UNIVERSITÀ
DEGLI STUDI
FIRENZE


Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Telematics and Information Society


# Analysis and development of algorithms for cloud and smart city

Settore Scientifico Disciplinare ING-INF/05


*Candidate*
Claudio Badii

*Supervisors*
Prof. Paolo Nesi

Dr. Pierfrancesco Bellini

*PhD Coordinator*
Prof. Luigi Chisci



ciclo XXX, 2014-2017

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

*A Silvia*

Research is to see what everybody has seen
and think what nobody has thought.
*Albert Szent-Györgyi*

# Acknowledgments

I would like to acknowledge the efforts and input of my supervisor,
Prof. Paolo Nesi, and all my colleagues of the Disit Lab who were of great help during my research.

In particular, my thanks go to Pierfrancesco Bellini, Michela Paolucci, Riccardo Mariucci, Irene Paoli and Angelo Difino who collaborated on the main parts of my research work, to Imad Zaza who shared with me this research path and all the others who were at Disit Lab when I started researching.

Obviously, I thank my parents Massimo and Viviana, who made this possible.

Finally, I thank Silvia, to whom I dedicate this thesis, who suffered and enjoyed with me throughout my academic career that has taken me to this goal.

# Abstract

This thesis is concerned with cloud computing and big data for smart cities.

As far as the cloud is concerned, a framework has been developed, which can create patterns relating to the workload of a virtual machine for a certain period of time and for all the resources that are considered useful during the simulation phase. Using these patterns, it was possible to simulate the workload of a datacenter and find the best allocation of its virtual machines using heuristics to solve the Vector Bin Packing problem. All phases, i.e., the insertion of the datacenter's characteristics, the simulation of the datacenter and the visualization of the results are supported by a webapp.

As far as smart cities are concerned, an application has been developed that uses the model and tools made available by the Km4City framework. Based on the application architecture, a "Sii-Mobile Mobile App Development Kit" has been created, which allows other developers to create their own module to be integrated into the already developed application. A machine learning algorithm has been developed, based on data relating to parking lots with controlled access (large paid parking lots with bar), which carries out daily training of a "Bayesian Regularized Neural Network", which allows to generate predictions (every 15 minutes) of how many free spaces will be available in a specific parking lot, one hour after the prediction has been made. This algorithm has been successfully implemented within the application to make the data related to the predictions generated available to users.

# Contents

**Contents** **vii**

**Introduction** **1**

**I Part One** **5**

**1 Cloud Computing** **7**
1.1 Concept . . . . . 7
1.2 Virtualization . . . . . 11
1.3 Simulators . . . . . 12

**2 State of the art** **15**
2.1 Software Simulators . . . . . 15
2.2 Allocation of Virtual Machines . . . . . 28
2.2.1 Multi-Dimensional Multiple-Choice Multi-Knapsack problem . . . . . 29
2.2.2 Bin Packing Problem . . . . . 32
2.2.3 Over-provisioning . . . . . 35
2.2.4 Item Centric Heuristics . . . . . 37
2.2.5 Bin Centric Heuristic . . . . . 40

**3 Pattern Generator** **43**
3.1 Concept . . . . . 43
3.2 Workload Models . . . . . 44
3.3 Virtual Machine Allocation . . . . . 62

**4 Icaro Project** **65**
4.1 Introduction . . . . . 65
4.2 Architecture . . . . . 66
4.3 Icaro Knowledge Base . . . . . 67
4.4 Supervisor & Monitor . . . . . 69
4.5 Smart Cloud Engine . . . . . 72

**5 Icaro Cloud Simulator - ICLOS** **75**
5.1 Requirements . . . . . 75
5.2 General Architecture . . . . . 76
5.3 Cloud workload . . . . . 80
5.4 Experimental results in simulating . . . . . 82
5.5 Experimental results in VMs allocation . . . . . 84

**II Part Two** **89**

**6 Smart City** **91**
6.1 Concept . . . . . 91
6.2 Knowledge Model 4 City - Km4City . . . . . 93
6.3 Sii-Mobility Architecture . . . . . 96

**7 Mobile Application Developer Kit** **99**
7.1 Architecture . . . . . 99
7.2 Modularity and Dynamicity . . . . . 102
7.3 Personalization and Profiling . . . . . 103
7.4 Alerting and Tracking . . . . . 104
7.4.1 Tracking Service . . . . . 104
7.5 Multiplatform . . . . . 108
7.6 Usability test . . . . . 109
7.6.1 Objective . . . . . 109
7.6.2 Organization . . . . . 109
7.6.3 Sample . . . . . 110
7.6.4 Test Descripton . . . . . 111
7.6.5 Test Results . . . . . 112
7.6.6 Use of App . . . . . 113

**8 Predicting Free Parking for ParkingSearcher Module 115**
8.1 Introduction . . . . . 115
8.2 State of the Art . . . . . 116
8.3 Data Description . . . . . 119
8.4 Features . . . . . 124
8.5 Forecasts Techniques . . . . . 125
8.5.1 Artificial Neural Networks with Bayesian Regularization 127
8.5.2 Support Vector Regression . . . . . 128
8.5.3 ARIMA models . . . . . 129
8.6 Evaluate Tecniques with MASE . . . . . 129
8.7 Experiments and Results . . . . . 130

**Conclusions 139**

**Publications 145**

**Acronyms 147**

**List of Figures 151**

**List of Tables 153**

**Bibliography 155**

# Introduction

This thesis describes the PhD activity carried out at DISIT laboratory (Distributed Data Intelligence and Technology Lab.) of the Department of Information Engineering (DINFO) at the University of Florence. The work done is composed of two parts.

The first part concerns Cloud Computing and was developed within the iCaro project in collaboration with Computer Gross Italia, Liberologico and CircleCap.

Most of the Small and Medium-sized Enterprises (SME) are based on old architectures hosted on their local servers. A clear advantage offered by Cloud Computing for SMEs' is the cost reduction, increment of flexibility, enhance and accelerate the renewal. Housing and hosting solutions available are rigid, have inertia adaptation to new requirements, in increments of market, require huge investments in infrastructure and/or re-engineering of processes and management software. For SMEs' need to go towards the concept of business as a service.

iCaro aims (with an action of industrial research, innovation and experimental development) to produce prototypes of innovative technology solutions to solve these difficulties, providing an integrated and gradual access to cloud services (i.e. business platform) as a service, with customized cost models and consumption, accessible to the business owner.

Creating and maintaining a Cloud Computing infrastructure, such as that of the iCaro project, can lead to significant costs for the company that will have to manage it. If the physical machines in the infrastructure increases, the costs also increase, as these machines must be powered and the environment, in which they are stored, cooled. Therefore, reducing the number of physical machines can lead to cost savings, and this can be done by optimizing the allocation of virtual machines in fewer physical machines, taking

great care not to compromise infrastructure performance.

Possible allocation strategies cannot be tested directly within the infrastructure that offers services to customers so as not to compromise their data from an integrity and privacy point of view. The most convenient solution is to use a simulator that calculates the best allocation in a virtual environment and then make the necessary changes in the actual structure. Currently, simulators in the literature do not allow the simulation of complex Business Configuration (needed to provide Business Platform as a Service to SMEs'), nor address Service Level Agreements, complex/real workload pattern models, with the aim of exploring, assessing and predicting the best resource allocation based on consumption of resource in the real cloud infrastructure for a long time ahead.

For this reason, a cloud simulator has been developed that makes it possible to compensate for the missing features in those already developed. Furthermore, a framework has been developed, which can create patterns relating to the workload of a virtual machine for a certain period and for all the resources that are considered useful during the simulation phase.

In Chapter 1 an introduction is made to the new concept of cloud computing that began to expand from last years to the present day, highlighting how machine virtualization technology can lead to high savings in terms of energy consumption and data center maintainability. To take full advantage of virtualization, the best allocation of all virtual machines in the datacenter should be found, but to do the allocation tests it is not possible to move machines directly into the production environment. So, there is the need for a simulator that allows to perform such tests without damaging the datacenter or violating the privacy of the data contained in it.

In Chapter 2 an analysis of the literature is made to find the best state-of-the-art simulator. This state of the art shows that many simulators are created to perform simulations with low-level workloads, with low characterization of the virtual machines considered and without high-level connections between the various virtual machines. For the iCaro project we need a simulator that can simulate high level workloads, virtual machines that are strongly characterized and connected in different ways to create business configurations. The remainder of this chapter analyzes the most popular methods for solving allocation problems such as the Knapsack Problem and Bin Packing Problem, focusing on heuristic methods for solving the latter because it is more similar to the problem of allocating virtual machines within

physical hosts.

Chapter 3 deals with one of the main problems of simulators in the state of the art of Chapter 2: the lack of real workload in order to simulate virtual machines within a data center. Starting from the workloads of real machines in the DISIT Lab (where this thesis has been realized) an algorithm was developed, together with the student Riccardo Mariucci, which generates patterns to be reused in the simulation phase. These patterns allow to realize a simulation with a workload that correlates between them all the resources that are deemed necessary and can be considered high level because it can be associated with a certain type of virtual machine (i.e. web server, processing, balancer), in order to group it with others to create a business configuration.

Chapter 4 briefly describes the general architecture of the iCaro project and which other tools of the iCaro project are relevant and used to carry out the simulation.

Finally, the new developed simulator is described in Chapter 5. First of all, the requirements, it will have to meet in order to overcome the problems found in the other simulators (described in Chapter 2), are listed. The architecture of this simulator and how patterns generated in Chapter 3 are used to perform the simulations, are then shown. The statistics related to the simulation performance and, in particular, to the allocation of virtual machines, with heuristics described in Chapter 2, are finally reported.

The second part concerns Smart City and Big Data and was developed within the Sii-Mobility project for the study of mobility and transport aspects (for the evaluation of service quality, for the study of events) and RESOLUTE H2020 project for resilience aspects, data collection related to mobility, transport system, flows of people in the city and risk assessment.

The Smart City and Big Data concepts are both fundamental in this context, because to be able to make a city really Smart it is necessary to collect an enormous amount of data on it to help citizens in their needs. In order to efficiently collect data from a wide variety of sources, a knowledge model must be developed to represent all the data that can be measured by sensors in the city.

This may not be enough without an architecture that optimally connects all the systems dedicated to this work. These issues are described in the first chapters of the second part with the description of Knowledge Model for City (KM4City) and Sii-Mobility architecture. Once the data has been collected, they must be made available to citizens and content creators must

be able to create functional tools for use on mobile devices quickly and easily. For this reason, a Mobile Application Developer kit has been developed that allows developers to create mobile applications, based on the Sii-Mobility architecture and the Km4City framework, simply and fast.

Finally, it is demonstrated in the last chapter that, in such a system, it is also possible to make predictions about the future values of the data that have been recorded to offer citizens a service that makes it evident that they live in a Smart City.

Chapter 6 describes the concepts related to smart cities and then describes the Km4City framework developed within the DISIT Lab, which forms the basis of the Sii-Mobility architecture for collecting and enriching data from all available sensors in the cities covered by the project.

Chapter 7 describes the Mobile Application Developer Kit developed to enable faster implementation of modules for the information contained in the Sii-Mobility project. With this Kit is given the opportunity to develop your own module and insert it into the mobile platform already created, so as to completely skip the study of how a hybrid application should be made, focusing the developer's attention on the logic and data of the new module. This chapter also describes the operation of the service that allows to acquire data from the sensors of the devices that install the applications generated with the Kit. Finally, the results of an usability study on the *Tuscany where, what... Km4City* app (developed with the Kit), are reported.

Finally, Chapter 8 describes the logic inside the module that allows to view free parking spaces in real time and with a forward prediction of one hour, inside the *Tuscany where, what... Km4City* app. Starting from the data on free parking places in some controlled access car parks in the municipality of Florence, on weather, on traffic and on time (meaning day of the week, parking hour, etc.), together with the colleague Irene Paoli, a machine learning algorithm has been developed which, by carrying out daily training on the data prior to the day considered, manages to make a prediction at one hour ahead of the free slots of the main parking spaces in the city.

# Part I

# Part One

# Chapter 1

# Cloud Computing

This chapter briefly describes the technologies that have contributed to the exponential development of Cloud Computing in the early years of this decade and the terminologies that have been created to describe its functionalities and components. Following, virtualization, one of the most important technologies for Cloud Computing growth, is analyzed to understand how it can be used to reduce the costs of such an infrastructure. A first introduction is given to how simulators can be used in conjunction with virtualization to save energy consumption by the Cloud Computing infrastructure.

## 1.1 Concept

The concept of Cloud Computing (CC) dates back to John McCarthy's vision stated at the MIT centennial celebrations in 1961:

> *If computers of the kind I have advocated become the computer of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry.*

Fifty years later the NIST in [94] and other authors in [130] defining the CC as:

> *A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, server, storage, applications and services) that can be rapidly provisioned and released with minimal management effort of service provider interaction to external customers over the Internet.*

In addition to this definition, the NIST [94] lists 5 features that must be possessed by any structure that defines itself as a cloud:

**On-demand self-service** A consumer can **unilaterally** provision computing capabilities such as server time and network storage, as needed **automatically** without requiring human interaction with each service provider.

