tp}_{ik}\lambda_p - \sum_{p\in\Omega^{t-1}}\sum_{\substack{(j,k)\in\\ K\times K}} \tilde{y}^{t-1,t,p}_{ijk}\lambda_p = 0 \ , \ \begin{matrix}\forall i\in A,\forall k\in K\\ \forall t\in T\setminus\{1\}\end{matrix} \tag{5.32}$$

$$\sum_{p\in\Omega^t} \lambda_p = 1 \ , \ \forall t \in T \tag{5.33}$$

The corresponding pricing problem can be decomposed for each time-slot $t \in T$. It makes use of free dual variables μ^t_{ik} of constraints (5.32) and free dual variable η_t of constraints (5.33). Given a fixed time-slot $\hat{t}$, the corresponding IP formulation is the following:

$$\min \ -\eta_{\hat{t}} + \sum_{i\in A}\sum_{k\in K} c^{\hat{t}}_{ik} x^{\hat{t}}_{ik} \tag{5.34}$$

$$\text{s.t.} \sum_{k\in K} x^{\hat{t}}_{i,k} = 1 \qquad \forall i \in A \tag{5.35}$$

$$\sum_{i\in A} d^{\hat{t}}_i x^{\hat{t}}_{ik} \leq C_k \qquad \forall k \in K \tag{5.36}$$

$$c^{\hat{t}}_{ik} = \beta d^{\hat{t}}_i m_{ik} + \begin{cases} +\min_{j\in K}\{\alpha d^{\hat{t}+1}_i l_{kj} + \mu^{\hat{t}+1}_{ij}\} & \text{if } \hat{t} = 1 \\ -\mu^{\hat{t}}_{ik} & \text{if } \hat{t} = |T| \\ -\mu^{\hat{t}}_{ik} + \min_{j\in K}\{\alpha d^{\hat{t}+1}_i l_{kj} + \mu^{\hat{t}+1}_{ij}\} & \text{if } 1 < \hat{t} < |T| \end{cases} \qquad \forall i \in A, \forall k \in K \tag{5.37}$$

$$x^{\hat{t}}_{ik} \in \{0,1\} \tag{5.38}$$

This pricing problem does not possess integrality property. As a consequence, its optimal resolution is not straightforward but the LP relaxation bound retrieved at the end of the CG should be tighter than the LP relaxation bound of the original model (5.1)–(5.7).

We performed computational evaluation of this formulation using the same setup presented in Section 5.6, and solving the pricing problem with the ILP general solver of CPLEX. With respect to the first instance of Dataset B with 12 time-slots, while our HBP algorithm retrieves a first heuristic solution in 2 seconds, with an optimality gap of 0.8% (in Table 5.2b), this approach was not able to improve the initial solution after reaching the time limit of 2 hours and the dual bound of the CG was far from the end.

We believe that this formulation is not suited for our instances: (i) our instances contain a high number of APs making optimal resolution of this pricing problem difficult, and (ii) decomposing the problem removing connections of consecutive time-slots lead to a very slow convergence of the CG.

Part III

Predicting User Mobility

Chapter 6

Predicting User Mobility With Network Data

Accurate algorithms require accurate data, and one of the insights coming from the design of MEC Network described in Chapters 3 and 4 is the following: the knowledge of user mobility represents key data for the optimal design of a MEC Network. These are however very hard to retrieve due to both availability and privacy issues. Therefore, in this chapter we face the problem of estimating user mobility given very aggregated data. In particular, we suppose that only mobile access point demands in a time horizon are known, together with the knowledge of some general statistical features of the mobility patterns.

We define the scope of our approach in Section 6.1, while in Section 6.2 we present an extended mathematical programming formulations of our problem, for which we provide a column generation based algorithm in Section 6.3. We experiment on both synthetic datasets, obtained through generative models from the literature, and real world datasets for which ground truth is not available (Section 6.4). Our approach proves to be accurate enough to faithfully estimate mobility on the synthetic datasets, and efficient enough to tackle real world instances (Section 6.5). Finally in Section 6.6 we draw our final conclusions.

Preliminary results were presented in [24].

6.1 Introduction

The knowledge of user trajectories in urban areas is key data in many applications. In our MEC Network context users' movements cause changes to the network load, and this latter must be balanced to optimize the use of resources and to decrease congestion risks. Moreover the knowledge of users movement is also vital for application requiring data to *follow* the user to be accessed in a short time.

The modeling of human mobility is a well studied research field, that the availability of spatiotemporal information of mobile phone users has push forward with a great number of research projects and potential applications [23].

Human trajectories are far from random, as revealed by Gonzalez *et al.* in [95], but rather they show a high degree of temporal and spatial regularity. Indeed, while the distribution of the displacements lengths of an individual can be approximated with a truncated Lévy flight distribution, the probability to return in an already visited location is much higher than that approximated by a Lévy flight. This regularity is explained by the fact that individuals spend most of their time in a small number of locations. Song *et al.* in [96] further investigate the differences between user mobility and traditional random-walk models, such as Lévy flight and continuous-time random-walk. In particular they identify two generic mechanisms which are unique to user mobility:

- exploration: the tendency to explore additional locations, which decreases with observation time;
- preferential return: users show significant propensity to return to locations they visited frequently before, such as their home or workplace.

As mobility traces are not random, and users often return to their previous visited locations, the authors of [97] face the problem of the predictability of user mobility, deducing that an appropriate algorithm could predict up to 93% of user locations on average, thanks to the temporal correlations of the users' displacements.

Different philosophies have been devised to model user mobility, the two most used being based on: (i) the solely traveled distance; and (ii) the so called intervening opportunities [98] (also known as activity-based models), that take into consideration the reasons that lead people to move.

Distance based models assume that mobility is directly deterred by the costs (in time and energy) associated to physical distance. Among these, the gravity model [99], in analogy with Newton's law of gravity, assumes that the number of individuals

that move between two locations per unit time is proportional to the population of the source and destination locations, and decays as a function of the distance between the locations.

On the other hand, the intervening opportunities theory, originally presented in [98], states that a person aims to search for destinations where to satisfy her needs. Hence migration from an origin to a destination is assumed to depend on the number of opportunities to satisfy the underling need that lie closer to the origin than the destination. Implementation of this theory includes models that require the direct knowledge of the human activities [100]. Several proxies to intervening opportunities have been proposed, such as the land use of locations, the *importance* of locations acting as point of interest that attracts individuals [101, 78], the density of population [102, 103], among others.

However, to the best of our knowledge, no law for user mobility has been proved to be *universally* valid for all spatial scales, time periods and urban topologies. In Appendix 6.A we propose a literature review of the human mobility prediction models, differing by: (i) the typology of mobility that they aim to predict, i.e. inter-city commuters mobility or intra-city mobility; (ii) the range of time that they want to predict, as it is assumed that in different time of the day, day of the week and week of the year mobility can change from the normal routine; (iii) the underlying mobility model, i.e. gravity-based or activity-based or a mixture of different models or without a specific model and (iv) the amount of information required to train the predictive model. This last point opens several issues: the need of information about socio-economical and geographic attributes may involve availability and privacy issues.

Main contributions and outline To the best of our knowledge, the majority of human mobility prediction approaches presented in literature, a subset of which are presented in Appendix 6.A, requires a considerable amount of information, such as: complete census data of the considered area, origin-destination matrices, the history of individual movements, social media check-in, among others.

With respect to all these works, our approach requires the knowledge of very low amount of information that can be accessed more easily, that is: (i) the mobile network demand across the day and (ii) the location of the underlying mobile access points, or alternatively the area from which each portion of data is retrieved. As a trade-off, we do not aim to model universal *human* mobility, but rather to model the mobility of the *users* of the mobile network.

Our approach aims to rebuild user mobility from the spatiotemporal fluctuations of network usage; however, linking network load fluctuation to user trajectories is not enough to predict the actual users' mobility: in order to faithfully predict this latter, we have to constraint trajectories to follow statistical features typical to user mobility. To keep as low as possible the amount of information needed to exploit our models, we take into account only aggregated statistical features. In particular we include in our model the following feature, highlighted in [95, 96]: the distribution of the displacements lengths on each individual path can be reliably approximated with well known probability distribution.

In Figure 6.1 we present the sketch of our User Mobility Prediction Framework. As already stated, our prediction algorithm requires three inputs data: we partition the region covered by a mobile network into cells, one for each AP, and we suppose to be given: (a) the adjacency matrix between cells, and (b) the demand in each cell at each point in time, that is the number of users connected to the corresponding AP; the time horizon is split in discrete time-frames; we also assume that an aggregated information about user mobility is given, namely the probability distribution of trajectory lengths.

Our aim is to find an estimate of the *path* of each user, in terms of sequence of cells traversed by the user during the considered time horizon, that explains the network usage fluctuations and fits the chosen statistical feature of mobility. In Figure 6.2 an example of such path is presented: given a region split in cells corresponding to the AP covering areas, we define as *paths-over-time* (paths in the remainder) the sequence of adjacent APs cells, i.e. geographically connected, and which are assumed to be visited by users in *consecutive* time frames (from time $t = 1$ to $t = 4$ in the Figure).

Since demand is usually easy to forecast, e.g. by time series analysis, our methods can be seen in the long term as means of *predicting* the corresponding user mobility. These paths are also meant as a refined way of estimating user *trajectories*, that is pairs of origin and destination for each user.

We model such a problem as that of finding a suitable set of paths-over-time on a time-dependent graph, proposing extended mathematical programming formulations and column generation algorithms: we present a bi-objective linear programming model (Section 6.2), solved through a hierarchical algorithm (Section 6.3). Our problem belong to the class of flows over time problem (also known as *dynamic network flows*) over discrete time with a single commodity, multiple sources and multiple destinations [22]; however, to the best of our knowledge, this is the first

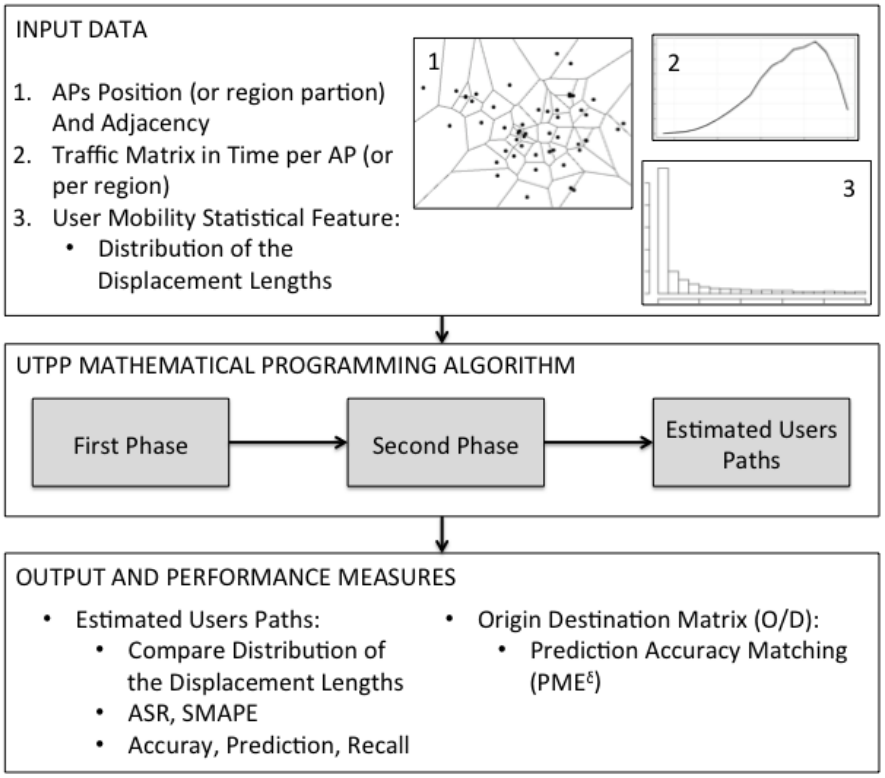


Figure 6.1: User Mobility Prediction Framework

time mathematical programming techniques have been used to link network load fluctuation to user trajectories.

We experiment on both real-world and synthetic datasets (Section 6.4): in order to test effectiveness of our approach we generate synthetic user trajectories through three generative models inspired by literature. We propose several performance measures that evaluate both the quality of predicted origin-destination trajectory matrices and the full paths. Our approach proves to be accurate enough to faithfully estimate mobility on the synthetic datasets, and efficient enough to tackle real world instances; it performs well in a rush hour scenario, that is considering a urban-size area for a limited amount of time with great mobility, while it performs poorly in a commuters mobility scenario, that is considering a long time horizon and long distances (Section 6.5).

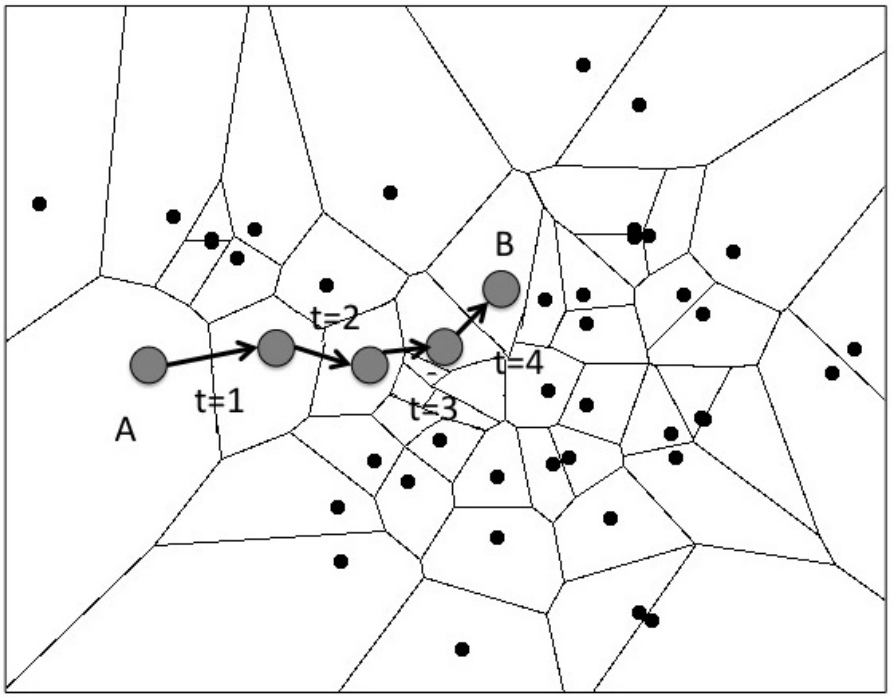


Figure 6.2: User Path-Over-Time

6.2 Formulation

In order to model our network, let $T = \{1, \ldots, |T|\}$ be a set of time-frames within a time horizon and N be a set of APs, each lying at coordinates (x_i, y_i) in a plane that models our urban area. For each time $t \in T$ and AP $i \in N$, let $d_i^t \in \mathbb{N}_0$ be the number of users connected to AP i during time frame t. This set of data is problem specific, and is supposed to be provided by the operator.

Modelling User Trajectories We denote as Ω the set of feasible *paths-over-time* (paths in the remainder), each being a sequence of APs whose cells are adjacent, i.e. geographically connected, and which are assumed to be visited by users in consecutive time frames. Notation-wise, for each $p \in \Omega$, we indicate with $\omega(p, t)$ the AP visited at time t in path p, and we suppose $\omega(p, t)$ to be set to a dummy value "-" if path p starts after, or ends before t. Let $l(p)$ be the total length of each path $p \in \Omega$, that is the sum of distances between consecutive APs in the path. The starting and ending APs of each path (the first and last values of $\omega(p, t)$ which are different

from "-") identify a trajectory; the same trajectory can be identified by many feasible paths. We remark that indeed the same set of trajectories can be explained by multiple sets of paths.

Modelling User Mobility Statistical Feature As stated previously in the Introduction, user trajectories are far from random. Linking network load fluctuation to user trajectories is not enough to predict the actual users' mobility: in order to faithfully predict this latter, we have to constraint trajectories to follow statistical features typical to user mobility. To keep as low as possible the amount of information needed to exploit our models, we take into account only aggregated statistical features. In particular we include in our model the following feature, highlighted in [95]: the distribution of the displacements lengths on each individual path can be reliably approximated with well known probability distribution.

Therefore, let $K = \{1, \ldots, |K|\}$ be a set of traveling distance classes, obtained by partitioning Ω according to the length of its paths. For each $k \in K$, let l_k (resp. l_{k-1}) be the upper (resp. lower) bound on the length of each path in class k, with $l_0 = 0$; let also $n_k \in \mathbb{Z}^*$ be the number of users whose path is in class k. This set of data is *not* problem specific: it can be estimated without any sensitivity issue.

Summarizing, from an application point of view, we assume: (i) AP locations coordinates (x_i, y_i) and expected number of users d_i^t to be given, e.g. by a telecommunication service provider; (ii) Ω to be easily definable, e.g. by Voronoi tessellations and street maps, and (iii) l_k and n_k to be estimated by previous knowledge on users travel distance distributions like [95, 96].

Variables Our aim is to assess how many users are expected to follow a certain path over our time horizon, that we indicate as x_p for each $p \in \Omega$. We also consider the possibility that users enter or quit the system, or that simply data d_i^t is affected by noise, allowing a positive (resp. negative) correction $\bar{\epsilon}_i^t$ (resp. $\underline{\epsilon}_i^t$) for each $i \in N, t \in T$. As we are dealing with averages of quantities or, from a different point of view, of expected values, we can relax integrality constraints on our variables and allow them to take non-negative real values.

Our primary objective is to find a setting of the variables that explains our data with minimum absolute value correction, that is to find user trajectories that best explain network load fluctuations. We define this problem as the User Trajectories Prediction Problem (UTPP):

We propose and compare two options to model the link between paths and

network load: a flow conservation and a demand conservation formulation.

Flow conservation formulation We optimize the following Linear Programming (LP) model:

$$\min \varepsilon = \sum_{t\in T}\sum_{i\in N}(\bar{\epsilon}_i^t + \underline{\epsilon}_i^t) \tag{6.1}$$

$$\text{s.t. } d_i^t = d_i^{t-1} + \sum_{j\in N}\sum_{\substack{p\in\Omega \\ |\omega(p,t-1)=j \\ \wedge\omega(p,t)=i}} x_p - \sum_{j\in N}\sum_{\substack{p\in\Omega \\ |\omega(p,t+1)=j \\ \wedge\omega(p,t)=i}} x_p + \bar{\epsilon}_i^t - \underline{\epsilon}_i^t \qquad \begin{matrix}\forall i\in N, \\ \forall t\in T\setminus\{t_0,T\}\end{matrix} \tag{6.2}$$

$$\sum_{\substack{p\in\Omega \\ |l(p)<\bar{l}_k}} x_p \geq \sum_{k'\leq k} n_{k'} \qquad \forall k\in K \tag{6.3}$$

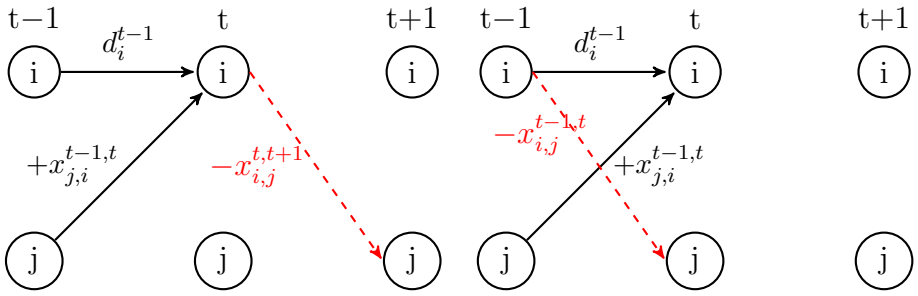
$$x_p \geq 0\ ,\ \bar{\epsilon}_i^t \geq 0\ ,\ \underline{\epsilon}_i^t \geq 0 \tag{6.4}$$

We consider a solution to be feasible if it respects three classes of constraints: (i) constraints (6.2) represent the flow conservation constraints for the mobile traffic demand in time t: demand d_i^t at node i at time i is given by the demand in the same node in the previous time-frame $t-1$, plus the demand incoming in the node in time range $(t-1,t)$, minus the demand leaving the node in time-range $(t,t+1)$; (ii) constraints (6.3) imply that number of users migrating with a traveling distance in class $k\in K$ meet at least the expected number of users migrating in that class $n_k$we constrain the cumulative values rather than the values in each single class; (iii) finally constraints (6.4) impose the non-negativity of the variables.

We name the model composed by equations (6.1)-(6.4) as *flow conservation variant* of the UTPP (f-UTPP in the remainder).

Demand conservation formulation Given that the flow we consider represents demand changes in time, we propose a variant of the flow-conservation constraints (6.2), replacing them with the following:

$$d_i^t = d_i^{t-1} + \sum_{j\in N}\sum_{\substack{p\in\Omega \\ |\omega(p,t-1)=j \\ \wedge\omega(p,t)=i}} x_p - \sum_{j\in N}\sum_{\substack{p\in\Omega \\ |\omega(p,t)=j \\ \wedge\omega(p,t-1)=i}} x_p + \bar{\epsilon}_i^t - \underline{\epsilon}_i^t \quad \begin{matrix}\forall i\in N, \\ \forall t\in T\setminus\{t_0\}\end{matrix} \tag{6.5}$$



(a) Flow Conservation (b) Demand Conservation

Figure 6.3: UTPP Variants of User Trajectories - Network Loads Link

in which demands d_i^t of node i at time t are given by the demands in the same node in the previous time-frame $t-1$, minus the demands that leave the node in the range of time $(t-1,t)$, plus the demands that arrive in the nodes in the range of time $(t-1,t)$. In Figure 6.3 differences of the two different conservation constraints are presented graphically.

We name the model composed by equations (6.1), (6.3), (6.4), (6.5) as *demand conservation variant* of the UTPP (d-UTPP in the remainder).

Result. As encoded by our decision variables, the output obtained by optimizing our models corresponds to the number x_p of users whose path over time is $p \in \Omega$. For our prediction task, we draw an aggregated result, computing for each $i \in N$ and $j \in N$ the value F_{ij} as the sum of x_p variables corresponding to paths starting in i and ending in j, that is the predicted number of users on the trajectory from i to j.

6.2.1 Hierarchical bi-objective approach

Once an optimal ε value is found, we adopt a hierarchical bi-objective approach, and as secondary objective we try to match the path lengths distribution as close as possible. That is, we minimize the maximum difference between the number of users migrating on paths of each class k and the expected ones n_k, fixing the total amount of absolute value correction to use. The modified LP model is:

$$\min \eta \tag{6.6}$$

$$\text{s.t. } (6.3), (6.4), (6.5)$$

$$-\sum_{\substack{p\in\Omega \\ |l(p)\in[\underline{l}_k,\bar{l}_k)}} x_p - n_k \geq -\eta \qquad \forall k \in K \tag{6.7}$$

$$\sum_{i\in N}\sum_{t\in T\setminus\{t_0\}} \bar{\epsilon}_i^t + \underline{\epsilon}_i^t \leq \varepsilon \tag{6.8}$$

$$\eta \geq 0 \tag{6.9}$$

The new objective (6.6) aims to minimize the maximum difference between the predicted and expected number of users in a traveled distance class, represented by the variable η valorized by constraints (6.7). Constraint (6.8) states that the total amount of value correction change cannot exceed the optimal amount found by the primary objective, possibly *moving* the correction to different nodes and times.

A complete notation table for UTPP model can be found in Table 6.1.

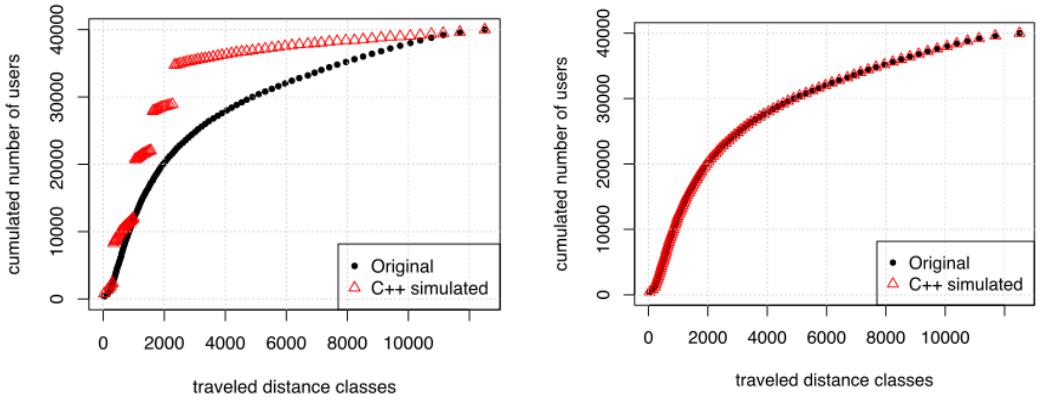
The rationale for the use of this hierarchical model is given by the laminar structure of constraints (6.3) which generates solutions presenting a poor fit of the probability distribution of trajectories lengths. In order to satisfy those cumulative constraints, the model may generate short paths associated to a number of users high enough to satisfy the constraints for several successive classes in K. This behavior is clear in Figure 6.4a, showing the expected Cumulative Distribution Function (CDF) of the traveled distances with black circles and the CDF predicted by the first stage of our model with red triangles: the model associates to short distances a high number of users, such that cumulative constraints (6.3) are satisfied. In Figure 6.4b the same fit is presented after the second stage of our model: a significant improvement can be observed.

6.3 Algorithms

Both stages of our model require to solve only LPs. However, as the cardinality of Ω grows combinatorially, it is computationally infeasible to solve the problem directly. Instead we perform column generation [70] on the set of variables x_p. That is, we consider a restricted LP in which Ω is replaced by a small subset $\bar{\Omega}$, we solve it, we collect the values of optimal dual variables, and we search for variables x_p of

Sets	
T	set of time frames in a time horizon
N	set of two dimension coordinates points of access points
U	set of users in the system
K	classes of travelling distances
$\Omega = \{(t,i),(t+1,j), \ldots,(t+k,n)\}$	set of feasible trajectory paths, composed by pairs (times, locations)
$E \subseteq (N \times N)$	adjacent pair of locations nodes
Parameters	
$d_{i,j}$	distance between pair of nodes $(i,j) \in E$ [km]
d_i^t	demands in terms of number of users in access point i at time t
$(\underline{l}_k, \bar{l}_k)$	range of distance characterizing a distance traveling class k
n_k	expected number of users moving with distance in traveling class k
$\omega : \Omega \times T \to N$	function retrieving the location of a path p in time t
$l : \Omega \to \mathbb{R}$	function retrieving the total length of a path p
Variables	
$x_p \in \mathbb{N}$	number of user moving using path $p \in \Omega$
$\bar{\epsilon}_i^t \in \mathbb{Z}$	number of users entering the system in access point i at time t
$\underline{\epsilon}_i^t \in \mathbb{Z}$	number of users quitting the system in access point i at time t
ε	total number of value correction to minimize in (6.1)
η	min max support variable in (6.6)-(6.7)

Table 6.1: UTPP - Notation Table



(a) 1^{st} stage fit (b) 2^{nd} stage fit

Figure 6.4: Traveled Distance Probability Fitting

negative *reduced cost* whose corresponding path p is therefore not in $\bar{\Omega}$. If no such a variable can be found, the current solution is optimal for Ω as well, otherwise $\bar{\Omega}$ is enlarged with the paths corresponding to the variables found, and the process is iterated.

Let us consider d-UTPP variant: for the primary objective problem, let λ_i^t be the dual variables associated to constraints (6.5) and let μ_k be the dual variables associated to constraints (6.3). The reduced cost of a variable x_p is:

$$\bar{c}_p = \sum_{\substack{t \in T \\ |\omega(p,t) \neq -}} (\lambda^t_{\omega(p,t-1)} - \lambda^t_{\omega(p,t)}) - \sum_{\substack{k \in K \\ |l(p) > \bar{l}_k}} \mu_k \tag{6.10}$$

For each $k \in K$, the search for the most negative reduced cost variable encoding a path in class k can be mapped into a *resource constrained minimum cost path* problem in a time-expanded directed graph $G = \{N', A\}$ [104]. The set of nodes N' is composed by each pair (i, t) of AP $i \in N$ and time frame $t \in T$, together with two additional dummy nodes acting as origin and destination; i.e. $N' = (N \times T) \cup \{(o, t_{-1}), (d, t_{T+1})\}$. The set A includes one arc $(i, t-1; j, t)$ connecting nodes $(i, t-1)$ and (j, t) if and only if the cells of APs i and j are adjacent. The dummy origin has an outgoing arc to every other node except the dummy destination, and no incoming arc. The dummy destination has an incoming arc from every other node except the dummy origin, and no outgoing arc. Indeed, the graph node is organized in layers, one for each time frame; paths in G can only be composed by nodes of

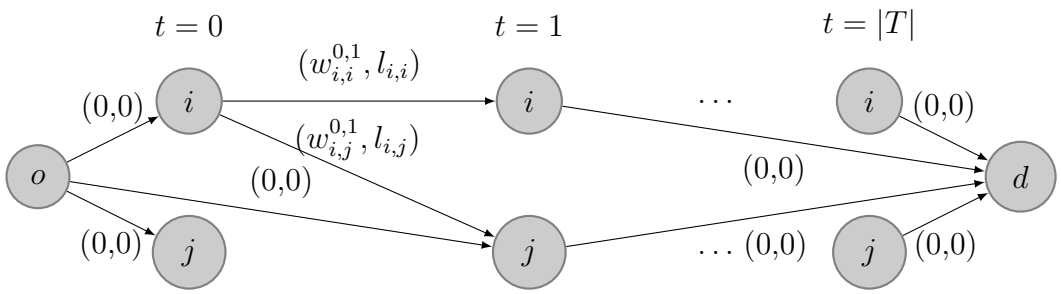


Figure 6.5: Time-Expanded Directed Graph G

different layers, and by arcs connecting one layer with the subsequent one. Modeling of waiting decisions is included, as represented by arcs $(i, t-1; i, t)$. An example of graph G is presented in Figure 6.5.

Dual variables give weights to nodes of the graph $i \in N$; we reformulate the problem to a graph with negative weights on arcs. Each arc $(i, t-1; j, t)$ has a weight related to the value of the dual variables of the constraints in the master in which the link is used, i.e. $w^{t-1,t}_{i,j} = \lambda^t_i - \lambda^t_j$, and lengths $l_{i,j} = \|(x_i, y_i) - (x_j, y_j)\|$, except those incident to either the origin $(o, 0)$ or the destination $(d, T+1)$, whose cost and length are set to 0.

A feasible path in G belonging to traveling distance class $k \in K$ starts from the origin node $(o, 0)$, ends in the destination node $(d, T+1)$ and the sum of its arc lengths falls into the range $[\underline{l}_k, \bar{l}_k)$ We aim to find for each class of traveling distance $k \in K$ a path in the graph G that respect the length given by the class and with negative reduced cost or proving that none exist.

The pricing problem can be formulated as an integer linear program as follows.