**Broad Network Access** Capabilities are available **over the network** and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms.

**Resource pooling** The provider's computing resources are pooled to serve **multiple consumers** using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of **location independence** in that consumer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at higher level of abstraction.

**Rapid elasticity** Capabilities can be **elastically provisioned** and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be **unlimited** and can be appropriated in any quantity at any time.

**Measured service** Cloud systems **automatically control and optimize resource** use by leveraging a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled and reported, providing transparency for both the provider and consumer of the utilized service.

In [130] are listed the more important benefits of using CC which could not otherwise be realized and which may be derived from the essential characteristics indicated by the NIST:

**Scalable** Clouds are designed to deliver as much computing power as any user wants. While in practice the underlying infrastructure is not infinite, the cloud resources are projected to ease developer's dependence on any specific hardware

**Quality of Service (QoS)** Unlike standard data centers and advanced computing resources, a well designed Cloud can project a much higher QoS than typically possible. This is due to the lack of dependence on specific hardware, so any physical machine failures can be mitigated without user's knowledge.

**Specialized Environment** It is possible to use the most up-to-date version of libraries, toolkits or create a legacy environment to have greater compatibility with previous versions of the tools: the users can utilize custom tools and services to meet their needs.

**Cost Effective** Users find only the hardware required for each project. This greatly reduces the risk for institutions who may be looking to build a scalable system. Thus providing greater flexibility since the user is only paying for needed infrastructure while maintaining the option to increase services as needed in the future

According to the authors in [18], developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their services or the human expense to operate them. They need not to be concerned about over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under-provisioning for one that becomes wildly popular, thus missing potential customers and revenue.

When the cloud was emerging, there were three possible service models, based on the NIST [94] definition from the lowest to the highest level:

**IaaS - Infrastructure as a Service** The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components.

**PaaS - Platform as a Service** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

**SaaS - Software as a Service** The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser, or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, except for limited users' specific application configuration settings.

Later, each possible specialization of the previous 3 models became a service and maintained the acronym **XaaS** in order to indicate that anything can be a service, such as [105]:

- HaaS - Hardware as a Service
- DaaS - Development, Database, Desktop as a Service
- BaaS or BPaaS - Business (Platform) as a Service
- FaaS - Framework as a Service
- OaaS - Organization as a Service

In particular, this research work will focus on BPaaS, as the iCaro project (Chapter 4) builds on them to facilitate the migration of Small Medium-sized Enterprises (SMEs) on CC. In fact, as reported in [122], CC makes eminent sense for SMEs; however, there are significant technical, operational and organizational issue which need to be tackled before clouds are used extensively at the enterprise level: iCaro project tries to eliminate these problems.

As mentioned above, the CC has taken half a century to move from idea to everyday reality. This period was necessary to develop all the technologies,

services and infrastructure needed to develop CC as a utility, because only recently the hardware and software has been available to support the concept of *utility computing* on a large scale [130]. In [90] they are identified as enabling technologies: virtualization, multi-tenancy and web services:

**Virtualization** is the technology that hides the physical characteristics of a computing platform from the users, instead presenting an abstract, emulated computing platform [11].

**Multitenancy** whereby a single instance of an application software serves multiple clients. This allows better utilization of a system's resources, the requirements of which could otherwise be considerable if the software instance had to be duplicated for each individual client.

**Web services [90]** The definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate over the HTTP protocol used on the Web. Web services help standardize the interfaces between applications, making it easier for a software client to access server applications over a network.

## 1.2 Virtualization

Virtualization is one of the three technologies described in the previous section that has contributed most to the development of CC. Virtualization concept has made progress since 1960s within IBM [11] mainframe systems (M44/M44X systems [130]). Only recently with the increase in computer and network power it is possible to reach a level, with virtualization, that the user cannot distinguish whether he is using a system installed on a physical or a VM [90]. This reason has led virtualization to expand from mainframe systems to computers with the X86 architecture [130]. In addition, the concept has matured a lot since its inception and is now applied to every possible resource in a system: memory, storage, processing, software, networks, as well as services the IT offer [11]. Not being tied to a particular physical machine allows to have an agile and flexible system and to better maintain VMs: for example, each VM can be cloned or moved to another physical machine at any time to increase system scalability or maintain an active service in the event of a physical machine failure [105] [90].

These features make virtualization extremely well suited to a dynamic cloud infrastructure, because it provides important advantages in sharing,

manageability and isolation [105]. By [18] is noted that many traditional SaaS providers developed their infrastructure without using VMs, either because they preceded the recent popularity of VMs or because they felt they could not afford the performance hit of VMs, but since VMs are *de rigueur* in utility computing, the virtualization may make it possible to capture valuable information in ways that are implausible without VMs.

Virtualization also offers an opportunity to reduce the power consumption of the entire data center. The policy with which VMs are allocated can lead to significant savings in terms of energy and maintainability. In particular, CC model has immense potential as it offers significant cost savings and demonstrates high potential for the improvement of energy efficiency under dynamic workload scenarios [32]. This has led to the development of algorithms to find the best allocation of VMs in a data center [33], [130], [113] [78] and to the development of specific simulators to control power consumption based on the migrations of VMs made by these algorithms [77].

The use of simulators is necessary to test a new configuration of VMs, in order to avoid real migrations within the physical machines that operate in production. If the simulation of the workload of each individual VM has been done with a good estimate, compared to the actual workload, it is possible through it, to highlight and thus avoid problems of over-provisioning of VMs within a single physical machine. If on the one hand, by bundling VMs as much as possible within a limited number of physical machines it is possible to shut down unused physical machines saving energy. On the other hand, CC vendors have Service Level Agreements (SLAs) to respect and an unexpected peak in the workload can lower QoS levels provided to customers, mainly if it occurs in a physical machine with an already high resource consumption due to VMs overload.

## 1.3 Simulators

As mentioned in the previous section, companies offering services XaaS must be able to optimize the use of resources available to prevent damage and to minimize energy consumption, but this optimization cannot be tested on the resources that provide services, as it would risk impairment of the data or customer privacy.

For this reason, companies and researchers have developed, in the last years, many cloud simulation tools to test every new progress mentioned

above (optimize the use of resources), avoiding that tests impair production machines. The first step in developing a new simulator is to decide which of the below challenges the developer wants to engage. In fact, most of the simulators already developed are focused on one challenge only. In [133] and in [50] are described the major challenges:

**Security** since all the storage and computations are processed in cloud servers, the importance of confidentiality, data integrity and non-repudiation issues are predominant.

**Cost Modeling** CC has a unique pay-as-you-go service model; through which organizations pay only for what is being used and nothing more. For example, it might be highly beneficial for a company if a brand new high powered server farm could be obtained to introduce a new web based market offering with zero upfront capital.

**Energy Management** due to fluctuation of workload, the average load is only 30% of data center resources and rest of the 70% account putting resources in sleep mode, so the main goal is to run an application with a minimum set of computing resources and maximize the resources that are in sleep mode [83] and [77].

**VM Migration** since CC is a distributed system, when the workload is increased in a particular data center, VM migration helps to prevent performance degradation of the system. [13]

iCaro project is trying to develop solutions that offer services with the BPaaS model, so the simulation of the data center must take place at a higher level than that of considering only the allocation of VMs. Providing a Business Platform can mean offering the customer more systems that build it and can be located on different VMs. For more maintainable management of the system, it is desirable to keep in the same physical machine the VMs that make a platform for a single customer.

Currently, some cloud simulators have been proposed and they are mainly suitable to simulate and assess specific cases and workloads, by adopting specific models for energy, cloud capacity, allocations, networking, security, etc. complex Business Configurations, BCs, need to be allocated on the cloud to satisfy specific demands. These configurations include several hosts and VMs with many services/applications arranged as multi-tier solutions. When several BCs need to be allocated or changed on cloud, the assessment

of free resources into a set of hosts or external storages (CPU, memory, network and storage) cannot be based on the simple estimation of current conditions in a limited time interval, with simple workload patterns.

A deeper simulation of the cloud conditions can be performed by using realistic workload partners and longer time forward is needed. Thus, the same host could have some VMs with heavy work during day-time and a bit less at night-time, while other VMs on the same host could have a complementary behavior in time, with typical weekly, monthly and seasonal behavior. A new allocation on the cloud may imply changes in the distribution of resources exploited in the cloud. The duty of a cloud simulation should include the verification of resource consumption and the assessment of its capability.

Not all the configurations can be viable. For example, by deploying a VM on a given Host may be unfeasible due to lack of resources (i.e., CPU clocks and/or memory). The resources on a given host may be over-assigned by the VMs to exploit the compensation of the different CPU and memory exploitation during the day, week and months of different allocated VMs.

In chapter 2, the state of the art of cloud simulators is studied, looking for a system that allows high-level simulation for the above reasons. In the remainder of the chapter, the literature on the solving algorithms for Knapsack Problem and Bin Packing and on heuristics to solve them within a reasonable time is studied, because they mirror the problem of the allocation of VMs within physical machines.

# Chapter 2

# State of the art

This chapter provides an overview of related work on the Cloud Simulator. For each simulator in the literature, the characteristics that distinguish it from others, its qualities and defects are analyzed in detail. All the data are reported in a table and a report is made on the various simulators found regarding the characteristics that are missing to carry out a simulation with the requirements of the iCaro project. The literature on *Knapsack* and *Bin Packing* problem resolution algorithms is then analyzed as these two problems represent the generalization of the problem of virtual machine allocation. As these problems have NP complexity, part of the chapter is dedicated to heuristics that can solve them within a reasonable time.

## 2.1 Software Simulators

Most part of cloud simulation tools are software because until a few years ago, it was very expensive to create a simulated data center buying the hardware infrastructure on which to run tests. Also, creating a practical test-bed, consisting of a reasonable number of servers (say, 40 machines) can still be out of reach for most researchers when one needs to consider space, power and cooling infrastructure. Nowadays some researchers have taken the effort to create a scaled data center using the Raspberry Pi device reducing drastically the cost of this solution [121], but most of the literature covers articles related to software simulators and therefore the research work

focuses on those last ones.

The present state of the art of cloud simulators is quite wide. Many surveys on cloud simulation have been presented [12, 13, 104, 107, 108, 116] stressing the different kinds of goals meant for such simulators and their mathematical models.

**CloudSim [41]** A new, generalized, and extensible simulation framework that allows seamless modeling, simulation, and experimentation of emerging CC infrastructures and application services. By using CloudSim, researchers and industry-based developers can test the performance of a newly developed application service in a controlled and easy to set-up environment [41].

**Features:**
- Modeling and creating a huge data center, unlimited number of VMs, introducing brokering policy and support the important feature of CC pay-as-you-go model [13]. The time to instantiate an experiment setup with 1 million hosts is around 12s [41].
- It implements generic application provisioning techniques that can be extended with ease and limited efforts [88].

**Cons:**
- Lack of GUI.
- Does not run in real time. So, you would not be able to evaluate how long does it take for your algorithm to decide.
- For complex simulation you must learn the architecture of CloudSim and you must write in Java your simulation. This fact heavily affects the design of experiments, which must be large enough to extract interesting conclusions, but on the other hand must be small enough to fit into 2 GB of memory: Java can only handle at most 2 GB of memory in 32bits systems. However, this limitation does not affect 64 bits systems [99].
- CloudSim is not a framework as it does not provide a ready to use environment for execution of a complete scenario with a specific input [88].
- Users of CloudSim must develop the Cloud scenario it wishes to evaluate, define the required output, and provide the input parameters [88].

- Basically built for single server architecture and become insufficient for real cloud model, deploying different type of applications from different customer [13].
- CloudSim implements an HPC-style workload, with Cloudlets (jobs) submitted by users to VMs for processing. It can be used to simulate a transactional, continuous workload such as a web server or other service, but it lacks a detailed model of such an application [119].

**CloudAnalyst [126]** Supports visual modeling and simulation of large-scale applications that are deployed on Cloud Infrastructures. CloudAnalyst, built on top of CloudSim, allows description of application workloads, including information of geographic location of users generating traffic and location of data centers, number of users and data centers, and number of resources in each data center [126].

**Features:**
- Provides modelers with a high degree of control over the experiment, by modeling entities and configuration options [13].
- Allows modelers to save simulation experiments, input parameters, and results in the form of XML files so the experiments can be repeated [13].

**Cons:**
- CloudAnalyst is favorable for testing the performance of social networking sites such as Facebook, Twitter etc. [50].
- Need to restart of CloudAnalyst for every simulation.
- OS and architecture are transparent to simulation. Only x86 and Linux can be selected.
- Workload based on users' requests, a preliminary study is needed for the number and size of users' requests (Inheritance of CloudSim).
- Allows the configuration of high level parameters only [99].

**GreenCloud [77]** For advanced energy-aware studies of CC data centers in realistic setups. Extracts, aggregates, and makes information about the energy consumed by computing and communication elements of the data center available in an unprecedented fashion. In particular, a special focus is devoted to accurately capture communication patterns of currently deployed and future data center architectures [77].

**Features:**
- Developed as an extension of a packet-level network simulator Ns2 [73].
- Implements a full TCP/IP protocol reference model which allows integration of different communication protocols with the simulation [88].