Integer Programming Pricing Model The pricing problem can be modeled as an integer programming model, having a binary variable $y^{t-1,t}_{i,j}$ for each arc in graph G with value 1 if arc (i, j) is used in time frame $(t-1, t)$, 0 otherwise. Formally:

$$\min_{k \in K} \sum_{t \in T \setminus \{0\}} \sum_{(i,j) \in N \times N} y_{i,j}^{t-1,t} w_{i,j}^{t-1,t} - \mu_k \tag{6.11}$$

$$\text{s.t.} \sum_{t \in T} \sum_{i \in N} y_{o,j}^{-1,t} = 1 \tag{6.12}$$

$$\sum_{t \in T} \sum_{i \in N} y_{j,d}^{t,T+1} = 1 \tag{6.13}$$

$$\underline{l}_k \leq \sum_{t \in T \setminus \{0\}} \sum_{(i,j) \in N \times N} y_{i,j}^{t-1,t} d_{i,j} \leq \bar{l}_k \tag{6.14}$$

$$\sum_{j \in N \cup \{o\}} y_{j,i}^{t-1,t} = \sum_{j \in N \cup \{d\}} y_{i,j}^{t,t+1} \qquad \forall i \in N, \forall t \in T \tag{6.15}$$

$$y_{i,j}^{t',t''} \in \{0,1\} \qquad \forall (i, t-1; j, t) \in A \tag{6.16}$$

Objective function (6.11) aims at minimizing the total cost, composed by the weights of the arcs belonging to the chosen path and the dual variable μ_k of the traveling distance class $k \in K$ to which the path belong. Constraints (6.12) and (6.13) imply respectively that a single unit of flow exits the dummy origin at time -1 to any node in any time frame (except the destination), and that a single unit of flow enters the dummy destination at time T+1 coming from any node in any time frame (except the origin). Constraint (6.14) links the total length of the path to the distance range given by class $k \in K$. Constraints (6.15) ensure flow conservation at each point of the network at each time frame. Finally constraints (6.16) define the variables and their possible values. Notation of this ILP model is presented in Table 6.2.

ILP model (6.11)–(6.16) can be potentially sovled with general purpose tool. However, such an approach is viable only for very small instances. Instead, we devise an ad-hoc dynamic programming algorithm, that is able to optimize over all classes simultaneously, presented in the following section.

Dynamic programming approach The pricing model to generate new paths can be conveniently tackled with a dynamic programming approach that is able to optimize over all classes simultaneously.

Our graph G is directed and layered, there is no possibility for cycles and all paths are elementary.

Sets	
$G = (N', A)$	time-expanded node graph
N'	nodes of graph G, $N' = (N \times T) \cup \{(o, t_{-1}), (d, t_{T+1})\}$
A	arcs of graph G, in the form $(i, t-1; j, t)$
Parameters	
$w_{i,j}^{t-1,t}$	weight of arc between nodes i and j in consecutive times $t-1, t$
λ_i^t	dual variables associated to constraints (6.5) for d-UTPP variant
θ_i^t	dual variables associated to constraints (6.2) for f-UTPP variant
μ_k	dual variables associated to constraints (6.3)
Variables	
$y_{i,j}^{t-1,t} \in \mathbb{B}$	has value 1 if arc $(i, t-1; j, t) \in A$ belong to the chosen path

Table 6.2: UTPP Algorithms Notation Table

We exploit the layered directed graph G where each node i in layer t contains *states* of the dynamic programming execution, while arcs in the graphs represent state transitions. Different states, associated with the same node (i, t), correspond to different feasible paths p reaching i at time t. They differ by: (i) the sum of arcs lengths L of the path; and (ii) the sum of the weights C of the traversed arcs. Hence states can be represented by a label $(C, L, (i, t))$. The length L in each state can be seen as a value of consumption of a resource whose total availability is $\mathbb{L}$, where $\mathbb{L} = \bar{l}_k$ for each given class $k \in K$.

The main steps of the algorithm are the following:

- **Initialization.** We create a single starting label $(-\sum_{k \in K} \mu_k, 0, (i, t))$ for each $i \in N, t \in T$;
- **Propagation.** We proceed layer by layer and node by node, that is, for each $t \in T$ and for each $i \in N$, we iteratively *select* each label $(C, L, (i, t))$ and *extend* it to all the nodes $(j, t+1)$ having $(i, t; j, t+1) \in A$, creating a new label $(C', L', (j, t+1))$ for each of them that has $L' = L + l_{i,j}$ and

$$C' = C + w_{i,j}^{t,t'} + \sum_{\substack{k \in K \\ |L > \bar{l}_k}} \mu_k - \sum_{\substack{k \in K \\ |L + l_{i,j} > \bar{l}_k}} \mu_k \tag{6.17}$$

The creation of labels having $L' \geq l_{|K|}$ is skipped, as encoding paths which are infeasible for all classes.

- **Fathoming.** The efficiency of the dynamic programming algorithm heavily relies upon the possibility of fathoming feasible states that cannot lead to an optimal solution, which results in applying *dominance rules* between pairs of labels. After treating each label we perform the following check: if any label $(C'', L'', (j, t+1))$ has already been created, having $C'' \leq C'$ and $L'' \leq L'$, at least one inequality being strict, then $(C', L', (j, t+1))$ is fathomed; similarly, if $C'' \geq C'$ and $L'' \geq L'$, at least one inequality being strict, then $(C'', L'', (j, t+1))$ is fathomed.
- **Stopping.** We stop when all pairs (i, t) have been considered. All labels whose cost C is negative encode paths of negative reduced cost.

We remark that, given the laminar structure of constraints (6.3), this aggregated dynamic programming algorithm is able to produce in a single run the labels of all non dominated paths for each class k; a formal proof is omitted. This allows us on one hand to improve efficiency, since only one resource constrained minimum cost path problem needs to be solved at each column generation iteration, and on the other hand to obtain an effective multiple pricing strategy, that consists in enlarging the set $\bar{\Omega}$ at each column generation iteration with the minimum reduced cost path for each class $k \in K$, if any negative reduced cost exists.

The overall dynamic programming procedure is formalized in Algorithm 8. For each state (i, t) we indicate with $\Gamma_{(i,t)}$ the set of its associated labels, with $l_{i,t}$ a generic single label and with $\Delta_{(i,t)}$ the set of successors of the state. *Extend(l(i,t), (j,t'))* is the extension procedure: it extends a label for (i, t) specified as a first argument to the state (j, t') specified as a second argument; this procedure checks the resource constraints and produces only feasible states. *EFF(Γ , l)* is the procedure that inserts label l into set Γ applying the dominance rules. Finally with $P(l)$, $L(l)$ and $C(l)$ we indicate respectively the path, the length of the path and the cost of the path of label l.

Path generation for f-UTPP For the f-UTPP variant the reduced cost of a variable x_p is given by the dual variables θ_i^t associate to constraints (6.2):

$$\bar{c}_p = \sum_{\substack{t \in T | \\ \omega(p,t+1) \neq '' - ''}} \theta^t_{\omega(p,t)} - \sum_{\substack{t \in T | \\ \omega(p,t-1) \neq '' - ''}} \theta^t_{\omega(p,t)} - \sum_{\substack{k \in K | \\ l(p) > \bar{l}_k}} \mu_k \tag{6.18}$$

Components θ_i^t cancel each other out, except those relative to first and last visited node, which contributes respectively positively and negatively. The equation

Algorithm 8 d-UTPP Dynamic Programming Pricing

for all $t \in T$ **do**
  for all $n \in N$ **do**
    for all $l_{t,n} \in \Gamma_{(t,n)} \cup \{(p = \{(-1, o), (t, n)\}, 0, 0, (t, n))\}$ **do**
      for all $(t', n') \in \Delta_{(t,n)} \cup \{(T + 1, d)\}$ **do**
        $l_{t',n'} \leftarrow Extend(l_{t,n}, (t', n'))$
        $\Gamma_{(t',n')} \leftarrow EFF(\Gamma_{(t',n')}, l_{t',n'})$
      end for
    end for
    $\Gamma_{(t,n)}$ references can be deleted
  end for
end for
for all $k \in K$ **do**
  $l^k_{min} = \arg\min_{l_{T+1,d} \in \Gamma_{(T+1,d)}} \{ C(l_{T+1,d}) \mid \underline{l}_k \leq L(l_{T+1,d}) \leq \bar{l}_k \ \wedge \ C(l_{T+1,d}) < 0\}$
  $p_k \leftarrow P(l^k_{min})$
end for

can be written equivalently:

$$\bar{c}_p = \sum_{\substack{t \in T| \\ \omega(p,t-1) = ''-''}} \theta^t_{\omega(p,t)} - \sum_{\substack{t \in T| \\ \omega(p,t+1) = ''-''}} \theta^t_{\omega(p,t)} - \sum_{\substack{k \in K| \\ l(p) > \bar{l}_k}} \mu_k \tag{6.19}$$

In the dynamic programming algorithm that generate paths, the weights of the layered directed graph change to $w^{t-1,t}_{i,j} = \theta^{t-1}_i - \theta^t_j$. The algorithm remain unchanged.

Second Stage We adapt our algorithm to the second stage of our bi-objective model as follows. Since dual variables ν_k of constraints (6.7) need to be taken into account, the pricing objective function becomes:

$$\min_{k \in K} \sum_{t \in T \setminus \{0\}} \sum_{(i,j) \in N \times N} y^{t-1,t}_{i,j} w^{t-1,t}_{i,j} - \mu_k + \nu_k \tag{6.20}$$

while other constraints in the pricing model remains unchanged. The dynamic programming algorithm presented previously can still be used, changing the formula that updates the label cost (6.17):

$$C' = C + w_{i,j}^{t,t'} + \sum_{\substack{k \in K \\ |L \leq \bar{l}_k}} \mu_k - \sum_{\substack{k \in K \\ |L + l_{i,j} \leq \bar{l}_k}} \mu_k - \sum_{\substack{k \in K \\ |L \in [\underline{l}_k, \bar{l}^k]}} \nu_k + \sum_{\substack{k \in K \\ |L + l_{i,j} \in [\underline{l}_k, \bar{l}^k]}} \nu_k \tag{6.21}$$

Formally, since the structure of constraints (6.7) is not laminar anymore, the dominance rules need to be slightly relaxed to take into account of the contribution of the new dual variables. In our implementation, instead, we found it computationally useful to keep the original rules and resort to heuristic pricing. As discussed in Section 6.5 the routine obtained in this way proved to be able to produce high quality solutions with limited effort.

6.4 Experimental Analysis Methodology

Unfortunately, the real world origin-destination matrices we have access to (presented in Chapter 4) include very limited data on the fine-grained user trajectories, to be used as ground truth. Since obtaining a statistically significant testbed from these data is not possible, in order to test both the computational viability and the prediction accuracy of our methods, we create synthetic datasets as described in the next subsections.

In Figure 6.6 we sketch the experimental analysis methodology. First, we draw APs coordinates at random; we generate instances as collections of user paths-over-time on this set of APs, to which we refer to as the *original* paths. We propose three generative models of original paths (Subsection 6.4.1): (i) an *ad-hoc* model that generates the paths in the same way our algorithm generates paths in the column generation process, described in Section 6.3; (ii) a model based on the definition of Point of Interests (POI) from which users starts their trips and to which are attracted; and (iii) the generative model defined in [96].

Using the generated original paths we compute: (i) the network load in time, i.e. the number of users d_i^t connected to each AP $i \in N$ at time $t \in T$; (ii) the details l_k and n_k of path length classes $k \in K$. These data are used as sole input of our methods.

In order to test effectiveness of our algorithm, we have selected a probabilistic predictive model from literature as benchmark (Subsection 6.4.2). In particular we have selected the intra-urban mobility model presented in [103], that we classify as

a gravity model. We used the original paths to train this model, that retrieve an estimation of the origin-destination matrix.

The full collection of *original* paths is kept only for cross-checking (as post-processing) the quality of *predicted* paths, that are those produced as output solutions of our methods.

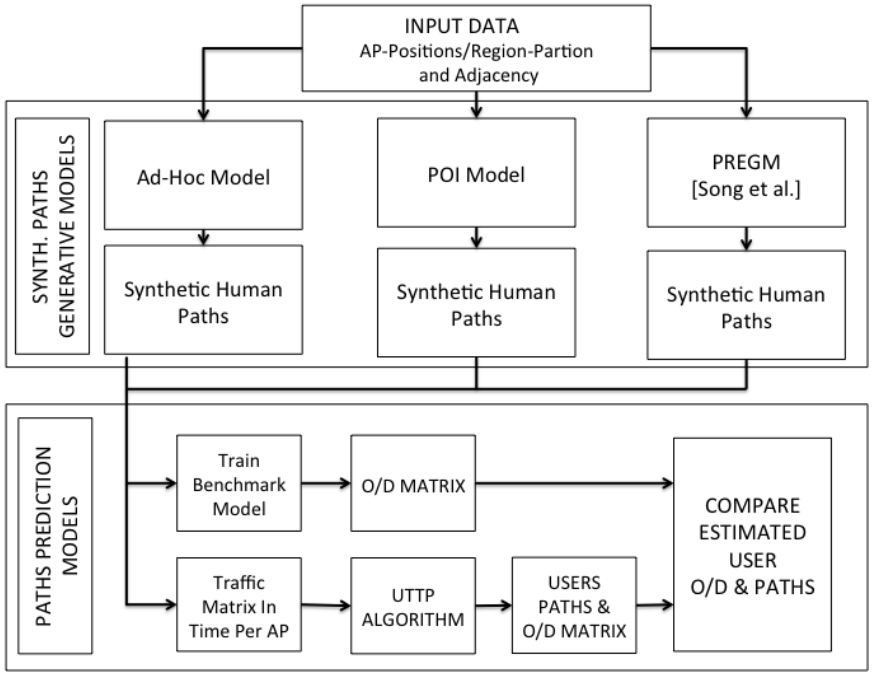


Figure 6.6: Experimental Analysis Methodology

6.4.1 Generative Models

Ad-Hoc Generative Model

The ad-hoc model mimics the creation process of the paths of the dynamic programming algorithm presented in Section 6.3: (i) we ask for a set of users U as input of the generator; (ii) for each user $u \in U$ we choose an origin AP o uniformly at random and a displacement distance r from a power law distribution with exponent $\alpha = 0.2$; (iii) as destination node d we choose uniformly at random among those whose distance from origin o is in the range $[0.8r, 1.2r]$. A path from o to d is then

created as a minimum hops path in the adjacency graph of APs, where the adjacency is defined by the Voronoi tessellation.

We assume that one hop is made in each time frame; the starting time frame is chosen uniformly at random in such a way that the complete path fits in the time horizon. In Algorithm 9 the pseudo-code of the ad-hoc model is presented.

Algorithm 9 UTPP - Ad-Hoc Mobility Generation

for all $u \in U$ **do**
  $l_{\text{init}} = getInitialRandomLocation()$
  $\Delta r = P(\Delta r \in [r_{\min}, r_{\max}]) \sim |\Delta r|^{-1-\alpha}$
  $l_{dest} = randomLocationAtDistance(l_{init}, \Delta r)$
  $p_u = ShortestPath(l_{init}, l_{dest})$
end for

Points of Interest Generative Model

As second generative model, we exploit the concept of the presence in a city of Points of Interest (POI) that attract users (such as industrial and commercial areas) and from which users start moving (such as residential areas), as proxies for the intervening opportunities such as presented in [98, 101]. Let us define the subset of POI $D \subseteq N$ which attracts mobility and the subset $S \subseteq N$ which generates mobility, drawn uniformly at random from the sets of nodes N. We estimate the number of users n_i^T that will be in cell i at the end of the time-horizon with $|D|$ pairs of normal distributions centered in the coordinates of each POI in D. In the same way we estimate the number of users $\bar{n}_i$ that start their movements in cell i with $|S|$ pairs of normal distributions centered in the coordinates of each POI in S.

We define *attractiveness* of an AP i as $a_i = n_i^T / \sum_k n_k^T$, i.e. as the percentage of users expected to be in cell i at the end of the time-horizon. Following the gravity model as the one presented in [103], we define the probability for a user to move from cell i to cell j to be proportional to the attractivity of the destination cell j and inversely proportional to the distance between the cells d_{ij}, formally:

$$P_{i \to j} = \frac{\frac{a_j}{d_{ij}}}{\sum_k \frac{a_k}{d_{ik}}} \tag{6.22}$$

For each cell i and for each users starting in cell i, $u \in \bar{n}_i$, a destination cell j

is drawn with the probability distribution given by (6.22). The actual mobility path is given by the shortest path in the adjacency graph of the APs between cell i and cell j, where the adjacency if defined by the Voronoi tessellation.

Preferential Return and Exploration Generative Model

As third generative model we implement the model described by Song *et al.* in [96], described in Algorithm 10. This model is based two mechanisms unique to user mobility: preferential return and exploration. If a user moves, she will prefer an already visited location, and among them the more visited location has more probability to be visited again. If the user moves to a new location, the destination is chosen in the same way as in ad-hoc generative model described in Algorithm 9.

Going into detail: given the number of distinct locations visited by a user S, when a user moves she will choose as destination with probability $P_{new} = \rho S^{-\gamma}$ a new location and with probability $P_{ret} = 1 - P_{new}$ an already visited location. If the user chooses an already visited location, the actual location l is drawn with probability $\Pi_l = f_l = \frac{m_l}{\sum_{l'} m_{l'}}$, where m_l is the number of times that location l where visited and hence f_l is the percentage of times location l were visited (the frequency of visit). Parameters ρ and γ defines the willing of a user to explore new locations. In our experiments γ is constant for all users with value 0.21, while ρ was drawn for every user from a normal distribution with mean $\mu_\rho = 0.56$ and standard deviation $\sigma_\rho = 0.1$.

For each user $u \in U$ the initial location is drawn uniformly at random from the sets of nodes N, as well as the initial time is drawn uniformly at random from the sets of times T. Movements occur after a waiting time drawn from a power law probability distribution with exponent $\alpha = 0.20$, truncated to a maximum value of 17 hours (suggested by [96]). The generation of new destinations end after reaching the time t_{end}, that we fixed to 25 hours of simulated trajectory. The final complete trajectory is given by the union of all paths from origin to destinations, that are composed by the shortest path in the adjacency graph of the APs.

We will refer to this generative model as PREGM.

6.4.2 Benchmark Model

In order to test effectiveness of our algorithm, we have selected a probabilistic predictive model from literature as benchmark.

Algorithm 10 Preferential Return and Exploration Generative Model

$S = 1, l_0 = P_{\text{init}}(N), t_0 = t_{\text{init}}, d = 0$
while $t_d \leq t_{\text{end}}$ **do**
  $d = d + 1$
  $P_{\text{new}} = \rho S^{-\gamma}, P_{\text{ret}} = 1 - P_{\text{new}}$
  if $P_{\text{unif}}(0, 1) \leq P_{\text{new}}$ **then**
    $\Delta r = P(\Delta r \in [r_{\min}, r_{\max}]) \sim |\Delta r|^{-1-\alpha}$
    $l_d = randomLocationAtDistance(l_{d-1}, \Delta r)$
    $S = S + 1$
  else
    $l_d = P(\Pi_l = f_l = m_l / \sum_l m_l(d))$
  end if
  update Π_l given new displacement l_d
  $p_d = ShortestPath(l_{d-1}.l_d)$
  $t_{\min} = travelTime(p_d)$
  $\Delta t = P(\Delta t \in [t_{\min}, t_{\max}]) \sim |\Delta t|^{-1-\beta}$
  $t_d = t_{d-1} + \Delta t$
end while
return $\{p_1, \ldots, p_d\}$

We select the intra-urban mobility model presented in [103], that we classify as a gravity model. In order to predict users mobility, it takes into consideration specific features of the area under analysis. In particular, this model splits a city in regions and considers the density of population Q_i of the region i together with the distances d_{ij} between regions i and j to build the probability $P(i \to j)$ of a user to move from region i to region j. Formally:

$$P(i \to j) = \bar{Q}_i \frac{Q_j / f(d_{ij})}{\sum_{k \neq i} Q_k / f(d_{ik})} \tag{6.23}$$

where $\bar{Q}_i$ is the normalized density of population in region i and $f(d_{ij})$ is a function of the distance between the regions i and j, defined as $f(d_{ij}) = d_{i.j}^{\sigma}$. Supposing that a reliable estimate of the number T_{ij} of actual trips between regions i and j is available for *training*, the optimal value for parameter σ can be obtained through a maximum likelihood computation:

$$\sigma = \arg\min_{\sigma} \left(- \sum_{\substack{i,j \in S \mid \\ i \neq j}} T_{ij} \log \left(\bar{Q}_i \frac{Q_j / f(d_{ij})}{\sum_{k \neq i} Q_k / f(d_{ik})} \right) \right) \tag{6.24}$$

In our case, we compute instance by instance the value A_{ij} of users whose trajectory starts in i and ends in j, and directly use it for training the benchmark, setting $T_{ij} = A_{ij}$, $Q_i = \sum_{j \in N} A_{ij}$, $\bar{Q}_i = Q_i / \sum_{i \in N, j \in N} A_{ij}$ and computing σ as in (6.24). Then, assuming a number U is given as input to the prediction model, corresponding to the overall number of users into the system, we consider a predicted (average) number of users on the trajectory from i to j equal to $F_{ij} = U \cdot P(i \to j)$.

6.4.3 Key Performance Measures

We remark that a first performance index is the value of the objective function in our optimization models, that is the residuals in the fitting of demands in our solutions. These values, as reported in the following, are always very small.

However, these small values could in principle be also the result of overfitting. Therefore, to provide a fair estimate on quality of our predicted mobility, we designed three further classes of performance indices.

- As encoded in constraints (6.3) of our model, the first index involves the comparison between the distribution of the distances of the original trajectories and the predicted ones. As a numerical measure of goodness of the fitting we use the average absolute residual defined as $\bar{e} = \sum_{k \in K}(|\tilde{Y}_k - Y_k|)/|K|$, given $\tilde{Y}_k$ the cumulative number of users predicted for class $k \in K$ and Y_k the actual number of users for the same class.
- After testing that users move as they are expected, we test if they move *where* they are expected. Given that our simulation takes into consideration the rush-hour time horizon, we assume for each trajectory the first and last visited location to be the two most important locations for the user, being the intermediate locations transitional ones. Still denoting as A_{ij} the original mobility matrix entries, that is the number of users whose trajectory starts in i and ends in j, and the predicted mobility matrix entries as F_{ij}, we consider a matching between original and predicted origins and destinations with increasing flexibility, to which we will refer to as *Prediction Accuracy Matching* in the remainder, that requires combinatorial algorithms to be computed and that we fully detail in Appendix 6.B. The matching is producing as output the measure PME^{δ}: we state that a predicted origin (resp. destination) point α matches with an original origin (resp. destination) point β if α lies in a circle centered in β with radius δ. A predicted path matches an original path if both their origins and destinations match, and PME^{δ} is the percentage of original paths which are successfully matched.
- Finally, we perform a fine grained check on the user mobility paths. In particular, we denote as A^t_{ij} (resp. F^t_{ij}) the number of users moving from AP i to AP j at time t in the original (resp. predicted) paths. We compute:
  1. the numerical difference for every cell values of the original and predicted OD matrices, as: (i) the absolute sum of residual ASR,

$$\frac{\sum_{t \in T} \sum_{(i,j) \in N \times N} |F^t_{i,j} - A^t_{i,j}|/U}{|T|},$$

     where U is the total number of users in the system, and (ii) the symmetric mean absolute percentage error $SMAPE$,

$$\frac{\sum_{t \in T} \frac{1}{|N|^2} \sum_{(i,j) \in N \times N} \frac{|F^t_{i,j} - A^t_{i,j}|}{A^t_{i,j} + F^t_{i,j}}}{|T|};$$

     both averaged over time-frames $t \in T$.

2. disregarding the number of users migrating between locations, we consider if any amount of mobility is observed between each pair of locations i and j at time t. In particular, we consider to be successfully predicted all those pairs for which $(F^t_{i,j} > 0 \wedge A^t_{i,j} > 0) \vee (F^t_{i,j} = 0 \wedge A^t_{i,j} = 0)$, and unsuccessfully predicted all the remaining pairs. Accordingly we can compute the binary classification metrics of accuracy, precision and recall.[1]

In Table 6.3 a notation table for the performance measures is presented.

6.5 Experimental Results

We implemented our algorithms in C++, using IBM ILOG CPLEX 12.6 [72] to solve LP problems. Our experiments ran on an Intel i7 4GHz workstation equipped with 32 GB of RAM. We created a synthetic set of 300 APs with coordinates randomly drawn from two independent normal distributions with mean $\mu = 0$ and standard deviation $\sigma = 1.2$. We considered a time horizon split in 15 time frames. Given this fixed set of APs and time frames, we created five instances for each of the generative models described in Section 6.4, setting the number of users $U = 40000$ in every case. Such a setting is chosen for two main reasons (i) it reflects values in real instances (ii) mobility matrices are sparse, and therefore more challenging to predict. Given the lengths of all original paths in each instance, we computed a set K of 100 path length classes, with l_k values given by the percentiles of lengths distribution and accordingly $n_k = U/100 = 400$. The d^t_i values are computed from original paths by simple counting operations. These data are fully synthetic, allowing us to make them openly available if required, and hence allow reproducibility.

In the following, we first compare the f-UPPP and d-UPPP variants (Subsection 6.5.1). Then, since the numerical resolution of our models requires non-trivial algorithms to be run, we evaluate the computational viability of our methods (Subsection 6.5.2). Subsequently, we focus on their prediction accuracy (Subsection 6.5.3), and we compare them with a benchmark method (Subsection 6.5.4). We complete our assessment by considering the effect of noise in data (Subsection 6.5.5) and longer planning time-horizon (Subsection 6.5.6). We conclude with experiments on a real world dataset (Subsection 6.5.7).

[1]Within the classification results let tp be the number of true positives, fp be the number of false positives and fn be the number of false negative, and let: $precision = \frac{tp}{tp+fp}$, $recall = \frac{tp}{tp+fn}$, and $accuracy = \frac{tp+tn}{|N \times N|}$

Symbol	Meaning
A_{ij}	*generated* OD matrix
F_{ij}	*predicted* OD matrix
B_{ij}	OD matrix generated by benchmark model
Y_k	cumulative number of users generated for class $k \in K$
$\tilde{Y}_k$	cumulative number of users predicted for class $k \in K$
$\bar{e} = \sum_{k \in K}(\lvert\tilde{Y}_k - Y_k\rvert)/\lvert K\rvert$	average absolute residual of fitting
$ASR = \sum_{(i,j) \in N \times N} \lvert F_{i,j} - A_{i,j}\rvert$	absolute sum of residual
$SMAPE = \frac{1}{n} \sum_{(i,j) \in N \times N} \frac{\lvert F_{i,j} - A_{i,j}\rvert}{A_{i,j} + F_{i,j}}$	symmetric mean absolute percentage error
Prediction Accuracy Matching Model	
$S \subseteq N \times N$	set of pairs location with predicted mobility $F_{ij} > 0$
$R \subseteq N \times N$	set of pairs location with original mobility $A_{ij} > 0$
$G = (V, E)$	bipartite graph
δ	distance threshold defining a neighbourhood of a location
$V = S \cup R$	vertices of graph G
$E \subseteq S \times R$	edges of graph G $E = \{(i, j; h, k) \in (S \times R) \lvert$ $(d_{i,h} < \delta \wedge d_{j,k} < \delta) \vee (d_{i,k} < \delta \wedge d_{j,h} < \delta)\}$
$x_{i,j}^{h,k} \in \mathbb{R}^+$	variable flow of users on edges $\forall (i, j; h, k) \in E$
$z \in \mathbb{R}^+$	total amount of mobility positively predicted
$PME^{\delta} = z / \sum_{(i,j) \in R} A_{i,j}$	prediction matching effectiveness measure with distance threshold δ

Table 6.3: UTPP - Performance Indexes Notation Table

	1^{st} stage				2^{nd} stage				
Variant	CG iter	master time	pricer time	n. paths	CG iter	master time	pricer time	n. paths	total time
f-UTPP	1587.8	3.38	0.07	2.58	210.8	0.52	0.47	21.81	6130.40
d-UTPP	81	0.66	1.27	59.08	32.8	8.61	2.35	68.12	514.40

Table 6.4: f-UTPP vs. d-UTPP - Computational Efficiency

6.5.1 Comparing Modeling Variants

We first compare the performance of the two variants of the linking constraints of user trajectories and network loads variations, namely f-UTPP and d-UTPP, limiting ourselves to the Ad-Hoc generative model.

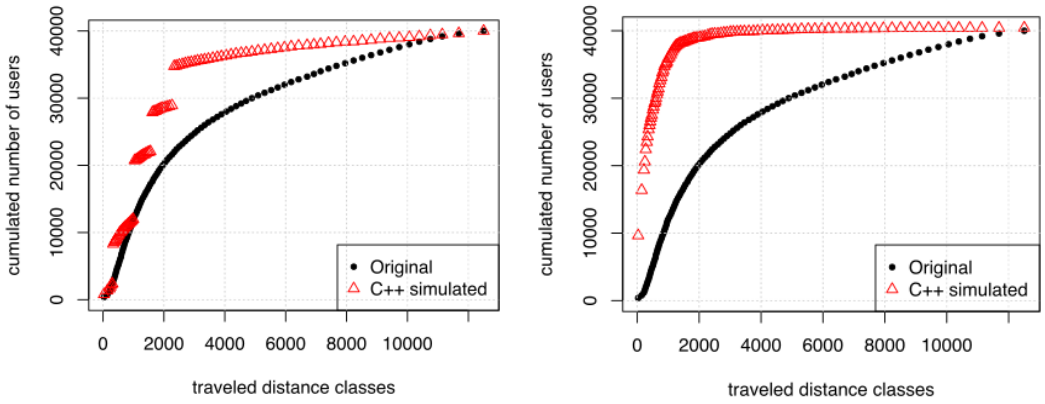
On the computational viability point of view in Table 6.4 we report for each stage of our algorithm (column blocks) and for the two linking constraints variants (table rows): (i) the average number of column generation iterations; (ii) the average execution time of each master LP optimization (in sec.); (iii) the average execution time of each dynamic programming pricing algorithm (in sec.); (iv) the average. number of paths added at each column generation iteration; and (v) the total execution time (in sec.). Values are averaged over the 5 instances.

The f-UPPP model requires a higher running time for its execution; it also requires a higher number of column generation iteration, in each of which it generates a lower number of new (negative reduced cost) variables.

On the prediction accuracy point of view, we first report a visual comparison of the CDF of distances for the original and predicted trajectories after both stages of our algorithm (Figure 6.4).

As a quantitative comparison measure, instead, we report the average residual fitting $\bar{e}$, averaged for the five data instances, for both stages of our algorithm for the two model variants (Figure 6.8). In the first stage of the algorithm f-UPPP provides higher residuals than d-UPPP, while after the second stage the residual is similar for both models. In Figure 6.7 a single representative data instance is depicted, in terms of fitting after the first stage of the algorithm for both model variants: f-UPPP associates to short paths almost the total amount of users in the system (Figure 6.7b), while d-UPPP shows a step-wise fitting (Figure 6.7a).

In Figure 6.9 we present the PME^{δ} index averaged on all data instances (y



(a) d-UTPP (b) f-UTPP

Figure 6.7: Fitting of Traveled Distance Probability 1^{st} stage f-UTPP vs d-UTPP

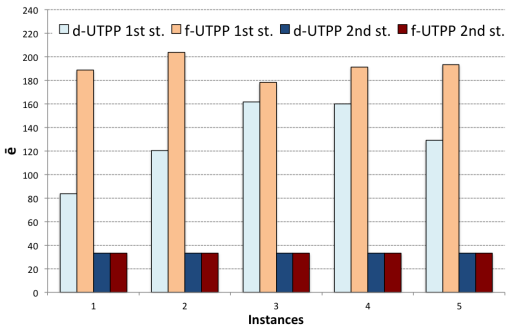


Figure 6.8: $\bar{e}$ - f-UTPP vs. d-UTPP

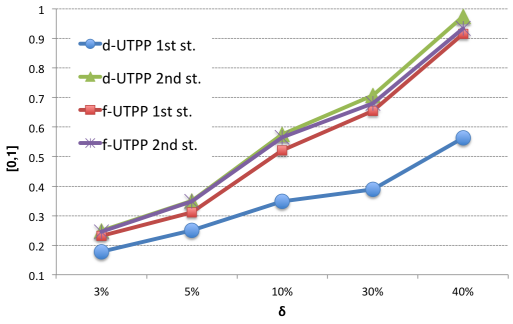


Figure 6.9: PME^{δ} - f-UTPP vs. d-UTPP

axis), for both stages of the algorithm for the two model variants, with 5 different values for δ (x axis). The PME^{δ} index of prediction accuracy is also detailed for all data instances in Table 6.5. After the first stage, f-UPPP outperforms d-UPPP. On the contrary, after the second stage d-UPPP performs slightly better.