**Cons:**
- The simulation duration is greatly influenced by the number of communication packets produced as well as the number of times they are processed at network routers during forwarding. As a result, a typical data center simulated in GreenCloud can be composed of thousands of nodes while the Java-based CloudSim and MDCSim can simulate millions of computers [77].
- User of this simulator needs to learn both programming languages i.e. C++ and Otcal to use this simulator, which is a noticeable drawback [13] two different languages must be used to implement one single experiment [99].
- Basically built for single server architecture and become insufficient for real cloud model, deploying different type of applications from different customer [13].
- Although it can support a relatively large number of servers, each server is considered to have a single core and there is no consideration of virtualization, storage area networks and resource management [12].
- Although it has a detailed workload model, it does not include any modeling of virtualization. As such, it is not suitable for virtualized resource management research [119].

**iCanCloud [99]** The main objective of iCanCloud is to predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware, and then provide to users useful information about such costs [99].

**Features:**
- Provides in-depth simulation of physical layer entities such as cache, allocation policies for memory and file system models [99].
- It has been designed to perform parallel simulations, so one experiment can be executed spanning several machines. In [99] this feature is not available yet.

- Several methods for modeling applications can be used in iCanCloud: using traces of real applications [99].

**Cons:**
- It does not provide models for power consumption, although this is included as future work [99].
- The essential capability to simulate the variety of heterogeneous networks involved in the end-to-end cloud service supply chain is lacking in this tool [49].
- Aimed at simulating instance types provided by Amazon without considering the underlying network behavior [121].

**NetworkCloudSim [60]** The main challenge addressed is to develop application and network models that are sophisticated enough to capture the relevant characteristics of real Cloud data centers, but simple enough to be amenable for analysis [73].

**Features:**
- It is equipped with more realistic application models than any other available Cloud simulator [60].
- Network flow model for Cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations [60].
- For helping users to model such communicating tasks, the NetworkCloudlet class was developed that represents a task executing in several phases/stages of communication and computation [60].
- It uses Network Topology class which implements network layer in CloudSim, reads a BRITE file and generates a topological network [88].

**Cons:**
- Although, users can model complex applications in their simulation environment, still the precise execution of such applications depend highly on the underlying network model [60].

**EMUSIM [40]** It is not only simulator; it provides both simulation and emulation of a cloud application. It is developed for SaaS, applications having huge CPU-intensive and which are very costly for actual deployment. For these types of applications, customer needs to analyze before renting the resources [13].

**Features:**
- For improving the accuracy, relevant information of the application is taken out during emulation and is used during the simulation [13].
- Output from the emulation stage and input to the simulation stage are loosely coupled; the emulation generated a performance file that is later translated into a simulation model of the application [40].
- Information that is typically not disclosed by platform owners, such as location of VMs and number of VMs per host in a given time, is not required [88].

**Cons:**
- For emulation it is necessary a real cluster with Xen Hypervisor installed on the host.

**MDCSim [82]** It is a commercial discrete event simulator developed at the Pennsylvania State University. It helps the analyzer to model unique hardware characteristics of different components of a data center such as servers, communication links and switches which are collected from different dealers and allows estimation of power consumption [88].

**Features:**
- It allows measuring power and analyze each layer of 3-layer architecture model and can modify any layer without affecting other layer of the architecture [13].
- It can also model hardware characteristic such as links between two communication nodes and switches connected with these nodes [13].
- It is supplied with specific hardware characteristics of data server components such as servers, communication links and switches from different vendors and allows estimation of power consumption [77].
- The simulator featured IBA and Ethernet communication protocols over TCP/IP and support many functions of IBA. There is no restriction in adding any new communication protocol [13].

**Cons:**
- The drawback of this simulator is that it is commercial (since it is built on CSIM [108], a commercial product [99]), so users need to buy it for full functionality [13].

**DCSim [119]** It is an extensible data center simulator implemented in Java, designed to provide an easy framework for developing and experimenting with data center management techniques and algorithms. It is an event-driven simulator, simulating a data center offering IaaS to multiple clients. It focuses on modeling transactional, continuous workloads (such as a web server), but can be extended to model other workloads as well [119].

**Features:**
- It provides the additional capability of modeling replicated VMs sharing incoming workload as well as dependencies between VMs that are part of a multi-tiered application. SLA achievement can also be more directly and easily measured and available to management elements within the simulation [119].
- The resource needs of each VM in DCSim are driven dynamically by an Application, which varies the level of resources required by the VM to simulate a real workload [119].
- Multi-tier application model which allows simulating interactions and dependencies between VMs, VM replication as a tool for handling increasing workload, and the ability to combine these features with a work conserving CPU scheduler [119].

**Cons:**
- Simulations as large as 10000 hosts and 40000 VMs can be executed in approximately 1 hour [119].

In the tables 2.1 and 2.2 the cloud simulators previously described are compared mainly on these attributes [13]:

**Underlying Platform** Some simulators are built upon any existing simulation framework. The features of existing platform are inherited in the new simulation framework.

**Availability** This is important to know the availability of a simulator is commercial or open source.

**Programming Language** Most of simulator uses Java language for scripting or modeling any system. This is very important, since the users have to learn the language first to use the simulator.

**Cost Modeling** Since pay-as-you model go is one of fundamental service of CC, or utility computing and one of the challenging issues of cloud simulator. The user can model any new policy by using the simulator that has this module.

**Graphical User Interface** Graphical user interface is for visual purpose and for simplicity when modeling. Many of the above simulators have an interactive GUI.

**Communication Model** Communication Model is one of the important in CC, especially for networking within the data center and message passing between applications.

**Simulator Time** This is the execution time of the simulator during testing. This will determine whether simulator is heavy.

**Energy Modeling** Energy modeling is very important in CC research because of huge energy consumption in the data center and various networking elements (router, switch etc.).

**Federation Policy** Since, cloud is distributed system. Many cloud service providers are located in different geographical locations. The federation policy allows coordinating different cloud service provider that supports inter-networking of application and workload migration to benefit high quality of service.

**Services [25]** The type of Cloud Services supported by the simulator (e.g. IaaS, PaaS, and SaaS).

At this list of attributes taken from [13], we added other self-explanatory attributes taken from the papers indicated near the attribute name. The symbol '-' in a cell means that the simulator presents in the column was not compared on the attribute presents in the row with another simulator by the author of the above paper.

Other attributes are not self-explanatory and they are described here:

**Change Structure during Simulation** is set to yes if it is possible, for example, to add one or a set of new VMs during the simulation of a data center. The possibility to perform this operation it is important to analyze changes on the workload of the data center, if independent from previous workload, a VM is added. This operation is different from adding VMs to satisfy peak of workload.

**Full Description of VM** This attribute indicates if VMs in the simulation are fully described with information about, OS, CPU, memory, storage, service and application and not merely with CPU and memory. That is, the value is set to yes if each VM has its own "identity". This attribute is important for high level simulation: if it is possible to add a VM during the simulation that needs a service in other VM, then the workload of the first machine affect the workload of the second.

**Programming Skill for Create Scenarios** It is set to yes, with an indication of what language it is necessary, if the skill is required to generate scenarios of the simulation.

| | **CloudSim** | **Cloud Analyst** | **Network CloudSim** | **EMUSIM** |
|---|---|---|---|---|
| **Underlying Platform** | SimJava | CloudSim | CloudSim | AEF CloudSim |
| **Available** | Open Source | Open Source | Open Source | Open Source |
| **Programming Language** | Java | Java | Java | Java |
| **GUI** | No | Yes | No | No |
| **Cost Model** | Yes | Yes | Yes | Yes |
| **Application Model [77]** | Computation Data transfer | Computation Data transfer | Computation Data transfer | - |
| **Services [25]** | IaaS | IaaS | IaaS | - |
| **Communication Model** | Limited | Limited | Full | Limited |
| **Support of TCP/IP [77]** | No | - | - | - |
| **Energy Model** | Yes | Yes | Yes | Yes |

| | **CloudSim** | **Cloud Analyst** | **Network CloudSim** | **EMUSIM** |
|---|---|---|---|---|
| **Power Saving Modes [77]** | No [77] Yes (create by user) [41] | - | - | - |
| **Simulation Time** | Seconds | Seconds | Seconds | Seconds |
| **Simulation Type [88]** | Event Based | Event Based | Event Based | Event Based |
| **Federation Policy** | Yes | Yes | Yes | No |
| **Models for public cloud providers [99]** | No | No | - | - |
| **Physical models [99]** | No | No | - | - |
| **Change Structure During Simulation** | No | No | No | No |
| **Full Description of VM** | No Yes (if created by user) | No | No | No |
| **Programming Skill For Create Scenarios** | Yes (Java) | No | Yes (Java) | No |
| **Inserting in other project** | Yes (as Java Library) | No | Yes | No |

Table 2.1: Comparison between software cloud simulators previously developed (First Set)

| | **GreenCloud** | **iCanCloud** | **MDCSim** | **DCSim** |
|---|---|---|---|---|
| **Underlying Platform** | Ns2 | SimCan [13] Omnet, MPI [99] | CSIM | - |
| **Available** | Open Source | Open Source | Commercial | Open Source (on github) |
| **Programming Language** | C++/OTcl | C++ | Java/C++ | Java |
| **GUI** | Limited | Yes | No | No |
| **Cost Model** | No | Yes | No | Yes |
| **Application Model [77]** | Computation Data transfer Execution deadline | - | Computation | Multi-tier Sharing workload [119] |
| **Services [25]** | IaaS | IaaS | - | IaaS, PaaS |
| **Communication Model** | Full | Full | Limited | No |
| **Support of TCP/IP [77]** | Full | - | No | - |
| **Energy Model** | Yes Precise (Servers, Network) [77] | No | Rough (Server Only) [77] | Rough (Host Only) [119] |
| **Power Saving Modes [77]** | DVFS, DNS and both | - | No | - |

| | **Green-Cloud** | **iCan-Cloud** | **MDCSim** | **DCSim** |
|---|---|---|---|---|
| **Simulation Time** | Minutes | Seconds | Seconds | Minutes |
| **Simulation Type [88]** | Packet Level | - | Event Based | Event Based [119] |
| **Federation Policy** | No | No | No | No |
| **Models for public cloud providers [99]** | No | Amazon | No | No |
| **Physical models [99]** | Available using plug-in | Full | No | No |
| **Change Structure During Simulation** | No | No | - | No |
| **Full Description of VM** | No | No | - | No |
| **Programming Skill For Create Scenarios** | Yes (Tcl) | Yes (NED) | - | Yes (Java) |
| **Inserting in other project** | No | No | No | No |

Table 2.2: Comparison between software cloud simulators previously developed (Second Set)

According to our analysis,

**CloudSim** [37] is the most popular cloud simulator; developed in Java as

a library it has been used as a basis for other simulators as CloudAnalyst. CloudSim is mainly focused on modeling IaaS aspects, allocating VM into single and multiple datacenters. CloudSim environment allows simulating specific configuration by programming and exploiting a limited number of aspects in modeling cloud resources at level of PaaS and SaaS which are left to high level programming. On the other hand, it has been used to create low level cloud simulators as: **EMUSIM**, **CDOSim** [88].

**CloudAnalyst** [126] is an extension of CloudSim where the GUI and network modeling have been added,

**NetworkCloudSim** [60] is an extension of CloudSim where networks topologies/aspects are addressed supporting HPC, e-commerce and workflows.

**GreenCloud** [77] was based on Ns2 a discrete cloud simulator implementing simulation of full TCP/IP. GreenCloud has been proposed to simulate the energy/power consumption aspects of cloud, and the networking level, thus suitable to simulate workload distributions and make decisions based on mathematical models of energy consumption. GreenCloud does not address higher level aspects of cloud stack and complex BCs. GreenCloud presents a limited graphic user interface and provides low performance in simulation by limiting the size of the simulated clouds configurations.

**MDCSim** [82] addressed the simulation of large scale multitier datacenters, taking into account the aspects related to NIST layers, communication aspects, etc. MDCSim is a library and does not provide a user interface, thus forcing to cloud configuration and related workload programming; therefore it shows limited capabilities in both modeling and simulating complex BCs which change over time.

**iCanCloud** [99] was developed with the aim of solving some of the limitation of CloudSim, GreenCloud and MDCSim. It is based on SIMCAN, OMNET, MPI, and provides the modeling of the infrastructure permitting the modeling of the hypervisor (with its related math model that could be used for any estimation of power consumption, temperature, costs, etc.) and can be executed on parallel instances. iCanCloud has a relevant graphic user interface. At the state of the art, cloud

simulators are mainly based on addressing low level aspects of communicating processes on cloud such as NetworkCloudSim and MDCSim, or on energy consumption as GreenCloud, iCanCloud even modeling the hardware aspects.

To this purpose, a number of simulators are based on direct math model for: energy consumption (relating clock, storage access, and bandwidth to power consumption and temperature), network simulation in terms of packets, storage and database simulation in terms of latency, etc. For such reasons, it is very complex to make a full comparison of the different clouds, since the consumed memory and speed in simulation strongly depend on the resource and the adopted mathematical models [99].