As a further comparison we consider the OD matrices produced by f-UPPP and d-UPPP. In Figure 6.10 we report ASR, SMAPE, accuracy, precision and recall averaged on the 5 data instances for both stages of our algorithm and for the two model variants. We can notice that for the ASR and SMAPE both variants have similar outcome (d-UPPP performs only marginally better), but for precision and recall indexes after both algorithm stages d-UPPP clearly outperforms f-UPPP. We

		1^{st} stage δ					2^{nd} stage δ				
Inst.	Variant	3%	5%	10%	20%	40%	3%	5%	10%	20%	40%
1	f-UTPP	0.17	0.23	0.32	0.35	0.40	0.25	0.35	0.59	0.72	0.98
	d-UTPP	0.24	0.33	0.53	0.66	0.92	0.24	0.35	0.56	0.67	0.93
2	f-UTPP	0.18	0.26	0.36	0.40	0.52	0.25	0.34	0.58	0.72	0.98
	d-UTPP	0.21	0.28	0.49	0.63	0.89	0.25	0.35	0.57	0.69	0.96
3	f-UTPP	0.18	0.25	0.35	0.40	0.80	0.25	0.36	0.57	0.69	0.97
	d-UTPP	0.25	0.34	0.55	0.67	0.93	0.25	0.35	0.56	0.69	0.94
4	f-UTPP	0.18	0.25	0.35	0.39	0.58	0.25	0.36	0.60	0.74	0.98
	d-UTPP	0.22	0.30	0.52	0.65	0.92	0.24	0.35	0.56	0.67	0.92
5	f-UTPP	0.18	0.26	0.36	0.41	0.52	0.25	0.35	0.55	0.67	0.97
	d-UTPP	0.23	0.31	0.52	0.65	0.91	0.25	0.35	0.57	0.68	0.92

Table 6.5: f-UTPP vs. d-UTPP - Prediction Accuracy PME^{δ}

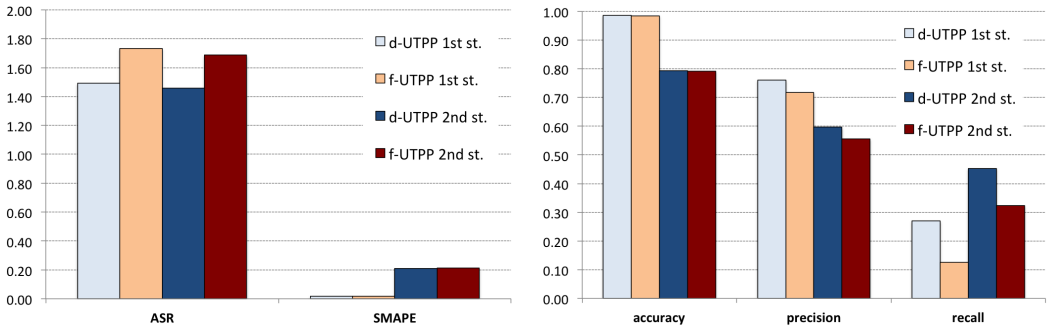
can also notice that the second stage of our algorithm worsen the SMAPE measure while does not change the ASR. However this worsening is combined to a substantial improvement of the recall, this latter being more important in the evaluation of the overall quality of the solution.

Given the outcome of this experiments we decided to focus only on the d-UTPP model. Hence, in the remainder, every experiment is intended to be executed with it.

6.5.2 Computational Viability

In a second round of experiments we focus on the computational viability of our method, using d-UPPP, testing it on all generative models. Table 6.6 reports the column generation performances for both stages of our algorithm (column blocks) and for each data instance (table rows), for each generative model. The structure of the Table is the same as that of Table 6.4. Our methods show to be computationally stable, the most critical point being the master LP optimization during second stage optimization.

Even with the same number of APs, users and time frames, different generative models require different execution times, while performance is similar for all instances



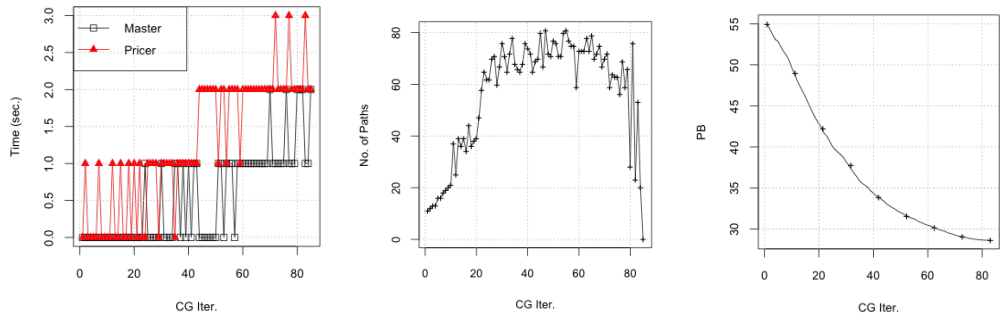
(a) ASR, SMAPE (b) Accuracy, Precision, Recall

Figure 6.10: Benchmark Comparison Flow Conservation Variants

of the same generative model. In particular the POI model requires the highest computing effort (Table 6.6b): it amounts to almost the double of the Ad-Hoc model (Table 6.6a). The Ad-Hoc model requires twice the column generation iterations for the first stage and twice the time for the average master execution. On the other hand, PREGM (Table 6.6c) requires fewer CG iterations for the first stage of the model, but it requires a higher number of CG iterations for the second stage than the Ad-Hoc model, which leads to higher total execution time.

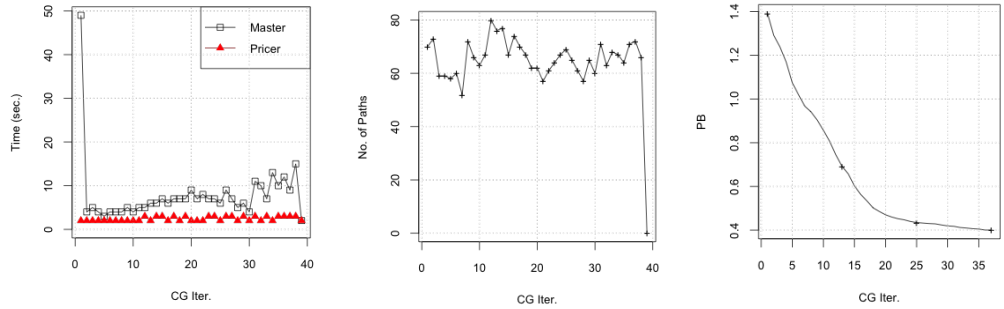
As CG iterations add paths to the model we can assume that, not surprisingly, a higher number of CG iterations in the first stage leads to fewer number of CG in the second stage of the algorithm, as happens in POI model, given that potentially useful paths for the secondary objective have already been added to the model in the first stage. The higher number of paths added to the model in the first stage also leads to a higher time required for the first master execution of the second stage, as we can see for the POI model in Table 6.6b, given that the model has to deal with a higher number of variables. On the contrary fewer CG iterations in the first stage leads to a higher CG iterations in the second stage, as for the PREGM.

Figures 6.11 and 6.12 report, respectively for the first and second stage of the model and for a single data instance of the Ad-Hoc generation model, several details of column generation iterations: the execution time, the number of path generated and the primal bound value of the master model. We notice (Figure 6.12a) that the first execution of the master of the second stage of the algorithm requires the majority of time of the overall execution, and afterward the execution time falls down to increase again as CG iterations are executed and new paths are added to the model. Execution of the pricer requires low times even as CG iterations go



(a) Exc. Time (b) N. Paths (c) PB

Figure 6.11: 1^{st} stage Computational Efficiency Ad-Hoc Single Instance



(a) Exc. Time (b) N. Paths (c) PB

Figure 6.12: 2^{nd} stage Computational Efficiency Ad-Hoc Single Instance

forward. On the contrary, for the first stage the time required for both master and pricer increases as CG iterations increase, with both times being rather low.

Our multiple pricing strategy proves to be effective: the number of generated paths remains almost constant through all CG iterations and consists in almost half of the maximum number of paths that can be possibly generated in every iteration, that is the number of traveling distance classes used in the data.

6.5.3 Prediction Accuracy

In a third round of experiments we focus on the prediction accuracy of our methods. In Figure 6.13 we present, for each generative model and for both stages of our algorithm, the PME^{δ} averaged for all data instances. Table 6.7 reports the same PME^{δ}

	1^{st} stage				2^{nd} stage				
Inst.	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
1	78	0.68	1.24	59.88	32	9.44	2.28	70.28	525
2	80	0.71	1.43	62.49	29	9.66	2.41	69.24	521
3	84	0.56	1.29	57.50	38	8.05	2.47	65.84	555
4	80	0.74	1.24	59.26	31	8.26	2.35	69.32	487
5	83	0.61	1.18	56.24	34	7.65	2.21	65.91	484

(a) Ad-Hoc

	1^{st} stage				2^{nd} stage				
Inst.	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
1	126	1.83	1.78	56.02	15	25.60	2.80	58.93	880
2	124	1.81	1.66	54.40	10	23.60	3.40	59.90	701
3	108	1.89	1.70	62.71	28	21.25	2.39	62.25	1050
4	132	1.88	1.81	53.17	8	36.63	3	58.88	804
5	117	2.38	1.82	63.15	38	27.61	2.50	62.32	1636

(b) POI

	1^{st} stage				2^{nd} stage				
Inst.	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
1	64	0.64	1.34	65.56	60	9.67	2.55	68.12	860
2	61	0.59	1.36	64.31	48	9.31	2.06	69.85	665
3	63	0.75	1.24	64.97	60	9.50	2.43	67.08	841
4	64	0.67	1.25	61.42	57	8.91	2.42	66.89	769
5	65	0.55	1.23	58.63	57	8.58	2.28	67.79	735

(c) PREGM

Table 6.6: d-UTPP - Computational Efficiency

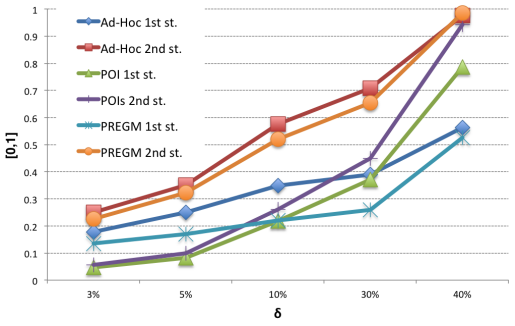


Figure 6.13: d-UTPP PME^{δ} averaged

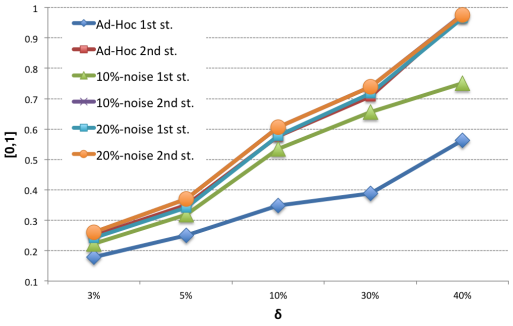


Figure 6.14: d-UTPP PME^{δ} averaged with noise

measures expanded for every data instance. We found low variability of measures among instances of the same generative model. We can notice that results on Ad-Hoc and PREGM instance are similar, with a great improvement after the second stage of the algorithm. On the contrary, POI trajectories are harder to predict, with few improvements given by the second stage of the algorithm: good accuracies require a δ threshold greater than 30%. After the first stage of the algorithm, performance is poor for Ad-Hoc and PREGM even with high threshold δ: we assume this effect to be due to the low number of paths generated by the first stage of the algorithm. Therefore, we keep only second stage results in the subsequent experiments.

6.5.4 Benchmark comparison

As fourth experiment we compare the prediction accuracy of our methods with that of the benchmark presented in Section 6.4.2. For the sake of completeness, we stress that a slight advantage is given to the benchmark, as the original matrices A^t_{ij} are used for testing, and their aggregation A_{ij} is used also for training; this is not the case for our method.

In Table 6.8 we report for each generative models (sub-tables) and for both our algorithm and the benchmark(row-blocks), the performance measures averaged for all instances (column-blocks). No comparison on PME^{δ} values is possible, as the benchmark is unable to predict individual user paths. We notice that:

- our algorithm is always better than the benchmark in terms of ASR and SMAPE, and in particular the SMAPE index is considerably higher;
- the benchmark model provides very poor results in terms of accuracy and

	1^{st} stage					2^{nd} stage				
Inst.	3%	5%	10%	20%	40%	3%	5%	10%	20%	40%
1	0.169	0.230	0.320	0.351	0.402	0.249	0.348	0.586	0.722	0.977
2	0.184	0.260	0.364	0.396	0.520	0.246	0.345	0.577	0.715	0.978
3	0.182	0.254	0.351	0.402	0.800	0.253	0.357	0.570	0.688	0.972
4	0.176	0.251	0.349	0.387	0.575	0.248	0.356	0.599	0.740	0.977
5	0.179	0.255	0.357	0.407	0.519	0.249	0.346	0.546	0.671	0.974

(a) Ad-Hoc

	1^{st} stage					2^{nd} stage				
Inst.	3%	5%	10%	20%	40%	3%	5%	10%	20%	40%
1	0.064	0.112	0.263	0.427	0.874	0.068	0.114	0.277	0.461	0.950
2	0.053	0.092	0.212	0.372	0.871	0.046	0.087	0.217	0.382	0.867
3	0.037	0.060	0.209	0.432	0.867	0.047	0.075	0.244	0.486	0.982
4	0.047	0.080	0.216	0.342	0.805	0.057	0.093	0.243	0.395	0.932
5	0.032	0.071	0.188	0.281	0.515	0.063	0.126	0.327	0.522	0.979

(b) POI

	1^{st} stage					2^{nd} stage				
Inst.	3%	5%	10%	20%	40%	3%	5%	10%	20%	40%
1	0.140	0.180	0.227	0.263	0.536	0.243	0.350	0.567	0.707	0.990
2	0.143	0.179	0.244	0.309	0.799	0.211	0.294	0.457	0.568	0.966
3	0.132	0.166	0.216	0.256	0.557	0.227	0.328	0.542	0.682	0.990
4	0.130	0.163	0.221	0.263	0.498	0.216	0.308	0.489	0.616	0.990
5	0.132	0.160	0.191	0.202	0.239	0.231	0.334	0.548	0.692	0.990

(c) PREGM

Table 6.7: d-UTPP - PME^{δ}

Algo.	ASR	SMAPE	accuracy	precision	recall
1^{st} stage	1.4904	0.0158	0.9864	0.7613	0.2696
2^{nd} stage	1.4595	0.2103	0.7938	0.5966	0.4530
bench.	1.5928	0.9907	0.0199	0.0166	1

(a) Ad-Hoc

Algo.	ASR	SMAPE	accuracy	precision	recall
1^{st} stage	1.6548	0.0114	0.9901	0.3080	0.4081
2^{nd} stage	1.6835	0.2084	0.7938	0.2540	0.5379
bench.	1.8560	0.9953	0.0105	0.0072	1

(b) POI

Algo.	ASR	SMAPE	accuracy	precision	recall
1^{st} stage	1.5013	0.0172	0.9849	0.7920	0.2502
2^{nd} stage	1.4182	0.2113	0.7933	0.6315	0.4740
bench.	1.5902	0.9904	0.0218	0.0185	1

(c) PREGM

Table 6.8: d-UTPP - Benchmark performance indexes

precision, while our algorithm provides balanced performances for the three indexes. We further observe that the benchmark recall is (artificially) high: it predicts mobility for almost all possible pairs of location, but the majority of those pairs is not related to actual mobility;

- little differences can be noticed among the generative models' performances: the greatest difference is given by the POI generative model which provides the worst results in terms of precision.

6.5.5 Demand matrix perturbation

In a more realistic scenario, trajectory predictions are not made on actual data, but rather on a forecasting of it, that might therefore be affected by errors. To test the performances of our algorithm in such a case, we designed the following experiment. We consider the instances generated by the Ad-Hoc model to represent

original data; the original paths yield trajectories and demand values d_i^t. We perform a perturbation by adding a random noise value to each d_i^t, considering the perturbed values as a possible outcome of a previous *forecasting* of the demand. Then, we use this representation of *forecast* values as input of our models, we optimize it and we retrieve predicted paths and corresponding trajectories. Finally, we compare these solutions with the *original* paths and trajectories.

In details, we analyze two cases, drawing a percentage noise uniformly at random in the ranges $[0.8, 1.6]$ (resp. $[0.9, 1.4]$), to which we will refer as 20%-*noise* (resp. 10%-*noise*) in the remainder.

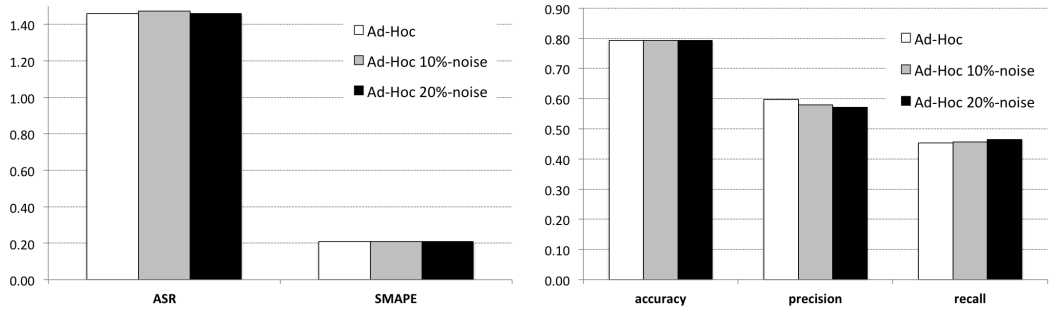
The average computational results of 20%-*noise* and 10%-*noise* cases are presented in Table 6.9, whose format matches that of Table 6.6. By comparing with values in Table 6.6a, we notice that the computing behaviour remains substantially unchanged. The first stage of the algorithm requires more CG iterations, while the second stage requires less CG iterations. The time required for master and pricer executions in the first stage are similar to those of the original dataset, while master's execution times of the second stage are higher for the perturbed data, given that the first stage generates more paths. We omit instance-by-instance results, since they add no further insight.

The most interesting results are, indeed, on the prediction accuracy point of view. In Figure 6.14 we report the average PME^δ (y axis) after the second stage of our algorithms, for different values of δ (x axis), when different degree of noise affects data (no noise, 10% or 20%). In Figure 6.15 we report histograms with the corresponding performance measures related to the OD matrices. The PME^δ and accuracy measures remain unaffected by noise. In Figure 6.15a we notice that ASR and SMAPE are only slightly higher even for high noise values; in Figure 6.15b we also notice that precision slighly decreases as noise increases, while recall has an opposite behaviour.

The is, our models prove to be very *robust* to noise in demand values. We conjecture that such a positive behaviour is given by the inner structure of our models, whose main target is to rebuild user paths: random noise changes demands, but leave path structures (and therefore also predicted trajectories) unchanged, since we are able to "collect" such noise in the ϵ_i^t terms during the optimization process.

	1^{st} stage				2^{nd} stage				
Variant	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
10%-noise	87.6	0.80	1.39	61.08	19.4	12.43	2.68	73.24	484.40
20%-noise	91.2	0.97	1.59	64.68	11.8	17.87	2.88	72.62	476.80

Table 6.9: d-UTPP - Computational Efficiency Perturbed Data



(a) ASR, SMAPE (b) Accuracy, Precision, Recall

Figure 6.15: d-UTPP - Benchmark Comparison Perturbed Data

6.5.6 Increasing Time-Frames

As further experiments we increase the number of time frames on which we generate and predict user mobility. In order to maintain computationally manageable instances, we used a smaller network of 50 APs with coordinates randomly drawn from a pair of normal distributions with mean $\mu = 0$ and standard deviation $\sigma = 1.2$ and then multiplied by one-thousand. We consider 40 and 80 times frames with PREGM, that was originally meant to deal with commuters trajectory on a long time scale rather than on a rush hour scale. Moreover during the trajectory generation also waiting times are randomly generated so that each hops in the trajectory does not happens in successive time frames.

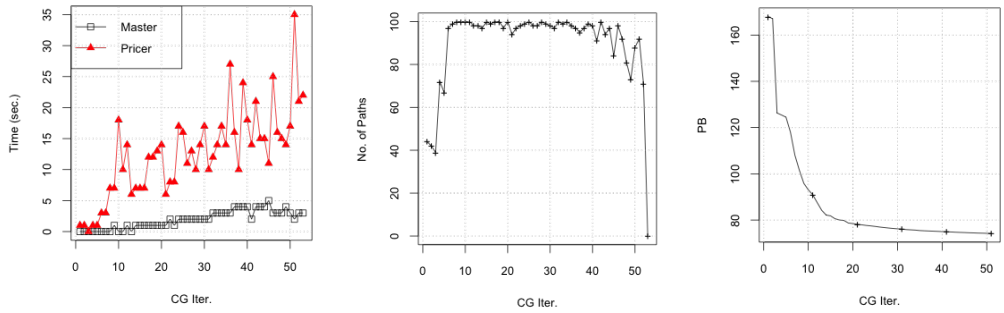
In Table 6.10 we report the computational results of these variants, in the same format as in Table 6.6. We can see how the number of time frames affects the required execution time. While in the previous experiments, using 300 nodes and 15 time-frames, ~ 750 seconds were require on average (Table 6.6c), using 50 nodes and 40 time frames requires ~ 110 seconds, while with double the time-frames requires ~ 50 times more execution times, i.e. ~ 5200 seconds. With the highest number of time frames a higher number of CG iteration are required, and in the first stage of the algorithm the time required to execute the pricer grows sensibly and is higher than the time required by the master execution, while in the second stage both master and pricers execution times are high.

Going into detail, in Figure 6.16 we report for a single instance with 80 time-frames and for each CG iteration: the execution time required for the master and the pricer, the number of negative reduced cost variables created and the primal bound. In Figure 6.17 the same information are presented for the second stage of the algorithm. We can notice in Figure 6.16a that in the first stage of the algorithm as path variables are added in the model the time required by the pricer increase faster than the time required by the master. On the contrary in Figure 6.17a we can notice that as paths variables are added in the model in the second stage of the algorithm, the time required for the master resolution increases, while the pricer execution time does not show an increasing trend.

In Figure 6.18 the performance measures on the OD matrices are presented, averaged on all instances: we can notice that we have worse results for the ASR and SMAPE indexes compared to the results presented in Table 6.8c, primarily in the SMAPE value. Moreover we can notice that, while the algorithm provides a precision of almost 100%, the accuracy and recall are poor. We observed that the mobility is estimated as grouped in few pairs of locations associated to real mobility, but

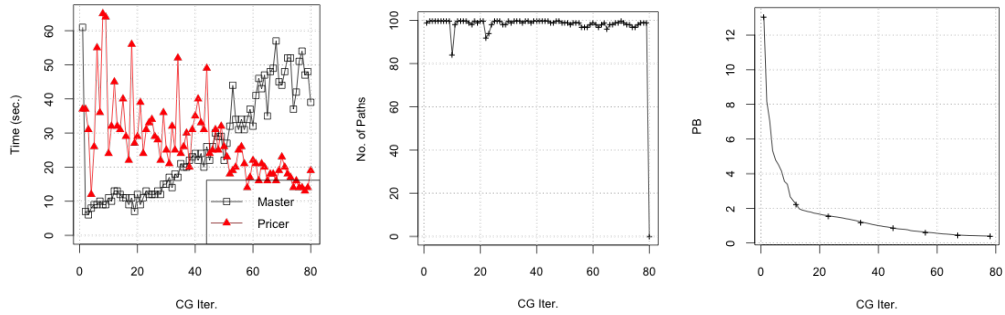
	1^{st} stage				2^{nd} stage				
Variant	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
40 times	34.6	0.21	0.83	62.29	24	1.35	1.74	79.10	109.80
80 times	50.4	1.89	14.30	93.64	82.4	25.91	27.63	99.02	5226.60

Table 6.10: d-UTPP - Computational Efficiency Long Time Horizon



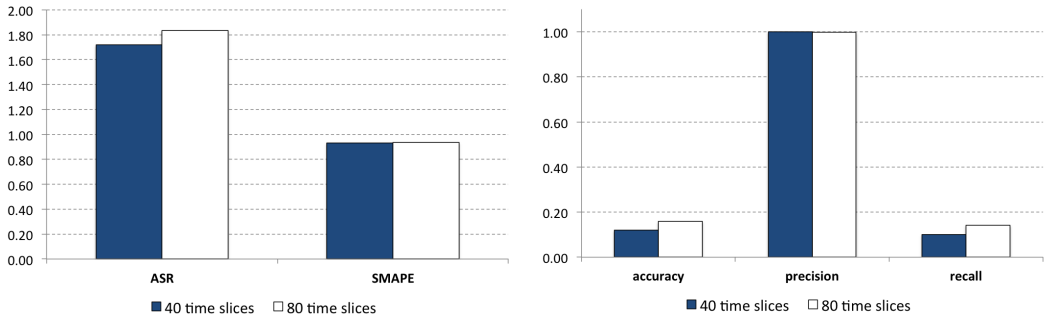
(a) Exc. Time (b) N. Paths (c) PB

Figure 6.16: 1^{st} stage Computational Efficiency PREGM Single Instance



(a) Exc. Time (b) N. Paths (c) PB

Figure 6.17: 2^{nd} stage Computational Efficiency PREGM Single Instance



(a) ASR, SMAPE

(b) Accuracy, Precision, Recall

Figure 6.18: d-UTPP - Performance Indexes Long Time Horizon

the majority of actual mobility pairs are not predicted, so that there are few false positive prediction, giving high precision, but high false negative, leading to low recall and accuracy. In Table 6.11 detailed performance indexes for every instance are presented.

6.5.7 Real-World Dataset

At last we perform experiments on real world dataset presented in Chapter 4, consisting in actual mobile data traffic (without the corresponding individual user trajectories); the user mobility was computed from aggregated data directly in form of OD matrix; namely, each entry (i, j) of the OD matrix contains the number of users having APs i and j as the two most frequently visited APs during the day. From this OD matrix we extract 100 classes of traveled distance, and therefore the l_k and n_k values as the percentiles of the corresponding distribution. That is, we obtain all input data for our algorithms, but no ground truth for checking the prediction accuracy of our solutions.

In Figure 6.19 we present the histogram of the strictly positive traveled distances of all users, according to our OD matrix. Almost 30% of the users does not move at all during the considered day, while a hybrid trend is observed for the remaining ones: the number of users with short (positive) traveled distance decade with a trend similar to a power law, but after a certain threshold the distribution increases mildly, but smoothly and constantly, until the maximum mobility probed.

For what concerns time discretization, we perform experiments with two set-

	Inst.	ASR	SMAPE	accuracy	precision	recall
40 time frames	1	1.722	0.933	0.118	1	0.100
	2	1.733	0.930	0.121	1	0.102
	3	1.740	0.932	0.119	1	0.098
	4	1.697	0.931	0.119	1	0.100
	5	1.709	0.930	0.119	1	0.100
80 time frames	1	1.820	0.933	0.155	0.997	0.138
	2	1.838	0.936	0.161	1	0.144
	3	1.849	0.934	0.159	0.997	0.141
	4	1.831	0.934	0.161	1	0.144
	5	1.837	0.934	0.158	1	0.140

Table 6.11: Long time horizon performance indexes

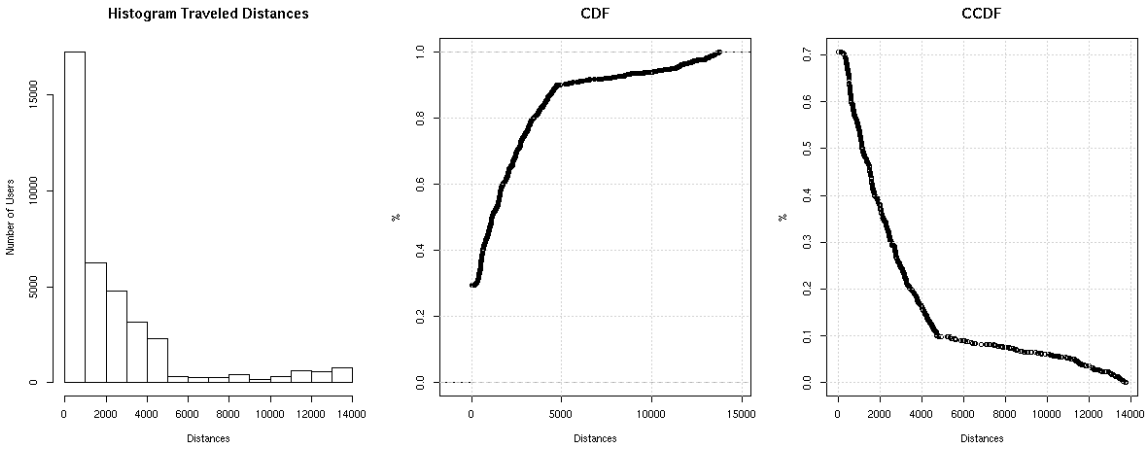


Figure 6.19: Traveled distances statistics

	1^{st} stage				2^{nd} stage				
Variant	CG iter	master t.	pricer t.	n. paths	CG iter	master t.	pricer t.	n. paths	total t.
10 time frames	148	0.50	0.53	36.47	18	8.39	1.33	51.44	327
20 time frames	245	3.63	7.55	60.30	49	129.80	13.18	69.96	9745

Table 6.12: d-UTPP - Computational Efficiency Real World Dataset

tings: (i) with 10 time frames, averaging traffic demand in time ranges of 24 minutes; (ii) with 20 time frames, averaging traffic demand in time ranges of 12 minutes.

We report in Table 6.12 the computational efficiency measures, in the same format of Table 6.6: even with ~ 600 APs in the network, the execution time needed with 10 time frames is reasonably low. With 20 time frames, instead, the execution time grows very fast.

Since we have no access to original individual paths, we limit our prediction accuracy tests on the comparison of the mobility matrices. In Figure 6.20 we present average PME^{δ} measures with interesting insights:

- the outcomes are similar to those on instances generated through the Ad-Hoc models, presented in Figure 6.13, and even slightly better: with a low threshold δ of 5% PME^{δ} is still higher than 50%; that is, more than half of the original OD matrix is correctly predicted;
- the prediction accuracy improves slightly using a higher number of time frames.

6.6 Conclusions

As reported in the introduction, our focus is in retrieving (i.e. predicting) the mobility of users in a mobile network of urban-size area during rush hours. The majority of human mobility prediction approaches presented in literature requires a considerable amount of information, such as complete census data of the considered area, origin-destination matrices, traces of individual movements and social media check-in, among others. Instead, we exploit very aggregated data: only the mobile AP demands in a given time horizon and generic statistical properties of the distribution

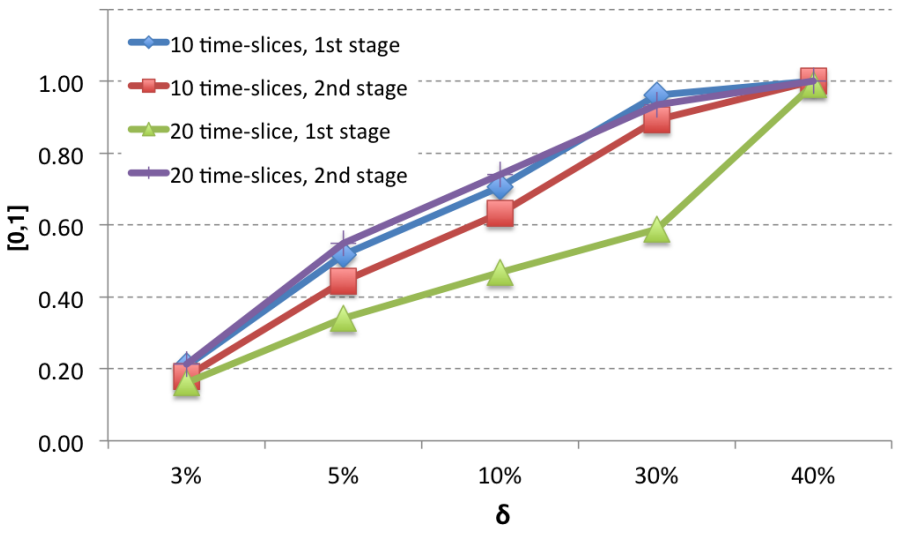


Figure 6.20: d-UTPP - Prediction Accuracy Real World Dataset

of mobility in the considered area.