As a result, cloud simulators at present do not allow the simulation of complex BCs, nor address Service Level Agreements (SLAs), complex/real workload pattern models, with the aim of exploring, assessing and predicting the best resource allocation based on consumption of resource in the real cloud infrastructure for long time ahead. The above revised simulators hardly cope with the huge amount of data produced by simulating the behavior along several weeks, taking into account workload patters describing the whole duration of the temporal windows. For example, the analysis of real monitored data from the services, VM and hosts in place, can be used to learn hourly, daily or weekly resource consumption patterns which can be used to produce a forward simulation and prediction.

## 2.2 Allocation of Virtual Machines

The problem of allocation is mathematically generalized by reformulating it through two more general approaches, the **Knapsack** and the **Bin Packing** problems. The two problems have both NP complexities for which requires the use of heuristic techniques to determine a sub-optimal solution. The differences between the two approaches are as follows:

**Knapsack Problem (KP)** A problem of optimization through which we try to maximize the profit deriving from inserting a set of objects inside the *knapsack*. Each object is associated with a value and a weight and the constraint to respect is that the capacity of the knapsack must not be exceeded.

**BP Problem (BPP)** The objective is to use the fewer containers, *bins*, to allocate a set of objects. Obviously each container has a certain capacity while each object requires a certain amount of space to be allocated inside the bin.

In the allocation of VMs within Host, it is desirable to be able to allocate all the VMs created in the smallest number of Host (BP), respecting the constraints imposed by the resources available within each Host (KP). In the KP, however, there is no explicit constraint to ensure that all objects are placed inside the Knapsack: in the case of the allocation of VMs it is evident that each one must be allocated to provide the requested service. The solutions proposed require dividing objects into groups and selecting from each group the object that maximizes profit. Later through some heuristics one or more objects are selected from the backpack and replaced by others within the same group. Therefore, even considering a single object as a group and thus guaranteeing the allocation of all objects in the knapsacks (multi-KP) if possible, most of the proposed heuristics could not be used efficiently.

For these reasons, the approach generally used is to divide the problem into two steps:

- maximize profit against a set of possible SLAs to be respected, so as to allocate it those BCs that ensure maximum functionality using the heuristics of KP resolution
- minimize the number of hosts used, as in the case of allocation, or compacting VMs using the heuristics of BP resolution

### 2.2.1 Multi-Dimensional Multiple-Choice Multi-Knapsack problem

The basic formulation of the KP [92], known in the literature as *0 - 1 Knapsack Problem*, is to choose between a set of objects with a certain value and weight. The goal of the optimization is to identify a configuration that maximizes the sum of the values of the objects contained in the knapsack without violating the maximum capacity constraint with the sum of the weight of the same objects. Let $v_1, v_2, \ldots, v_n$ the values associated with a set of $n$ objects, let them be $w_1, w_2, \ldots, w_n$ the respective weights and $x_1, x_2, \ldots, x_n$ of the indi-

cator variables. The problem, notes the $c$ capacity of the knapsack can then be formulated as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} x_i v_i \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i w_i \leq c \\ & x_i \in \{0,1\}, i = 1, \ldots, n \end{aligned}$$

The basic formulation is not applicable to the problem of VM allocation. It is necessary to consider at least 4 additional points:

**Multi-Dimensional** the capacities to be considered during the allocation will be as many as the resources considered and it will therefore be necessary to define an additional constraint for each resource (capacity) to be respected in the knapsack.

**Multi-Knapsack** the number of hosts to be considered will depend on the size of the data center being simulated.

**Multiple-Choice** certain VMs must provide for an exclusive allocation if they belong to certain categories.

**All VMs Allocated** a constraint must be added to ensure the allocation of all VMs that are present within the simulation.

The problem with these additional points becomes more complex and is referred to as *Multi-dimensional Multiple-Choice Multi-Knapsack Problem* (MMMKP).

Similarly to [43], supposing that the VMs are pooled in groups, it is possible to define:

- G as the number of groups. Each group is composed of $g_i$ VMs.
- Each VM has associated a vector of resource requests $D^{i,j} = (d_1^{i,j}, d_2^{i,j}, \ldots, d_k^{i,j})$, where $k$ $(1 \leq k \leq K)$ indicates the different resources, $i$ the group which the $j-th$ VM belongs.
- Each Host has associated an available resource vector $R^m = (r_1^m, r_2^m, \ldots, r_k^m)$, where $m$ $(1 \leq m \leq M)$ indicates the different Hosts.

The decision variable $x_m^{i,j}$ is 1 if the VM $ij$ is instantiated over the physical hosts m, otherwise is 0. Different from [43] in the following problem, several VMs belonging to the same group can be chosen and each VM must be inserted in at least one host.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{G} \sum_{j=1}^{g_i} \sum_{m=1}^{M} x_m^{i,j} D^{i,j} \\
\text{subject to} \quad & \sum_{i=1}^{G} \sum_{j=1}^{g_i} x_m^{i,j} D^{i,j} \leq R^m, \forall m \\
& \sum_{m=1}^{M} x_m^{i,j} = 1, \forall i, j \\
& x_m^{i,j} \in \{0,1\}, \forall i, j, m
\end{aligned}$$

The problem so formulated (MMMKP) has NP complexity and therefore requires that the optimal solution be approximated in some way by some heuristics. In literature there are various solutions to solve the generalization of the KP, nevertheless, there are not many resolution methods that consider the MMMKP case. In general, although the problem can be reworded as a Linear Programming (LP) problem and the solution obtained adapted to the discreet case, the temporal complexity makes this strategy impracticable in many real cases: almost all articles then define a heuristics to obtain a sub-optimal solution to the problem.

In [53], dynamic programming and branch-and-bound methodologies are combined to produce a hybrid algorithm for the Multi-choice KP (MKP). In [69], the authors approximately solve the Multi-dimensional Multi-choice KP (MMKP) with an algorithm which is based upon reactive local search and where an explicit check for the repetition of configurations is added to the local search. The algorithm starts by an initial solution and improve it by using a fast iterative procedure. Later, both deblocking and degrading procedures are introduced in order (i) to escape to local optima and, (ii) to introduce diversification in the search space. Also in [63] the authors develop an approximate core for the MMKP and utilize it to solve the problem exactly and in [72] the authors proposed two novel algorithms based on Ant Colony Optimization (ACO) for finding near-optimal solutions, although ACO algorithms are known to have scalability and slow convergence issues, here the authors have augmented the traditional ACO algorithm with a

unique random local search, which not only produces near-optimal solutions but also greatly enhances convergence speed.

In the literature on the resolution of the MMMKP the following articles can be highlighted:

In [43] the authors seek to maximise the profit from the allocation of some VMs to a number of hosts. VMs are divided into groups and each group allocates one and only one VM. The algorithm proposed is divided into two phases: in the first phase, a solution is sought by bringing the problem back to that of BP and then we introduce randomness trying to maximize a *goodness*. The work is based on the gain obtained by inserting one machine rather than another into a host, so it is necessary to associate each machine with a gain in order to take advantage of the algorithm proposed as a possible solution to the problem.

In [14], the authors define a new heuristics (A-HEU) to solve the problem of MMMKP. The algorithm uses a distributed resolution system and the authors provide a fairly in-depth study of the computational complexity of the method proposed.

From the literature analysis it is possible to consider the KP as a sort of *specialization* of the problem of BP: generally, in fact, the heuristic proposals try to improve an acceptable solution obtained by solving a problem of BP through heuristics elaborated on the specific problem. For this reason, the literature that deepens the heuristic resolution for BP has been analyzed.

Priority is given to the allocation of VMs only, without worrying about any additional costs and considering only the number of hosts used as an optimization parameter. For this reason, the literature aimed at limiting the number of hosts used was more relevant for the purposes of the discussion.

### 2.2.2 Bin Packing Problem

The basic formulation of BP Problem is to choose how to insert a certain number of objects into multiple containers, *bins*, in order to minimize the number of bins used. Formally, given:

- $O$ a set of $n$ objects, with $w_i$ the weight of object $i$
- $C$ a set of $m$ containers, with $v_j$ the capacity of container $j$

assign each item to one bin so that the total weight of the items in each bin does not exceed $v_j$ and the number of bin used is a minimum. A possible mathematical formulation of the problem is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{m} y_j \\
\text{subject to} \quad & \sum_{i=1}^{n} w_i x_{ij} \leq v_j y_j \ \forall j \in 1, \ldots, m \\
& \sum_{j=1}^{m} x_{ij} = 1 \ \forall i \in 1, \ldots, n \\
& x_{ij}, y_j \in \{0, 1\} \ \forall i, j
\end{aligned}
$$

In [112], the basic formulation is adapted to the problem of static allocation of VMs in physical hosts, known in the literature as Static Server Allocation Problem (SSAP). In this formulation, the problem is to allocate a set of services within some physical servers. Suppose you have $n$ services $i \in I$ that must be allocated using $m$ server $j \in J$. Each service requires a certain capacity of a particular resource $k \in K$ (CPU, memory, bandwidth, etc....). Each $i$ service to be allocated will require a certain amount of the $k$ resource, the amount indicated by $u_{ik}$. Conversely, each server will have some capacity for that particular $k$ resource, indicated with $s_{jk}$. With these considerations, the SSAP problem can be formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{m} c_j y_j \\
\text{subject to} \quad & \sum_{i=1}^{n} u_{ik} x_{ij} \leq s_{jk} y_j \ \forall j \in J, \ \forall k \in K \\
& \sum_{j=1}^{m} x_{ij} = 1 \ \forall i \in I \\
& x_{ij}, y_j \in \{0, 1\}, \forall i \in I, \ \forall j \in J
\end{aligned}
\tag{2.1}
$$

Where variables $c_j$ indicate the cost of keeping the server active while the $y_j$ and $x_i j$ variables are indicator variables that represent respectively the servers used and on which server $j$ the $i$ service is allocated. This formulation is similiar to Vector BP (VBP) or d-Dimensional Vector Packing (d-DVP) defined in [59], but in this case the servers do not have the same capacity C in all resources. Since d-DVP is a generalization of BP, this problem is strongly NP-hard [57].

As for the KP, various heuristics have been researched to find sub-optimal solutions and solve the problem in good time: BP is also used to reallocate any overloaded machines by building schedulers that work online during the life of the data center and that are responsible for migrating and moving such machines. It is therefore essential that the algorithms used in these applications are able to find a good solution quickly enough.

In [45] the authors study the approximability of multidimensional generalizations of three classical packing problems: multiprocessor scheduling, BP, and the KP, obtaining a variety of new algorithmic as well as inapproximability results for these three problems.

In [24] the authors introduce a new general framework for set covering problems, based on the combination of randomized rounding of the (near-)optimal solution of the LP relaxation, leading to a partial integer solution, and the application of a well-behaved approximation algorithm to complete this solution. Applying their general framework they obtain a polynomial-time randomized algorithm for d-dimensional vector packing with approximation guarantee arbitrarily close to ln d + 1.

In [127], the authors focus on dynamic environments where VMs need to be allocated to servers over time, studying simple BP heuristics and using them to place VMs. The authors also note that these placement heuristics can lead to suboptimal server utilization, because they cannot consider VMs, which arrive in the future and they found that combinations of placement controllers and periodic reallocations achieve the highest energy efficiency subject to predefined service levels.

In [54] the authors try to improve the BP algorithm applied to the online allocation of VMs introduce the concept of typology of machine load, differentiating the same machines according to the variability of the workload: in this way they introduce a kind of penalty into the optimization that makes that machines with a fairly constant workload are less subject to migration compared to those with more variable loads. The objective of the work, however, is not to solve the problem of On-line BP, but rather to find a solution to solve the generalized problem of SSAP by introducing the concept of *temporal variability* in the workload. As described in [112] the equation 2.1 represents a single service's resource demand as constant over time and it is possible to consider variations in the workload, and time is divided into a set of intervals $T$ indexed by $t = \{1, \ldots, \tau\}$: Cyclic workloads over time are now represented in the matrix $u_{ikt}$ describing how much capacity service

$i$ requires from resource type $k$ in time interval $t$. This assumption modifies the second constraint of the equation 2.1 as follows:

$$\sum_{i=1}^{n} u_{ikt} x_{ij} \leq s_{jk} y_j \ \forall j \in J, \ \forall k \in K, \ \forall t \in T$$

On the basis of these considerations, the most influential works for treatment are [54] and [101]. In [54] the trend of the workload over time is taken into account, rather than the maximum value of use of each individual resource. In [101] a series of innovative heuristics are presented for solving the problem of BP. Before analysing the approach to the problem, the concept of over-provisioning and a series of heuristics used to solve the problem of allocation is briefly presented.

### 2.2.3 Over-provisioning

Over-provisioning is a general technique used by most providers of telematics services to optimise revenues from a given infrastructure. The concept is very simple: cloud sell more resources than cloud actually do. The use of resources within a telematic system that is shared by several users is almost never homogeneous and there is an alternation of periods in which the system is stressed and periods in which resources are little used.

Obviously, an analysis of a system from this point of view requires that the temporal trend of the workload be considered at the centre of the study. It is therefore clear that, by recasting the problem of allocation, introducing time as a parameter with respect to which the workload varies, it is necessary to modify the model presented in Section 2.1. In particular, the $u_{jk}$ parameter will not be a scalar value representing the maximum value of use for that resource, but rather a matrix of dimensions $j, k$, with $k$ the number of resources of hosts and VMs considered and $j$ the number of time intervals taken into consideration when analyzing that particular workload (as discussed in subsection 2.2.2).

Although it may seem trivial, the concept of over-provisioning is quite delicate: if the estimation of the workload and consequent machine allocation is wrong, all users of the machines contained in the overloaded host will be affected. Any over-provisioning policy must be executed with caution, because the short-term benefit is in conflict with mid-term losses in the reputation of the provider [85]. The cost of overprovisioning is difficult to measure yet potentially equally serious: not only do rejected users generate

zero revenue, they may never come back. For example, Friendster's decline in popularity relative to competitors Facebook and MySpace is believed to have resulted partly from user dissatisfaction with slow response times (up to 40 seconds) [75].

Based on this assumption, systems have been studied to optimize economic resources: as in [84] which authors use an Economically Enhanced Resource Manager (EERM) (defined in [86]) instead of a simple Resource Manager (RM). If there are not enough unreserved resources at a given time, a classical RM will refuse a SLA proposal from the client. However, clients do not always use the total of resources that they have reserved, and these unused resources could be resold to other clients for increase the revenue. When the provider is not able to fulfill all the SLAs that has agreed, the EERM can perform a selective violation of some SLAs for minimizing the economic impact of the penalties [84, 86]. It is therefore essential to have a sufficiently thorough knowledge of the workload or BC before over-compacting VMs.

Two types of BCs were identified based on the use of services resident on machines and the application context of the application: low priority and high priority configurations:

**high priority** In BCs that represent a system designed to host high priority applications, it is essential to limit the overload of machines resident on hosts, in order to limit the degradation of speed introduced by exceeding the thresholds on the available resources of the host. In particular, a parameter must be defined that limits the amount of resources that can be allocated to the load of VM, so that any unexpected peaks can degrade too much the responsiveness of the system.

**low priority** In the case of configurations that do not require a high priority the threshold of available resources can be increased, and in this situation more memory is used than the host has, so that performance will fall in terms of efficiency.

In general a part of the host resources is dedicated to memory reserved for individual VMs and another part of all resources is necessary to ensure the proper functioning of the hypervisor virtualizing machines. In order to over-provisioning without degrading performance for high-priority applications, it is possible to consider a smaller amount of resources as available than would actually be available when considering the amount of resources described above. In this way the over-provisioning window will be found between

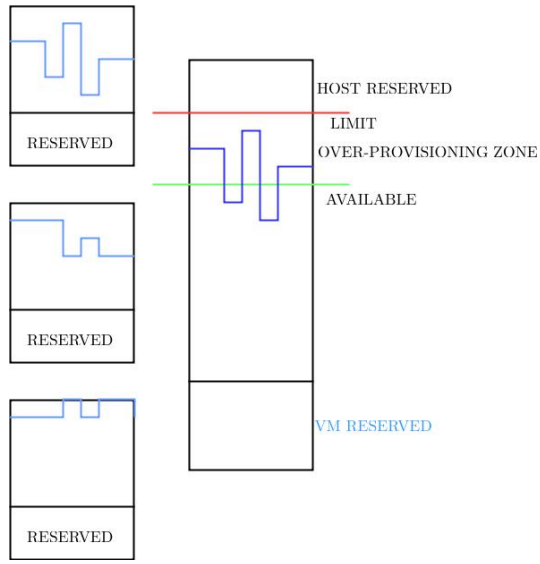



Figure 2.1: Allocate VMs for high priority applications.

the figure value indicated with limit and the available value (see Figure 2.1). Note that the resources used by machines within this window are still available and that being inside this window does not results in the loss of benefits.

### 2.2.4 Item Centric Heuristics

The problem of BP is NP-complete and there is no $\rho - approximation$ with $\rho < \frac{3}{2}$ unless $P = NP$. This implies that there is no algorithm polynomial with absolute small error at will. Generally, two different approaches are used to solve the problem of BP, one based on heuristics and the other based on LP. The LP approach involves either transforming the discrete problem to a linear problem and solving it through *branch and bound* methods or solving the relaxed linear problem, and then round off the solution obtained to bring it back to a discreet case. However, despite the good theoretical results in some studies ( [24] and [114]) report that the problem does not scale when the number of machines increases. The quality of the solution found through an approximate algorithm at polynomial time $A$ is measured through its approximation ratio, indicated with $R(A)$, compared to the excellent

algorithm $OPT$.

$$R(A) = \limsup_{n \to \inf, OPT(L)=n} \frac{A(L)}{OPT(L)}$$

**Heuristic Next Fit**

The Next Fit algorithm assigns the current object to the first bin containing it. If the currently open bin contains no object, a new bin opens. Previous bin are no longer taken into account. Let $x_{ik}$ the variable indicating the assignment of the $k-th$ object to the $i-th$ bin. Let $i$ the total number of binds used and $c$ the remaining bin capacity. In addition, be $w_k$ the capacity required by the object $k$. The pseudo-code is the following:

```
for all objects k {
    consider current bin i
    if (w[k] <= c) {
        x[i][k] = 1
        c = c - w[k]
    } else {
        adding a new bin i = i + 1
        x[i][k] = 1
        c = c - w[k]
    }
}
```

The Next Fit algorithm is an algorithm $2 - approximation$ for the Bin Packaging problem, i.e. the number of bins obtained $z$ is equal to $z \leq 2z^* - 1$ with $z^*$ the best solution to the problem. The execution time is linear with respect to the input data, i.e. $O(n)$. A first improvement could be to recheck any previous bin, so as not to waste space during the allocation.

**Heuristic First Fit**

This variant ensures a better use of space, reconsidering the bin that already contains objects. The pseudo-code is the following:

```
for all objects k {
    set placed to FALSE
    for all bin i {
```

```
            if (w[k] <= c) {
                x[i][k] = 1
                c = c - w[k]
                set placed to TRUE
                break
            }
        }
        if ( ! placed ){
            adding a new bin i = i + 1
            x[i][k] = 1
            c = c - w[k]
        }
    }
```

This heuristics is an algorithm $\frac{17}{10} - approximation$ for the Bin Packaging problem, i.e. the number of bins obtained $z$ is equal to $z \leq \frac{17}{10}z^*$ with $z^*$ the best solution to the problem.

From this heuristics it is possible to derive another one almost immediately, the **First Fit Decreasing** (FFD). In fact, thinking about the procedure one could think of ordering objects in a descending way compared to the required capacity and then applying the First Fit.

The time required to execute the algorithm depends on how long it takes to sort objects and the time it takes to allocate the i-th object in the bin. The total time is therefore $O(n \log n)$.

The **FFD** algorithm ensures that the solution found is $z \leq \frac{3}{2}z^*$ with $z^*$ the best solution to the problem.

### Asymptotic Polynomial Time Approximation Scheme

An **AP-TAS** is a family of algorithms such that for each $\epsilon > 0$ there is a $k^0$ number and an algorithm $(1+\epsilon) - approximation$ algorithm for $K^* \geq k^0$. This family exists for the problem of BP, however, the execution time is quite high despite being polynomial compared to $n$. In particular is worth the following theorem:

> $\forall\ 0 < \epsilon \leq \frac{1}{2}$ there is an algorithm performed in polynomial time with respect to $n$ and that finds an assignment that has at most
> $$k \leq (1+\epsilon)k^* + 1 \ \text{ bin}$$

Due to the high execution time in practice, however, this heuristics is not considered as a valid resolution method for the problem.

### 2.2.5 Bin Centric Heuristic

The work presented in [101] proposes a different algorithm for placing objects in bin called bin-centric approach. In particular the authors compare heuristics based on **FFD** variants empirically with the bin-centric variants they have proposed, obtaining better results and that in some cases reduce the number of necessary bins by about 10%. The construction of the training-set is detailed and tries to reproduce those particular cases in which the performance of the FFD degrades. The idea of the work is motivated by the fact that there are particular instances of problems in which the shape of objects makes heuristics based on FFD less efficient: in particular if objects have a strong correlation between dimensions, performance degrades. Unfortunately, this is quite a common case in the case of VMs where there will generally be a strong negative dependency between its resources compared to the use of the machine. The same case occurs when a service is only active during the day or at night. The problem of BP becomes more complicated as soon as bin capacity and the weight of the objects, increases in dimensionality: it has been demonstrated by various studies that there is no **PTAS** for the problem when the size of the capacities to be considered $d > 1$.

In particular, it has been shown that no algorithm with execution time $(n \log n)$ can give a better approximation than $d - approximation$. It should be noted that if $d > 1$ the heuristics presented so far cannot be applied directly, as it is necessary to assign a weight to the objects and then order them. In [101] two solutions have been proposed to assign a weight to the objects to be ordered called **FFDProd** (Eq. 2.2) and **FFDSum** (Eq. 2.3):

$$w(I) = \prod_{i \leq d} I_i \tag{2.2}$$

$$w(I) = \sum_{i \leq d} a_i I_i \tag{2.3}$$

Where $a = a_1, \ldots, a_d$ is a scale vector whose purpose is to normalise the demand for resources on the various dimensions, represented by variable $I$, and to weight the demand for the various resources against their relevance

during allocation. The determination of the scaling vector $a$ can be derived from two criteria, **FFDAvgSum** (Eq. 2.4) and **FFDExpSum** (Eq. 2.5):

$$a_i = \frac{1}{n} \sum_{j=1}^{n} I_i^j \tag{2.4}$$

$$a_i = \exp(\epsilon \frac{\dot{1}}{n} \sum_{j=1}^{n} I_i^j) \tag{2.5}$$

Analyzing the execution time of the **FFD** algorithm in the multidimensional case it is necessary to consider the shape of the object in order to be able to define which bin will actually contain it. The execution time then becomes $\Omega(n \log n + nk)$ with $k$ the number of bin of the solution.

The pseudo-code of the proposed **FFD Bin Centric** solution is as follows:

```
while there are objects to be allocated {
    adding a new bin
    while some object enters the bin {
        insert the "largest" object into the bin
    }
}
```

Although in the single-dimensional case the algorithms are identical (item-centric vs bin-centric), in the case of multidimensional results depends on how the result is selected the largest object.

### FFD Dot Product

This heuristics defines the larger object as the object that maximizes the scalar product between the vector of the remaining bin capacity and the vector of the object request. Each object is assigned a score defined as

$$score(I^j) = \sum_i a_i I_i^j r(t)_i \tag{2.6}$$

with $r(t)$ indicating the vector of the remaining capacities of the host

### FFD Norm Based

In this case the score is assigned on the basis of the difference between the host residue vector and the machine request according to a certain norm.Using $L2$ you get:

$$score(I^j) = \sum_i a_i (I_i^j - r(t)_i)^2 \tag{2.7}$$

### Grasp-k Variant

A certain randomness can be introduced into the solution search to run the algorithm in parallel and choose the best solution. One possibility is to introduce the $Grasp - k$: with this technique, during the selection phase of the object to be allocated, one is chosen randomly from among the first $k$ larger. The work done by the authors of [101] has also been taken up in [57] with the aim of providing a solution for those problems where bin sizes vary. The implementation, however, envisages normalizing the use of resources with respect to the size of the bin, while maintaining unchanged the heuristic proposals in [101].

# Chapter 3

# Pattern Generator

This chapter explains in detail the steps that are performed to obtain patterns to be used as a load of the virtual machines to be allocated. The patterns generated in this work differ from the others, present in literature, because within each one the resources related to a virtual machine can be correlated and can be considered as high level patterns, as they are different for machines that run different applications (this is very important to make a simulation at Business Configuration level). Finally, it is possible to generate them from the workload of real virtual machines (i.e. those running in production).

## 3.1 Concept

The problems analyzed in the Chapter 2 may seem simple to resolve in their one-dimensional or multidimensional version, which is not time-dependent. For the type of allocation that we want to achieve with this research work, we must consider time-dependent workloads as dimensions of previous problems.

Consider the huge variety of applications that can be found on data center machines. The fundamental concept is that it is not possible to determine in advance what the use of the resources of a certain VM will be, according to the type of applications installed inside it. The problem, therefore, is to determine a load pattern for the use of the resources of a VM as closely as possible to the real one.

An analysis of the literature highlights how the problem of estimating

the workload of one or more VMs is generally solved by considering the latter identical: with the same applications installed or performing the same work. Instead, one should assume that the workload of a VM depends on the services offered by it: for example, cpu-consuming, memory-consuming or disk-consuming. Another problem that can be seen in the data sets of real patterns (supplied in literature) from which artificial workloads should be generated, is that although these refer to different resources measured in the same period on the same machine, they cannot be related to each other, often lacking the metadata necessary for this operation.

In previous literature works [48], [64], [58], authors restrict the field to a single resource (often the CPU) or to the number of tasks that are performed on a supercomputer and focus on the workload of a single VM. These works, moreover, do not try to create workload patterns to be reused later to characterize a pool of VMs that are part of the same BC, but concentrate on predicting what the future trend will be knowing the past [131]. The work that come closest to this is [76], but authors analyze the workloads of all possible VMs looking for a common behavior of some compared to others, in order to create clusters of VMs. In this work, we already have clusters of VMs that correspond to those needed for the creation of a BC desired by the customer and generate workload models to characterize them.