Standard predictive models are unsuitable for such a task: in a comparison experiment a benchmark model from the literature yields, for instance, accuracy and precision values as low as 2%.

We propose instead dedicated mathematical programming models. To test their performances we introduce suitable performance measures.

We perform an extensive experimental campaign, adapting three generative models from the literature (ad-hoc, PREGM and POI), and considering aggregated data from a real world case study.

Since our models require us to design ad-hoc resolution algorithms, which exploit dynamic variable generation techniques, our first experiments focuses on the computational issues. Our algorithms prove to be efficient enough to tackle both synthetic and real world instances. As a main insight, we report that computational issues arise mainly as the time discretization granularity gets finer.

For what concerns prediction accuracy, POI instances are the hardest to predict: good PME^{δ} scores require a δ threshold greater than 30%; for ad-hoc and PREGM instances, instead, a δ threshold greater than 10% is enough to match the mobility of about 60% of the users.

Our methods prove also to be robust: we find low variability of measures among

instances of the same generative model; furthermore, an experiment artificially introducing noise in data shows that PME^{δ} scores remain unaffected even if demand values are perturbed by up to 20% on average.

While no ground truth is available for real world data, we find our experiments to match well those on ad-hoc generative datasets, yielding us to believe that the results provided by our models might be accurate also in this case.

Appendix

6.A Literature Review for Human Mobility Estimation

In Chapter 6 we present a new approach to estimate user mobility given very aggregated data. In particular we suppose that only mobile access point demands in a time horizon are known, together with the knowledge of some general statistical properties of the mobility patterns.

The modeling of human mobility is a well studied research field, that the availability of spatiotemporal information of mobile phone users has push forward with a great number of research projects and potential applications. In this Chapter we propose examples of the variety of human mobility prediction models available in the literature, that is far to be an exhaustive survey of all models. A further survey for mobility prediction models derived from mobile phone datasets analysis has been presented in [23]. With respect to all these works, in our approach for user mobility estimation presented we require the knowledge of very low amount of information.

As a direct implementation of the intervening opportunities theory, authors in [101] propose a *rank-distance* model: a rank value is computed for each possible migration between pair of locations, equal to the number of opportunities nearer to the origin than the destination. The probability that a migration take place is inversely proportional to the associated rank value. Opportunities was retrieved by a social media check-in data inside a city, used also to train and evaluate the predictive model.

Authors in [102] make use of the density of population in different areas throughout a region as a proxy for intervening opportunities, presenting the *radiation* model. This states that the average number of commuters between the two locations depends on the population of both locations and on the total population of the circular area centered in the origin location and with radius equal to the distance between the two locations. The rationale is that a dense area is full of opportunities for a person to satisfy her need, and hence she is willing to move to a distant place only if the nearer area has less opportunities, i.e. if it has low population density. Authors test the model with long distance commuters data taken from census, claiming that their radiation model out-perform gravity-models in this scope. However authors in [105] question the universal accuracy of the radiation model as a prediction model, experimenting that the gravity model perform better for transportation patters between cities and intra-city mobility. To tune gravity model's parameters they used long

distance commuters data taken from census and the scheduled timetables of public transport services.

Authors in [106] suppose that the mobility between two locations is proportional to the intensity of communication between these two locations and inversely proportional to the distance separating the two locations. Hence the intensity of communication is used has a proxy for the intervening opportunities, with the rationale that a personal communicate between two locations if she has strong relationship with these locations. In particular authors rely on the gravity-based model, replacing the population density of the pair of locations with the volume of communication between locations. In a similar fashion, authors in [107] make use of Erlang measurement as a proxy for the density of population in an area. The Erlang measurement indicates the aggregate call volume in a given cell during a given period. In particular the use of one mobile phone for one hour in a particular cell constitutes one Erlang, whereas the use of two phones for half an hour each also constitutes one Erlang. Their goal is to use the mobile network activity changes as a way to infer the patterns of changes of population distribution as a whole in time. With respect to these works, in our case we take into consideration a lower amount of data, that is the load of the network in a certain area during time in terms of number of connected UEs, without requiring the direction of communications.

Authors in [108] propose an individual mobility predictor that combines the person's past mobility choices and collective behavior. To model the individual behavior they follow a probabilistic approach, defining a probability for each location to be the next visit of the person as function of the person and collective past behaviors. The location with greatest probability is then chosen as predicted next visit of the person. In order to compute this probability matrix, a considerable history of individual movements is needed to train the model to include the periodic personal behavior. To model the collective behavior, they design the probability to choose a given destination to be a function of: (i) the distance of the destination, (ii) the presence of points of interests similar to the ones the collectivity has visited, and (iii) the type of land use the collectivity has been in. To have a single final probability to migrate between pair of location a combination of individual and collective behavior is performed. In order to get the information needed to train the model, they used a mobile phone CDR dataset containing the exact location of anonymous users, a reviews and recommendations social media dataset to get POI data and and administrative data to get land use information.

In [109] authors propose a probabilistic mobility prediction scheme based on: (i) the trip origin and current location; (ii) current and future directions of the

mobile users; (iii) the history of the trajectories followed by the users; and (iv) the information on the users' contextual knowledge, provided by the user itself, consisting in usual activities, interests, more frequented locations, etc.. The mobility is modeled with a second order Markov chain that is applied for predicting the transition that an arbitrary user makes from its current location within a certain time period.

In [110] authors provide predictions as a probability distribution of the likelihood of moving to a set of future locations. Taking into account the users' mobility history, they build a Discrete Time Markov Chain methods (DTMC), computing for every pairs of locations the probability of transition as the frequency of historical transition over the total number of transition starting from the same origin. A different DTMC is build for different classes of users. The membership of users to a class is computed as a probability through an expectation-maximization algorithm. They test the prediction accuracy using as training mobility history composed by cell-level accuracy and GPS-level accuracy.

In [103] authors starts from gravity model to predict intra-urban mobility, splitting the geographic areas in location and using as input data the number of trips between every pair of locations and the density of population of each locations. The parameters of the gravity model are computed through a maximum likelihood estimation algorithm. This model is used as benchmark in our experiments and will be detailed further in Section 6.4.

In [111] autors study the inferences of size and topology of geographical territory on intra-urban mobility. Authors present an extension of the mobility model presented in [95], embedding the geographical heterogeneity as a probability for each location to act as an attractive point for a user. In our case we model the geographical space with an adjacency matrix which takes into account all cells of the network: hence any topology can be tackled by our model.

In [100] authors combine activity-based analysis with a movement-based approach to model the intra-urban human mobility observed from social media check-in records, building a temporal transition probability matrix to represent travel demands during a time interval. They decouple the transition probability in two components: the time-dependent probability that an activity is fulfilled and the subsequent activity-depended probability that user move to a specific location.

Mobility simulation software already exists, often requiring a high amount of information to characterize the system to simulate. As examples:

- TRANSIMS [112] is an agent-based simulation model of the daily movement of individuals in virtual region or city with a complete representation of: (i)

the population at the level of households and individual travellers; (ii) the daily activities of the individuals, and; (iii) the transportation infrastructure. Demographic characteristics are taken from census data: (i) households are geographically distributed according to the population distribution; (ii) transportation network is a precise representation of the city's transportation infrastructure and includes multiple transport modes; (iii) data on activities include origin destination route timing and forms of transportation used;

- in [113] authors propose an agent-based simulation model based on the combination of individual regular movements with spatial considerations, represented by an expanded gravitation model, embedded in the properties of the agent. Agents are characterized by several socio-economic attributes. Geographical area is split in cells, which are characterized by land use. Day-time is split into periods. Transition probabilities differ for every agent socio-economical class, giving higher probabilities to different land use cells at different time periods.

6.B UTPP - Prediction Accuracy Matching

In Chapter 6 we present a new approach to estimate user mobility given very aggregated data. In Section 6.4.3 we present performance measures to qualify the goodness of the simulated user mobility. Among the measures, we define the *Prediction Accuracy Matching* (PME^{δ}) as a measure that tests if the predicted trajectories estimates the original trajectories in terms of their origins and destinations, rather than the complete set of traversed locations. In order to add some flexibility we can be satisfied if the model can generate trajectories that lies in the proximity of the original origin and destination. In this chapter we fully detail the computation of PME^{δ} measure.

We introduce a model to match the original trajectories to the predicted ones within a range of distance defining a neighborhood. To better clarify our approach, in Figure 6.B.1 we present an example: let O be an *original* path and P and P' two paths generated by our algorithm. Our aim here is not to retrieve exactly the original path, but rather to identify origins and destinations of users: we state that a path generated by our algorithm match the mobility of an original paths if origins and destinations of both paths are in the same area, this latter being identified by a circle with radius δ centered in the origin and destination of the original path. In the example in Figure 6.B.1 path P' matches the mobility of original path O, while path P does not match the mobility of O because its destination node its outside the feasible area of the destination of path O.

Formally, let: $R = \{(i,j) \in (\mathrm{N} \times \mathrm{N}) : A_{i,j} > 0\}$ be the set of pairs of locations with strictly positive original mobility $A_{i,j}$; $S = \{(i,j) \in (\mathrm{N} \times \mathrm{N}) : F_{i,j} > 0\}$ to be the set of pairs of locations with strictly positive predicted mobility $F_{i,j}$; δ to be the radius of the neighborhood of a location; $G = (V, E)$ to be a bipartite graph having vertices $V = S \cup R$ and edges from vertices of predicted mobility S to vertices of original mobility R if their respective origin and destination locations lies in the same neighborhood defined by δ, i.e. $E = \{(i,j;h,k) \in (S \times R) : (d_{i,h} < \delta \wedge d_{j,k} < \delta) \vee (d_{i,k} < \delta \wedge d_{j,h} < \delta)\}$.

On this graph we define an LP problem of maximum flow, where variables $x_{i,j}^{h,k} \geq 0 \ \forall (i,j;h,k) \in E$ represent the flow (or matching) of users between the pair of locations $(i,j) \in S$ and the pair $(h,k) \in R$. Formally:

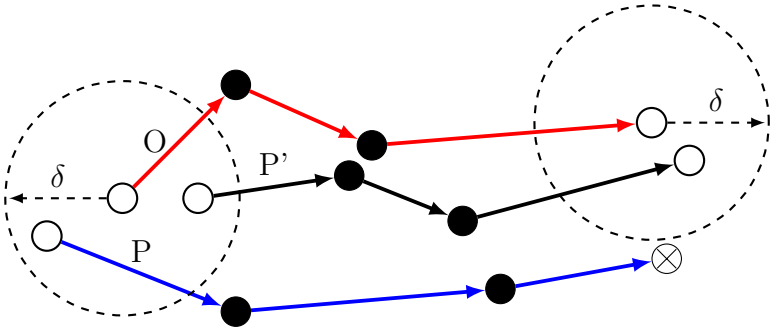


Figure 6.B.1: UTPP - Example of Mobility Matching with Neighborhood

$$\max\ z = \sum_{(i,j;h,k)\in E} x_{i,j}^{h,k} \tag{6.25}$$

$$\text{s.t.} \sum_{\substack{(h,k)\in S \\ |\exists(i,j;h,k)\in E}} x_{i,j}^{h,k} \leq A_{i,j} \qquad \forall (i,j) \in R \tag{6.26}$$

$$\sum_{\substack{(i,j)\in R \\ |\exists(i,j;h,k)\in E}} x_{i,j}^{h,k} \leq F_{h,k} \qquad \forall (h,k) \in S \tag{6.27}$$

$$x_{i,j}^{h,k} \geq 0 \qquad \forall (i,j;h,k) \in E \tag{6.28}$$

Optimal value z in (6.25) represents the number of moving users positively predicted by our algorithm. Constraints (6.26) impose that a flow from original mobility vertices $(i,j) \in R$ cannot exceed the corresponding original mobility A_{ij}. In the same way, constraints (6.27) impose that flow from predicted mobility vertices $(h,k) \in S$ cannot exceed the corresponding predicted mobility $F_{h,k}$. Flow from a vertex can be splitted to any outgoing edge. Finally constraints (6.28) impose non-negativity of flow variables for each edge.

The percentage value of users positively predicted $z/\sum_{(i,j)\in R} A_{i,j}$ is used as effectiveness measure and we label it as *prediction matching effectiveness*. Given that its value depends on the distance threshold δ that defines the neighborhood of a location, in the remainder we will refer to the prediction matching effectiveness as PME^{δ}.

Chapter 7

Conclusions

This thesis is motivated by the rise of the highly promising *Mobile Edge Computing* (MEC) paradigm (Chapter 1), bringing IT applications, computational and storage resources to the periphery, or edges, of the cellular mobile network. While attracting a lot of attention in recent years for its beneficial prospects, MEC is still a rather immature technology due to the many challenges that need to be addressed before its implementation.

In this thesis we tackled strategic, tactical and data-oriented planning issues of MEC networks exploiting an optimization point of view, devising mathematical models and combinatorial algorithms.

Part I For what concerns the strategic planning:

- we introduce for the first time at the state of the art a comprehensive framework for the MEC Network Design Problem with mixed integer linear programming extended formulations;
- new NP-Hard combinatorial optimization problems raised in the solution process, that required us to design novel algorithms, exploiting a combination of dynamic variable generation techniques in a matheuristic framework;
- we implemented our methods in prescriptive analytics tools, validating our approach on real world 4G cellular network dataset.

Our models allow to include several fine-grained features of MEC networks, thereby yielding a better matching between the structure of our solutions and a

real setting. Our analyses on the real cellular network dataset provided interesting insights:

- with ad-hoc matheuristic algorithms we could obtain methods for computing high quality solutions in affordable computing time;
- multi-period demands and user mobility must be explicitly taken into account in the models to ensure compliance of users' service level agreement; dynamic planning model also highlights counterintuitive solutions for the assignment of APs to MEC facilities, that do comply with SLA constraints do not form compact clusters;
- we qualified the eligibility of two different VM mobility strategies, namely VM Bulk and Live Migrations: VM Live Migration has proved eligible for the use both with delay-critical and delay-sensitive MEC services, while VM Bulk Migration constantly violates limits on network resources and seems to be a feasible alternative only when the size of VM files to synchronize is small.

Part II For what concerns the tactical planning:

- we proposed a data-driven MEC management framework for the assignment of APs to MEC facilities, exploiting preprocessing, data-mining and validation by simulation;
- the core of our framework entails a new combinatorial problem, representing a novel multi-period variant of a generalized assignment problem;
- we tackled the core optimization problem with a branch-and-price algorithm that, although exact in nature, performs well also as a matheuristics when combined with early stopping;

We verified that instances arising in practice strongly benefit from the explicit use of mathematical programming models in such an optimization core:

- our algorithms find near-optimal solutions very quickly with a good dual bound guarantees: this makes them well suited for *what-if* analyses;
- the joint use of mathematical programming optimization and data driven clustering proved to be beneficial in a *training-and-test* evaluation on real data: in particular, training through mathematical programming algorithm over fine time discretizations derived by the clustering proved to be effective also on testing data coming from real cellular network.

Part III As discussed in the introduction, accurate algorithms require accurate data to work as their are meant. In our strategical problem, the availability of user mobility data in applications is a critical point, due to both data collection and data sensitivity issues. While current approaches to estimate human mobility require a considerable amount of data, we propose a new approach to estimate *user* mobility:

- a new mathematical programming based model, that takes into account very aggregated data;
- adequate accuracy for a specific scenario: rush-hour in a urban-size region.

We found that:

- estimate of users mobility can can be retrieved with adequate accuracy by very aggregated data and simple (a-priori) statistical distributions on human trajectory features, provided suitable mathematical models and combinatorial algorithms are used;
- mathematical programming turned out to be an appropriate tool also in this context: our approach proved to be comparable in accuracy to previous mobility models from the literature requiring at the same time much more aggregated data for training. This makes it far more suitable in our context.