The enhancements proposed in this work are three:

- considering the workload as a function variable in time
- trying to group together the results obtained for some macro-categories of VMs according to the role played by them within a system *n-tier*.
- considering the correlation between the workloads of different resources measured on the same VM over the same period

## 3.2 Workload Models

Due to the variability of possible configurations that a machine can assume when considering the values of all the resources analyzed, the model must necessarily be compressed in order to be reused later. This made it necessary to take a series of intermediate steps in order to simplify the actual performance of each machine and limit the number of possible configurations.

The collection of data related to VMs takes place, in the system in examination, through the use of NAGIOS network manager [6] who stores data

| Time / Resources | $r_1$ | $r_2$ | ... | $r_k$ |
|---|---|---|---|---|
| $t_1$ | $r_{1,1}$ | $r_{1,2}$ | ... | $r_{1,k}$ |
| $t_2$ | $r_{2,1}$ | $r_{2,2}$ | ... | $r_{2,k}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $t_n$ | $r_{n,1}$ | $r_{n,2}$ | ... | $r_{n,k}$ |

Table 3.1: Representation of a time-series object

on Round Robin Database (RRD). The structure of an RRD database is different than other linear databases. In case of linear databases, new data gets appended at the bottom of the database table: its size keeps on increasing. RRD uses a scheme where data is logically organized as fixed size circular buffers: when new data reaches the starting point, it overwrites existing data. This way the size of an RRD database is determined at creation time and always remains constant [8].

The data saved on RRD can be exported in XML and it is possible to choose the time period to export. In this work the selected periods are 1 day, 1 week and 1 month. Depending on the period considered, the granularity of data increases or decreases (see tag $< step >$ in Figure 3.1):

**Day** every 60 seconds

**Week** every 5 minues

**Month** every 30 minues

When an export is performed, a description file containing the details of the resource that has been measured is also generated (see Figure 3.2). The values of this file that will be useful during the generation of the simulated workload are max and min to avoid generating abnormal values.

The exported data is then loaded into *time-series* (TS) objects. There are three TS for each machine. Each TS can be represented through a matrix (see Table 3.1): each column contains the value of a certain resource while the rows indicate the time at which the measurement was made.

The aim of the analysis is to obtain one or more models that can characterize the use of the resources of a analyzed VM during an interval of time:

**Daily Model** where the hours are modeled.

```
<xport>
  <meta>
    <start>1425996900</start>
    <step>60</step>
    <end>1426083240</end>
    <rows>1440</rows>
    <columns>3</columns>
    <legend>
      <entry>CPU_Avg_MIN</entry>
      <entry>CPU_Avg_MAX</entry>
      <entry>CPU_Avg_AVERAGE</entry>
    </legend>
  </meta>
  <data>
    <row><t>1425996900</t><v>3.0000000000e+00</v><v>3.0000000000e+00</v><v>3.0000000000e+00</v></row>
    <row><t>1425996960</t><v>3.0000000000e+00</v><v>3.0000000000e+00</v><v>3.0000000000e+00</v></row>
```

(a)

```
<xport>
  <meta>
    <start>1425478500</start>
    <step>300</step>
    <end>1426083000</end>
    <rows>2016</rows>
    <columns>3</columns>
    <legend>
      <entry>CPU_Avg_MIN</entry>
      <entry>CPU_Avg_MAX</entry>
      <entry>CPU_Avg_AVERAGE</entry>
    </legend>
  </meta>
  <data>
    <row><t>1425478500</t><v>2.4533333333e+00</v><v>2.4533333333e+00</v><v>2.4533333333e+00</v></row>
    <row><t>1425478800</t><v>2.2566666667e+00</v><v>2.4533333333e+00</v><v>2.4140000000e+00</v></row>
```

(b)

```
<xport>
  <meta>
    <start>1423492200</start>
    <step>1800</step>
    <end>1426082400</end>
    <rows>1440</rows>
    <columns>3</columns>
    <legend>
      <entry>CPU_Avg_MIN</entry>
      <entry>CPU_Avg_MAX</entry>
      <entry>CPU_Avg_AVERAGE</entry>
    </legend>
  </meta>
  <data>
    <row><t>1423492200</t><v>6.2516666667e+00</v><v>8.2366666667e+00</v><v>7.0496666667e+00</v></row>
    <row><t>1423494000</t><v>6.5000000000e+00</v><v>7.2450000000e+00</v><v>6.7493333333e+00</v></row>
```

(c)