Perspectives While the thesis provided effective methodological tools, and highlighted interesting insights into the creation and management of a MEC network, a further understanding and refinement of MEC technologies is certainly needed to bridge theory to implementation. For instance, different (more involved) topologies might be pertinent in a MEC network design process, as well as different VM mobility policies. These may require deep adaptations in our methods, or even to devise alternative ones. A further promising research direction is the tighter integration of data-driven techniques with ad-hoc optimization methods. For instance, in our case we expect that substantial improvements may be readily obtained by exploiting advanced clustering methods to preprocess data during the tactical planning.

More in general, network planning, being either at strategical, tactical or operational level, is the typical realm of network engineering. As such, decisions are often taken using consolidated techniques, best-practices and robust approaches. We believe that Mobile Edge Computing can become the workbench for an alternative approach, that is, integrating advanced mathematical modelling and algorithms design to support the decision process.

The thesis, indeed, supports such a belief with computational evidence. Custom methods, which up to few years ago were only of academic interest, thanks to the advance of methodological understanding can now be effectively used in real world data analytics. In particular, mathematical programming and data driven techniques proved to be of fundamental importance in our approach and can indeed be seen as appealing elements for next-generation of decision support systems in the context of network optimization.

Bibliography

[1] 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.

[2] CVN Index: Global Mobile Data Traffic Forecast Update, 2016–2021. White Paper, Cisco, 2017.

[3] Milan Patel, Brian Naughton, Caroline Chan, Nurit Sprecher, Sadayuki Abeta, and Adrian Neal. Mobile-Edge Computing introductory technical white paper. White Paper, European Telecommunications Standards Institute, 2014.

[4] Adrian Neal, Brian Naughton, Caroline Chan, Nurit Sprecher, and Sadayuki Abeta. Mobile Edge Computing (MEC); Technical Requirements. White Paper, European Telecommunications Standards Institute, 2016.

[5] Pavel Mach and Zdenek Becvar. Mobile Edge Computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, PP, 2017.

[6] Bernardetta Addis, Giuliana Carello, and Alberto Ceselli. Combining very large scale and ILP based neighborhoods for a two-level location problem. *European Journal of Operational Research*, 231(3):535–546, 2013.

[7] Camilo Ortiz Astorquiza. Multi-level facility location problems. Technical Report, Concordia University, April 2017. Available at `http://spectrum.library.concordia.ca/982462/1/OrtizAstorquiza_PhD_S2017.pdf`.

[8] Mohammad Masdari, Sayyid Shahab Nabavi, and Vafa Ahmadi. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66:106–127, 2016.

[9] Alberto Ceselli, Marco Premoli, and Stefano Secci. Heuristics for static cloudlet location. In *Electronic Notes in Discrete Mathematics*, volume 55, pages 21–24,

2016.

[10] Alberto Ceselli, Marco Premoli, and Stefano Secci. Cloudlet network design optimization. In *Proc. of IFIP Networking conference*, 2015.

[11] Alberto Ceselli, Marco Premoli, and Stefano Secci. Mobile Edge Cloud network design optimization. *IEEE/ACM Transactions on Networking*, 25(3):1818–1831, 2017.

[12] Peter Rost, Albert Banchs, Ignacio Berberana, Markus Breitbach, Mark Doll, Heinz Droste, Christian Mannweiler, Miguel A Puente, Konstantinos Samdanis, and Bessem Sayadi. Mobile network architecture evolution toward 5G. *IEEE Communications Magazine*, 54(5):84–91, May 2016.

[13] Haytham Assem, Teodora Sandra Buda, and Lei Xi. CogNet - initial use cases, scenarios and requirements. White paper, H2020 5G-PPP, 2015.

[14] Kan Zheng, Zhe Yang, Kuan Zhang, Periklis Chatzimisios, Kan Yang, and Wei Xiang. Big data-driven optimization for mobile networks toward 5G. *IEEE Network*, 30(1):44–51, January 2016.

[15] 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.

[16] 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.

[17] 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.

[18] Russell Halper, S. Raghavan, and Mustafa Sahin. Local search heuristics for the mobile facility location problem. *Computers & Operations Research*, 62:210 – 223, 2015.

[19] Iric Gourdin and Olivier Klopfenstein. Multi-period capacitated location with modular equipments. *Comput. Oper. Res.*, 35(3):661–682, March 2008.

[20] Alberto Ceselli, Marco Fiore, Marco Premoli, and Stefano Secci. Dynamic cloudlet assignment problem: a column generation approach. In *Contribution at CTW on graphs and combinatorial optimization*, 2017.

[21] Alberto Ceselli, Marco Fiore, Marco Premoli, and Stefano Secci. Optimized Assignment Patterns in Mobile Edge Cloud Networks. Technical Report, University of Milan, Dipartimento di Informatica, June 2017. Available at `http://www.optimization-online.org/DB_FILE/2017/08/6183.pdf`.

[22] Jay E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20(1):1–66, 1989.

[23] Vincent D Blondel, Adeline Decuyper, and Gautier Krings. A survey of results on mobile phone datasets analysis. *EPJ Data Science*, 4(1):1, 2015.

[24] Alberto Ceselli and Marco Premoli. Towards mathematical programming methods for predicting user mobility in mobile networks. In *Operations Research Proceedings 2016: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*, pages 45–50, 2017.

[25] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[26] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. Challenges and opportunities in edge computing. In *Smart Cloud (SmartCloud), IEEE International Conference on*, pages 20–26. IEEE, 2016.

[27] Arif Ahmed and Ejaz Ahmed. A survey on mobile edge computing. In *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*, pages 1–8. IEEE, 2016.

[28] Ejaz Ahmed and Mubashir Husain Rehmani. Mobile edge computing: opportunities, solutions, and challenges, 2017.

[29] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[30] Yaser Jararweh, Loai Tawalbeh, Fadi Ababneh, and Fahd Dosari. Resource efficient mobile computing using cloudlet infrastructure. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 373–377. IEEE, 2013.

[31] Elijah project. [Online `http://elijah.cs.cmu.edu`; accessed June 2017].

[32] Sergio Barbarossa, Stefania Sardellitti, and Paolo Di Lorenzo. Communicating

while computing: Distributed mobile cloud computing over 5G heterogeneous networks. *IEEE Signal Processing Magazine*, 31(6):45–55, 2014.

[33] Sarah Clinch, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 122–127. IEEE, 2012.

[34] Kiryong Ha, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 166–176. IEEE, 2013.

[35] Debessay Fesehaye, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. Impact of cloudlets on interactive mobile cloud applications. In *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*, pages 123–132. IEEE, 2012.

[36] Zhengyuan Pang, Lifeng Sun, Zhi Wang, Erfang Tian, and Shiqiang Yang. A survey of cloudlet based mobile computing. In *2015 International Conference on Cloud Computing and Big Data (CCBD)*, pages 268–275. IEEE, 2015.

[37] Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, and Sghaier Guizani. Mobile ad hoc cloud: A survey. *Wireless Communications and Mobile Computing*, 16(16):2572–2589, 2016.

[38] OpenFog Consortium. OpenFog Reference Architecture for Fog Computing. White Paper, OpenFog Consortium, February 2017.

[39] Michael Howard. Using carrier ethernet to backhaul LTE. White Paper, Infonetics Research, 2011.

[40] Miguel Angel Alvarez, Frederic Jounay, Tamas Major, and Paolo Volpato. LTE backhauling deployment scenarios. White Paper, NGMN Alliance, 2011.

[41] Ron Nadiv and Tzvika Naveh. Wireless backhaul topologies: Analyzing backhaul topology strategies. White Paper, CERAGON, 2010.

[42] Architectural considerations for backhaul of 2G/3G and long term evolution networks. White Paper, Cisco, 2010.

[43] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. Softcell: Scalable and flexible cellular core network architecture. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages

163–174. ACM, 2013.

[44] Tarik Taleb, Marius Corici, Carlos Parada, Almerima Jamakovic, Simone Ruffino, Georgios Karagiannis, and Thomas Magedanz. EASE: EPC as a service to ease mobile core network deployment over cloud. *IEEE Network*, 29(2):78–88, 2015.

[45] Robert Bradford, Evangelos Kotsovinos, Anja Feldmann, and Harald Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments*, pages 169–179. ACM, 2007.

[46] Patrick Raad, Stefano Secci, Dung Chi Phung, Antonio Cianfrani, Pascal Gallard, and Guy Pujolle. Achieving sub-second downtimes in large-scale virtual machine migrations with LISP. *IEEE Transactions on Network and Service Management*, 11(2):133–143, 2014.

[47] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[48] Jiri Herrmann, Yehuda Zimmerman, Laura Novich, Scott Radvan, and Dayle Parker. *KVM live migration*, chapter 4, pages 201–213. Red Hat, 1993.

[49] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson, and Andrew Warfield. Remus: High availability via asynchronous virtual machine replication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 161–174, 2008.

[50] OpenStack Tacker. [Online; accessed December 2016].

[51] OpenStack Nova. [Online; accessed December 2016].

[52] Vittorio Maniezzo, Thomas Stützle, and Stefan Voß. Matheuristics, volume 10 of annals of information systems, 2010.

[53] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[54] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[55] Jacques Desrosiers and Marco E Lübbecke. A primer in column generation. In *Column generation*, pages 1–32. Springer, 2005.

[56] Miguel Constantino and Luis Gouveia. Reformulation by discretization: Application to economic lot sizing. *Operations research letters*, 35(5):645–650, 2007.

[57] Stefan Hougardy. The Floyd–Warshall algorithm on graphs with negative cycles. *Information Processing Letters*, 110(8-9):279–281, 2010.

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

[59] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[60] Marco Boschetti, Vittorio Maniezzo, and Matteo Roffilli. Decomposition techniques as metaheuristic frameworks. In *Matheuristics*, pages 135–158. Springer, 2009.

[61] Günther R Raidl. Decomposition based hybrid metaheuristics. *European journal of operational research*, 244(1):66–76, 2015.

[62] Andreas Klose and Andreas Drexl. Facility location models for distribution system design. *European journal of operational research*, 162(1):4–29, 2005.

[63] Alireza Boloori Arabani and Reza Zanjirani Farahani. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, 2012.

[64] Gianfranco Guastaroba and Maria Grazia Speranza. A heuristic for BILP problems: the single source capacitated facility location problem. *European Journal of Operational Research*, 238(2):438–450, 2014.

[65] Tingying Wu, Feng Chu, Zhen Yang, and Zhili Zhou. A Lagrangean relaxation approach for a two-stage capacitated facility location problem with choice of facility size. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 713–718. IEEE, 2015.

[66] Suda Tragantalerngsak, John Holt, and Mikael Rönnqvist. An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123(3):473–489, 2000.

[67] Stefan Nickel and Francisco Saldanha da Gama. Multi-period facility location.

In *Location science*, pages 289–310. Springer, 2015.

[68] Alan P Reynolds, Graeme Richards, Beatriz de la Iglesia, and Victor J Rayward-Smith. Clustering rules: a comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):475–504, 2006.

[69] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2017. R package version 2.0.6.

[70] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[71] Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

[72] IBM corp. *IBM ILOG CPLEX 12.6 User Manual*, 2013.

[73] John E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, Nov 1990.

[74] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[75] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[76] ANR ABCD Project. [Online. Available at `http://abcd.lip6.fr`; accessed December 2017].

[77] Balázs Cs Csáji, Arnaud Browet, Vincent A Traag, Jean-Charles Delvenne, Etienne Huens, Paul Van Dooren, Zbigniew Smoreda, and Vincent D Blondel. Exploring the mobility of mobile phone users. *Physica A: statistical mechanics and its applications*, 392(6):1459–1473, 2013.

[78] James P Bagrow and Yu-Ru Lin. Mesoscopic structure and social aspects of human mobility. *PloS one*, 7(5):e37676, 2012.

[79] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Identifying important places in people's lives from cellular network data. In *International Conference on Pervasive Computing*, pages 133–151. Springer, 2011.

[80] Kenneth Church, Albert G Greenberg, and James R Hamilton. On delivering embarrassingly distributed cloud services. In *HotNets*, pages 55–60. Citeseer, 2008.

[81] Victor Bahl. Mobile gaming. MobiGames 2012 Keynote Speech, 2012.

[82] Dirk Lindemeier. MEC Proofs of Concept. Technical report, European Telecommunications Standard Institute.

[83] Stefano Secci, Patrick Raad, and Pascal Gallard. Linking virtual machine mobility to user mobility. *IEEE Transactions on Network and Service Management*, 2016.

[84] EC H2020 5G infrastructure PPP. pre-structuring model, version 2.0. Technical report, The 5G Public Private Partnership.

[85] Angelo Furno, Marco Fiore, and Razvan Stanica. Joint spatial and temporal classification of mobile traffic demands. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017.

[86] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[87] 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. Springer, 2005.

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

[89] 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.

[90] Silvano Martello, David Pisinger, and Paolo Toth. New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research*, 123(2):325–332, 2000.

[91] Silvano Martello and Paolo Toth. Algorithms for knapsack problems. *North-Holland Mathematics Studies*, 132:213–257, 1987.

[92] 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.

[93] 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.

[94] 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.

[95] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[96] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. Modelling the scaling properties of human mobility. *Nature Physics*, 6(10):818–823, 2010.

[97] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[98] Samuel A Stouffer. Intervening opportunities: a theory relating mobility and distance. *American sociological review*, 5(6):845–867, 1940.

[99] Marc Barthélemy. Spatial networks. *Physics Reports*, 499(1):1–101, 2011.

[100] Lun Wu, Ye Zhi, Zhengwei Sui, and Yu Liu. Intra-urban human mobility and activity transition: Evidence from social media check-in data. *PloS one*, 9(5):e97010, 2014.

[101] Anastasios Noulas, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. A tale of many cities: universal patterns in human urban mobility. *PloS one*, 7(5):e37027, 2012.

[102] Filippo Simini, Marta C González, Amos Maritan, and Albert-László Barabási. A universal model for mobility and migration patterns. *Nature*, 484(7392):96–100, 2012.

[103] Xiao Liang, Jichang Zhao, Li Dong, and Ke Xu. Unraveling the origin of exponential law in intra-urban human mobility. *Scientific reports*, 3, 2013.

[104] Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

[105] A Paolo Masucci, Joan Serras, Anders Johansson, and Michael Batty. Gravity versus radiation models: On the importance of scale and heterogeneity in commuting flows. *Physical Review E*, 88(2):022812, 2013.

[106] Vasyl Palchykov, Marija Mitrović, Hang-Hyun Jo, Jari Saramäki, and Raj Kumar Pan. Inferring human mobility using communication patterns. *Scientific*

reports, 4, 2014.

[107] Andres Sevtsuk and Carlo Ratti. Does urban mobility have a daily routine? Learning from the aggregate data of mobile networks. *Journal of Urban Technology*, 17(1):41–60, 2010.

[108] Francesco Calabrese, Giusy Di Lorenzo, and Carlo Ratti. Human mobility prediction based on individual and collective geographical preferences. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 312–317. IEEE, 2010.

[109] Apollinaire Nadembega, Abdelhakim Hafid, and Tarik Taleb. A destination and mobility path prediction scheme for mobile networks. *IEEE Transactions on Vehicular Technology*, 64(6):2577–2590, 2015.

[110] David Stynes, Kenneth N Brown, and Cormac J Sreenan. A probabilistic approach to user mobility prediction for wireless services. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*, pages 120–125. IEEE, 2016.

[111] Chaogui Kang, Xiujun Ma, Daoqin Tong, and Yu Liu. Intra-urban human mobility patterns: An urban morphology perspective. *Physica A: Statistical Mechanics and its Applications*, 391(4):1702–1717, 2012.

[112] Gerardo Chowell, James M Hyman, Stephen Eubank, and Carlos Castillo-Chavez. Scaling laws for the movement of people between locations in a large city. *Physical Review E*, 68(6):066102, 2003.

[113] Nimrod Serok and Efrat Blumenfeld-Lieberthal. A simulation model for intra-urban movements. *PloS one*, 10(7):e0132576, 2015.