Figure 3.1: File exported with daytime, weektime and monthtime measurements

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NAGIOS>
  <DATASOURCE>
    <TEMPLATE>check_snmp_storage</TEMPLATE>
    <RRDFILE>/usr/local/pnp4nagios/var/perfdata/disit.org-db-running@192.168.0.75/
    SNMP_WIN_Physical_Memory_Used.rrd</RRDFILE>
    <RRD_STORAGE_TYPE>SINGLE</RRD_STORAGE_TYPE>
    <RRD_HEARTBEAT>8460</RRD_HEARTBEAT>
    <IS_MULTI>0</IS_MULTI>
    <DS>1</DS>
    <NAME>Physical_Memory_Used</NAME>
    <LABEL>Physical Memory Used</LABEL>
    <UNIT>MB</UNIT>
    <ACT>1947</ACT>
    <WARN>2457</WARN>
    <WARN_MIN></WARN_MIN>
    <WARN_MAX></WARN_MAX>
    <WARN_RANGE_TYPE></WARN_RANGE_TYPE>
    <CRIT>2764</CRIT>
    <CRIT_MIN></CRIT_MIN>
    <CRIT_MAX></CRIT_MAX>
    <CRIT_RANGE_TYPE></CRIT_RANGE_TYPE>
    <MIN>0</MIN>
    <MAX>3071</MAX>
  </DATASOURCE>
```

Figure 3.2: Partial representation of the resource description file memory for a generic machine

**Weekly Model** where the days of the week are modeled.

**Monthly Model** where the weeks of the month are modeled.

The TS analysis is performed by subdividing the measured data into subset depending on the TS period being examined. In general called T the time interval analyzed, the objective is to group the measurements in sets that have as discriminating parameter Z, a value depending on the value of T. The size of the sets is indicated by D

$$\begin{aligned} T &= \{day, weekly, monthly\} \\ Z &= \{hour, day, week\} \\ D &= \{24, 7, 4\} \end{aligned} \tag{3.1}$$

The sets obtained are:

$$t_1, z_1 = \begin{cases} i_0^1 \rightarrow Measurements\ made\ between\ 00:00 - 00:59 \\ i_1^1 \rightarrow Measurements\ made\ between\ 01:00 - 01:59 \\ \dots \\ i_{23}^1 \rightarrow Measurements\ made\ between\ 23:00 - 23:59 \end{cases}$$

$$t_2, z_2 = \begin{cases} i_0^2 \rightarrow Measurements\ made\ Monday \\ i_1^2 \rightarrow Measurements\ made\ Tuesday \\ \dots \\ i_6^2 \rightarrow Measurements\ made\ Sunday \end{cases}$$

$$t_3, z_3 = \begin{cases} i_0^3 \rightarrow Measurements\ made\ first\ week \\ i_1^3 \rightarrow Measurements\ made\ second\ week \\ i_2^3 \rightarrow Measurements\ made\ third\ week \\ i_3^3 \rightarrow Measurements\ made\ fourth\ week \end{cases}$$

Taking into account the granularity of the measurements, the cardinality of the individual sets:

$$|i_j^1| = \ 60\ elements$$

$$|i_j^2| = 288\ elements$$

$$|i_j^3| = 336\ elements$$

In order to obtain a general statistical model of the range T, it is necessary to model individually all the D sets obtained from the subdivision of T on the basis of the parameter Z. One solution is to consider each measurement of each set as a class and to count the occurrences. In this way you get a probabilistic model for each Z that composes T, allowing both to model all T and to model some Z. Looking at the daily case, for the creation of patterns to be used as models it would be possible for each $i_j^1$ set to consider the sequences of 60 values that fall into that set. The number of possible classes, considering the measurements to be made in percentage values on the range $[0 - 100]$ and $k$ resources will be:

$$101^{60*k} \simeq 10^{120*k}$$

### Average and Quantization of Values

It is therefore necessary to reduce the dimensionality of the measurements that make up the various sets by introducing some approximation and in this way it is possible to reduce the noise present in the data. The solution introduced was to subdivide each time interval analyzed into sub intervals of $m$ minutes and take the average usage of the resource under the defined range as considered value. The number of values obtained from the initial values will be given by the constant $C$ calculated as:

$$C = \frac{time\ interval\ length\ of\ the\ set}{m}$$

Looking at the daily case with $m = 10$(minutes) and $z = hour$, each hour will be represented by a vector of $\frac{60}{10} = 6$ values, in which each individual value represents the average of the resource values over the previous 10 minutes. Taking an example with the set $i_0^1$ the calculated samples shall be:

$$\underbrace{0:00-0:09}_{sample_1}\underbrace{0:10-0:19}_{sample_2}\underbrace{0:20-0:29}_{sample_3}\underbrace{0:30-0:39}_{sample_4}\underbrace{0:40-0:49}_{sample_5}\underbrace{0:50-0:59}_{sample_6}$$

which can be formalized as:

$$sample_c = \frac{1}{n}\sum_{i=c*n}^{(c+1)*n-1} r_{k,i}$$

where $n$ corresponds to the number of values within the range (for example $n = 10$), $k$ indicates the $k-th$ analyzed resource and $c$ indicates the sample range. Each measurement within the sets $i_j^l$ considered can be represented as a vector of length $k * C$:

$$s_v = [r_{0,0}\ r_{0,1}\ \dots\ r_{0,C}|r_{1,0}\ r_{1,1}\ \dots\ r_{1,C}|r_{k,0}\ r_{k,1}\ \dots\ r_{k,C}] \tag{3.2}$$

where $v$ is the $v-th$ measure of all contained in the set $i_j^l$. Through this reduction in measured values, the number of possible sequences to be considered as classes for the probabilistic model of the daily case are:

$$101^{6*k} \simeq 10^{12*k}$$

which could be a good reduction in measurements, but if the weekly case, where the value of $C = 144$, is considered:

$$101^{144*k} \simeq 10^{288*k}$$

and it should also be kept in mind that the number of resources for which behavior will be simulated in this work is $k = 3$ (CPU, memory and disk), therefore the precise number of classes will be at present:

$$\underbrace{10^{36}}_{daily} \quad \underbrace{10^{864}}_{weekly}$$

| Number | Pattern (CPU-Memory) | Frequency |
|---|---|---|
| 1 | 000000 222222 | 0.75 |
| 2 | 000000 232222 | 0.1 |
| 3 | 000000 233222 | 0.025 |
| 4 | 000001 222222 | 0.1 |
| 5 | 100000 322222 | 0.025 |

Table 3.2: Possible patterns of a web server machine

So the average operation performed on the measured values, helped to reduce the number of classes, but not enough for the model to be generated. The quantization process allows to map the percentage values into $p$ possible values, in order to drastically reduce the possible configurations. This step limits the noise between the measured values, as the neighboring values will now be quantified through the same value, defined according to the $Q$ quantization scheme.

Since the measurements are taken in percentage values in the range $[0 - 100]$, a quantization of these values is carried out on a scale that can have a maximum of $p = 16$ values: by limiting the values to 16 it is possible to represent a class as a string of $k*C$ characters, using the hexadecimal representation. The quantization step can therefore be seen as a transformation applied to each measurement that restricts data space:

$$\overline{x_i} = Q(x_i)$$

with $x$ the $i-th$ measure of the sets expressed as a vector. If considered $p = 10$, the number of possible classes, in day analysis, becomes:

$$10^{120*k} \xrightarrow[average]{} 10^{12*k} \xrightarrow[quantization]{} 10^{6*k}$$

The instance of a string thus constructed represents therefore the situation of a certain hour for that particular machine. Ideally, a large number of identical patterns should be found for each machine, so as to further reduce the classes considered for the model.

Analyzing the time interval between $09:00 - 09:59$ of a VM hosting a web server, the number of patterns is 5: the frequencies of these are shown in the Table 3.2 and displayed in the Figure 3.3.

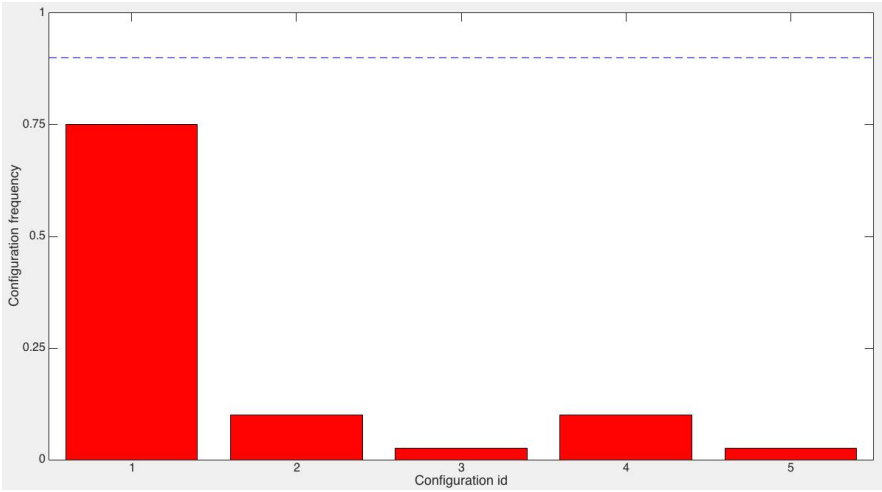



Figure 3.3: Possible patterns of a web server machine

This case is quite simple, as for each set there are few possible patterns, of which different patterns are identified that have a much higher probability than the others. The resulting model, taking direct advantage of the probability tables obtained, would therefore be quite simple and compressed in terms of the number of patterns.

In Figure 3.4, the situation is much more complex: as a result of the quantization step there are numerous possible patterns for each set and all with similar probability of occurrence. The case shown in Figure 3.4 is very frequent for machines that carry out a processing process. If the generated probability tables were used directly, the resulting model would be extremely wide.

### Clustering of Pattern

Analyzing the patterns obtained on some machines of those available, it can be seen that many of them are *similar* to each other. In this context, *similarity* refers to the fact that comparing two patterns between them, the $i-th$ value of a pattern differs by one or at most two levels of quantization compared to the value present in the same $i-th$ position of the second pattern.

See Table 3.2, the difference that exists between the patterns *number* 1

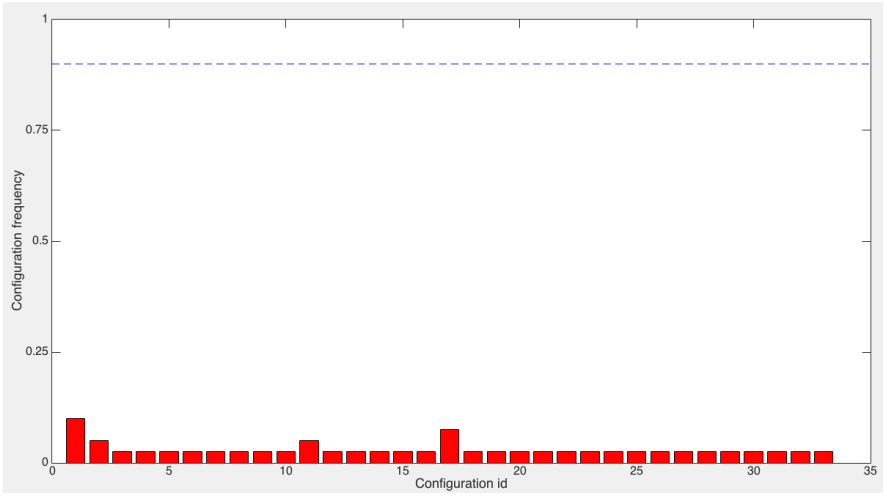



Figure 3.4: Possible patterns of a elaboration machine

and *number* 2 in the second value referring to memory or the difference that exists between the patterns *number* 2 and *number* 3 in the third value referring to memory. It is possible to bring one of the two patterns to be equal to the other by modifying the different value of a single quantization unit. In addition, all 3 patterns examined, when referring only to the CPU unit, are identical.

Let's consider 3 patterns, obtained with the algorithm developed so far, coming from the same machine that deals with intensive processing, measured at a time ranging from 00:00 to 00:59 and representatives of the 3 main resources: CPU, memory and disk.

$$m_1 = \left[\overbrace{111111}^{CPU}\ \overbrace{986679}^{Memory}\ \overbrace{000000}^{Disk}\right]$$

$$m_2 = [111111\ 986559\ 000000]$$

$$m_3 = [111111\ 875679\ 000000]$$

Patterns can also be seen as matrices, where each row indicates a different

resource:

$$m_1 = \begin{bmatrix} CPU \\ Memory \\ Disk \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 9 & 8 & 6 & 6 & 7 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

$$m_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 9 & 8 & 6 & 5 & 5 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$m_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 8 & 7 & 5 & 6 & 7 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Calculating the absolute value of the difference between the elements in the same position, the matrices result:

$$|m_1 - m_2| = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$|m_1 - m_3| = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

These matrices are different but, summing up line by line, in order to measure how much is *similarity* of the patterns according to the resource, we get the same vector:

$$d_p(|m_1 - m_2|) = \sum_{row} |m_1 - m_2| = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \tag{3.4}$$

$$d_p(|m_1 - m_3|) = \sum_{row} |m_1 - m_3| = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \tag{3.5}$$

The method of choosing between one of the two pairs depends on which patterns we consider to be more similar. In this work, those patterns are combined where the individual components have a small difference between the values of (i.e. 1 1 1 0 0 0 ) rather than favoring the aggregation of patterns with a few different components but with a greater difference between

these values of these components (i.e. 0 0 0 1 2 0 ). It was therefore necessary to formulate an aggregation criterion that took these considerations into account and that, with reference to the previous example, selected as aggregates the configurations $m_1$ and $m_2$ but not $m_1$ and $m_3$.

Before formulating the Criterion of Clustering (CoC) the following function was defined:

$$count(X) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} : \; x_l = \sum_{row} \begin{cases} 1 & val_{k,c} \neq 0 \\ 0 & val_{k,c} = 0 \end{cases}$$

which counts the elements different from 0 for each row of a given matrix. Using this function and the $d_p$ vector previously obtained (Eq. 3.4 and Eq. 3.4), we can define the CoC as follows:

$$CoC(m_i, m_j) = \begin{cases} true & count(|m_i - m_j|) \geq d_p(|m_i - m_j|) \\ false & else \end{cases}$$

Taking into account the pairs of patterns seen above:

$$\begin{array}{cc} CoC(m1, m2) & CoC(m1, m3) \\ \Downarrow & \Downarrow \\ count(|m_1 - m_2|) \geq d_p(|m_1 - m_2|) & count(|m_1 - m3|) \geq d_p(|m_1 - m_3|) \\ \Downarrow & \Downarrow \\ \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix} \\ \Downarrow & \Downarrow \\ TRUE & FALSE \end{array}$$

and then, with this criterion the pair $m_1$ and $m_2$ would be clustered.

When clustering occurs, it must be taken into account that the patterns that are merged may not have the same probability of being generated. Therefore, when two patterns are merged, the frequency of the new configuration will be given by the sum of the frequencies of the initial patterns.

$$freq_{m_{clustered}} = freq_{m_1} + freq_{m_2}$$

and the new configuration will be calculated by weighing the starting patterns with weights calculated on the frequencies of the initial patterns:

$$m_{clustered} = m_1 w_{m_1} + m_2 w_{m_2}$$

with weights given by the formulae:

$$w_{m_1} = \frac{freq_{m_1}}{freq_{m_{clustered}}} \quad w_{m_2} = \frac{freq_{m_2}}{freq_{m_{clustered}}}$$

To provide flexibility for the CoC, a tolerance value (Tol) is introduced within the formula of the criterion.

$$CoC(m_i, m_j) = \begin{cases} true & count(|m_i - m_j|) + Tol \geq d_p(|m_i - m_j|) \\ false & else \end{cases}$$

where Tol is a vector with $k$ components. The Tol parameter directly affects the generated model, because by increasing the value of the Tol parameter patterns, that are actually always more distant are considered similar. In the daily case, where $C = 6$ and the quantization intervals are 10 (0 *to* 9), the maximum tolerance value is obtained by setting all possible components of a resource to 9, giving an upper limit for the tolerance of $6 * 9 = 54$.

**Interrupt Clustering**

The second step to perform a clustering is to identify a valid criterion to stop the operation. This step is extremely important because clustering too little the resulting model will still be too complex, vice versa the model obtained will tend to the average value, making the model an unnecessary approximation of the real trend in the use of machine resources.

It is useful, define a Simulation Error (SE), i.e. the error that occurs by increasing tolerance and aggregating in the same cluster configurations that are actually always more distant. Let N, the number of actual measurements of a certain set of Z associated to period of analysis T. Suppose that a probabilistic model is already defined where at each pattern is associated a probability of occurrence (i.e. Table 3.2):

$$model = \{(patt_1, freq_1); (patt_2, freq_2); \dots \ (patt_n; freq_n)\}, \ \sum_{i=1}^{n} freq_i = 1$$

N patterns, sampled according to the probabilities in freq, getting a vector of simulations $s$. Remember that each element $s_i$ is a matrix of C columns and k rows (Eq. 3.3 ):

$$s = [s_1, s_2, \ldots, s_N]$$

Each element of this vector shall be compared with a vector of similar length containing real quantified measurements. To do this, the following matrices are created:

$$S = \begin{bmatrix} s_1 & s_2 & s_3 & \ldots & s_N \\ s_1 & s_2 & s_3 & \ldots & s_N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_1 & s_2 & s_3 & \ldots & s_N \end{bmatrix}$$

$$R = \begin{bmatrix} r_1 & r_1 & r_1 & \ldots & r_1 \\ r_2 & r_2 & r_2 & \ldots & r_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_N & r_N & r_N & \ldots & r_N \end{bmatrix}$$

and matrix D is calculated as follows:

$$\begin{aligned} D &= |S - R| \\ &= \left| \begin{bmatrix} s_1 - r_1 & s_2 - r_1 & s_3 - r_1 & \ldots & s_N - r_1 \\ s_1 - r_2 & s_2 - r_2 & s_3 - r_2 & \ldots & s_N - r_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_1 - r_N & s_2 - r_N & s_3 - r_N & \ldots & s_N - r_N \end{bmatrix} \right| \\ &= \left| \begin{bmatrix} d_{1,1} & d_{2,1} & d_{3,1} & \ldots & d_{N,1} \\ d_{1,2} & d_{2,2} & d_{3,2} & \ldots & d_{N,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{1,N} & d_{2,N} & d_{3,N} & \ldots & d_{N,N} \end{bmatrix} \right| \end{aligned}$$

Each element of $d_{i,j}$ of the matrix D is in turn a matrix. To reduce the SE to a single number, the average of the elements in each $d_{i,j}$ matrix is calculated.

$$\overline{d_{i,j}} = \frac{1}{C * k} \sum_{l=1}^{k} \sum_{c=1}^{C} d_{i,j}^{l,c}$$

Obtaining the $\overline{D}$ matrix:

$$\overline{D} = \begin{bmatrix} \overline{d_{1,1}} & \overline{d_{2,1}} & \overline{d_{3,1}} & \dots & \overline{d_{N,1}} \\ \overline{d_{1,2}} & \overline{d_{2,2}} & \overline{d_{3,2}} & \dots & \overline{d_{N,2}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{d_{1,N}} & \overline{d_{2,N}} & \overline{d_{3,N}} & \dots & \overline{d_{N,N}} \end{bmatrix}$$

On each element of this matrix the average is calculated as follows, obtaining the mean SE:

$$Mean\ SE = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \overline{d_{i,j}}$$

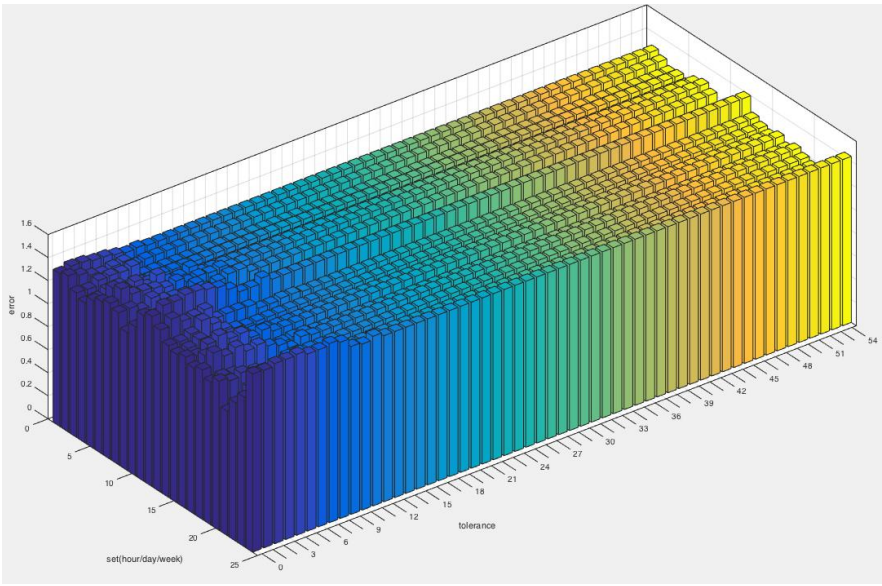



Figure 3.5: Trend of simulation error as tolerance increases

It can be assumed that this error increases as the tolerance increases. In reality, experiments have shown that this measurement does not follow a monotonous trend and the error oscillates up to converge when the tolerance is large enough to make the CoC always true. The simulation error stabilizes because the upper tolerance limit for that particular set has been reached (Figure 3.5), and the variance between simulations also stabilizes (Figure 3.6).

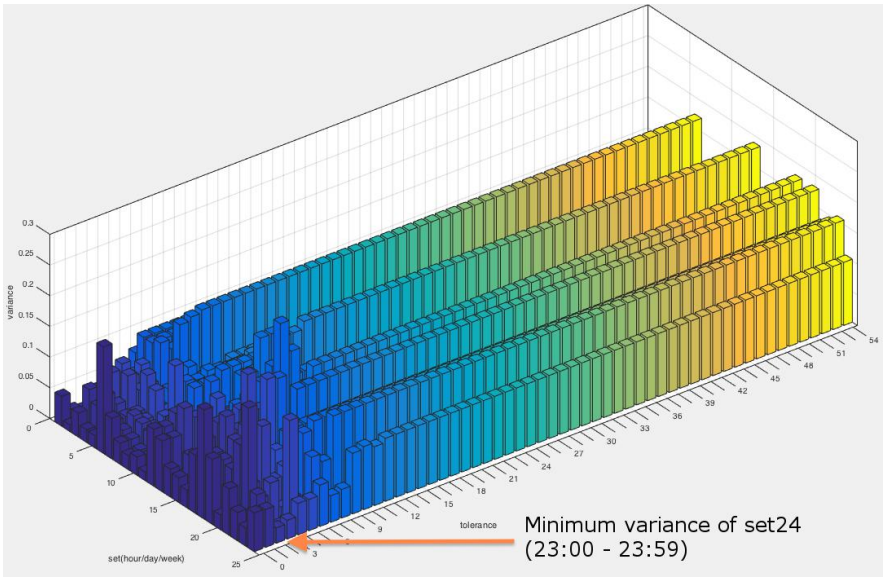



Figure 3.6: Trend of variance as tolerance increases

By progressively increasing the tolerance, the quantities converge until a single cluster is obtained. Aggregating all patterns into a single pattern, the model obtained is the average value calculated on all the initial data patterns (Figure 3.7).

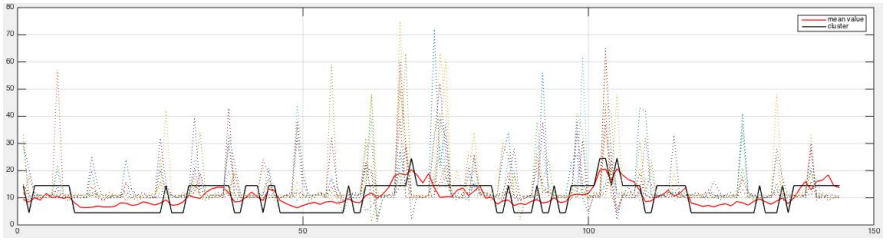



Figure 3.7: Single patterns results in the value model average. In red the mean value, in black the cluster

From Figure 3.8, on the other hand, we can see that the more the model tends to the average value, the more the variance increases. This is a direct consequence of the fact that a model with only one cluster cannot express

all the variability in the use of a machine's resources.

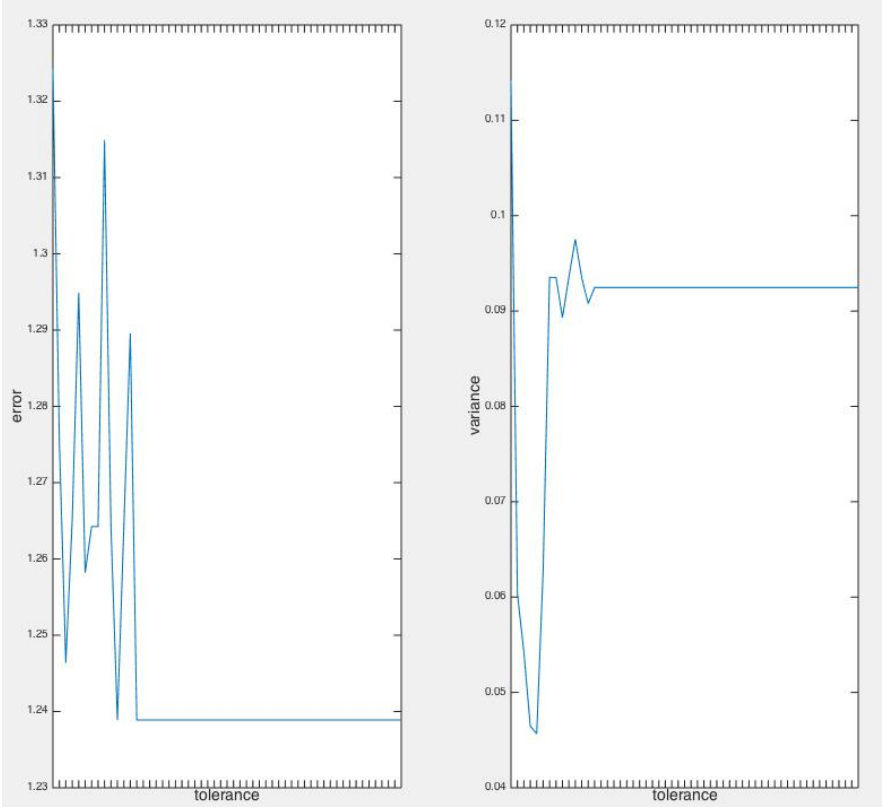



Figure 3.8: Error and variance with respect to tolerance

Remembering that the objective is to build a compressed model of the initial configurations but that it is still fairly faithful to the data, it has made the choice of the parameter to interrupt the clustering on the variance. To compress the model, the tolerance is progressively increased until a single configuration is obtained; the model chosen is the one with the lowest variance.

Using this compression strategy results in a much smaller number of configurations per machine than those obtained after the quantization step alone, see Table 3.3.

| Machine | A | B | C |
|---|---|---|---|
| SiiMobility-Master500-72 | 40.88 | 19.67 | 10 |
| SiiMobility-Node200(1)-70 | 40.83 | 20.77 | 10.04 |
| SiiMobility-Node200(1c-esx4)-92 | 40.79 | 11.50 | 10.75 |
| SiiMobility-Node200(2)-69 | 40.75 | 5.92 | 4.83 |
| SiiMobility-Node200(6)-42 | 40.79 | 3.7 | 3.63 |
| SiiMobility2-Mobility7-205 | 40.71 | 1 | 1 |
| disit.org-db-running | 40.75 | 1.08 | 1.08 |
| ebos0-eclap-bo-scheduler | 40.92 | 1 | 1 |

Table 3.3: In column (A) the number of average clusters per group. In column (B) the number of average clusters per assembly when tolerance is set to zero. In column (C) the number of average clusters per group when tolerance is set using the minimum variance criterion.

### Generation of workloads

At the end of the clustering procedure, for each machine there is a probability model for each possible time interval (similiar to Table 3.2). Considering the definitions in Eq. 3.1 and indicating with $M_i^t$ the resource model for the machine $i$ with respect to the $t \in T$ interval, this can be represented as a set of probability model, each one related to a time range $z \in Z$:

$$M_i^t = \{model_1, model_2, \ldots, model_D\}$$

where each model can be represented as:

$$model_j = \{(c_1, f_1), (c_2, f_2), \ldots, (c_p, f_p)\} \;\; \forall j \in 1, \ldots, D$$

where variable $p$ indicates the number of possible patterns identified for that interval and depends directly on the number of clusters obtained. The variable $c_i$ represents instead a vector such as that of Eq. 3.2: the values will now be represented by quantized levels, according to the results of the clustering process. From the models thus constructed it is possible to generate workloads by sampling $l$ values from each time interval according to the probabilities $f$ of each tuple and chain the values obtained to obtain a workload along $l$.

With this mechanism it is possible to generate an arbitrary number of workloads to be assigned to as many virtual machines. The workloads as-

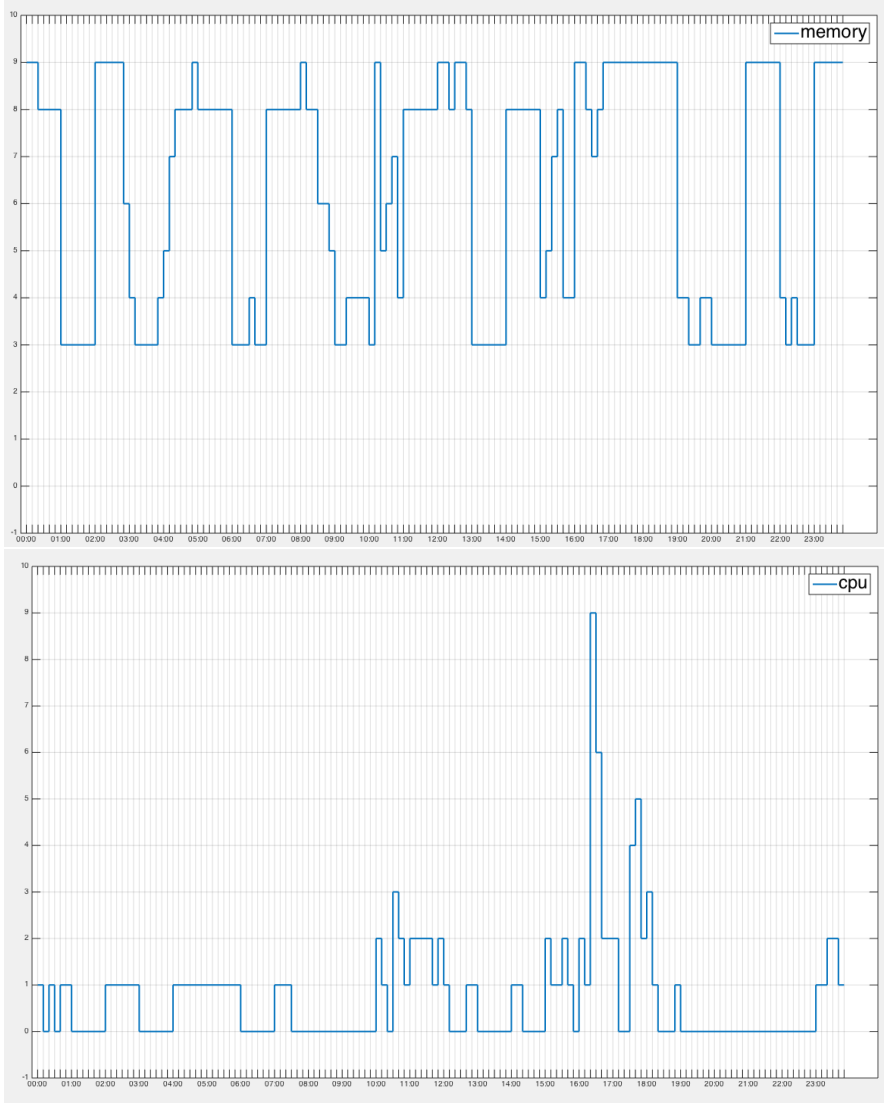



Figure 3.9: Simulated daily workload for memory and CPU of a single VM

signed to each machine will determine, at the time of allocation, the demand for resources for that specific machine in a given time interval (Figure 3.9).

## 3.3 Virtual Machine Allocation

The problem of allocating VMs within the hosts is resolved by considering CPU and memory resources and generalizing the problem as a Vector Bin Packing Problem (Section 2.2). The VM allocation algorithm expects to receive a set of $n$ VMs and $m$ Hosts. Each VM has associated a workload for all the resources considered.

The workload of the resources can be represented through a matrix

$$m_{i,j} = \begin{bmatrix} r_{CPU,1} & r_{CPU,2} & r_{CPU,3} & \dots & r_{CPU,L} \\ r_{Mem,1} & r_{Mem,2} & r_{Mem,3} & \dots & r_{Mem,L} \end{bmatrix}$$

in which the values are found on each row, instant by instant, for a certain resource. This matrix will consist of two lines and $L = C\times$(#in time intervals to be generated) columns. The elements refer to quantized values, so they contain integers numbered from 0 to $\#QuantizationLevels - 1$. These values must be scaled to percentages and then used in conjunction with the maximum values of the machine's available resources. A fixed static value or random value can be added to the scaled values as follows:

$$m_{i,j}^{scaled} = m_{i,j} \times \#QuantizationLevels + \begin{cases} \frac{Q}{2} \\ \\ random(0, Q) \end{cases}$$

where:

$$Q = \frac{100}{\#QuantizationLevels}$$

Based on the resource considered, the percentage values thus obtained are multiplied by the available capacity for that resource in the VM whose workload is to be generated:

$$CPU^{scaled} = m_{CPU,j}^{scaled} \times \#CPU \times CPU\ Speed$$

$$Memory^{scaled} = m_{Mem,j}^{scaled} \times AvailableMemory$$

The allocation process must check whether the number of hosts can at least guarantee the reserved resources for each machine. If this were not the case, the constraints on the SLAs would certainly be breached, since they do not have the minimum resources required and guaranteed in the contract.

The total CPU and memory resources available within each Host, on which the allocation is made, are indicated in MHz and GB respectively. Having re-scaled the use of VM resources compared to the original units of measurement now ensures that they can be normalized with respect to the total resources of the Host considered.

Virtualization management software also requires resources. An adequate amount of CPU and memory must be reserved for the hypervisor to run smoothly. Reserving few resources for the hypervisor causes slowdowns and all those problems of overprovisioning (Section 2.2.3. In order to carry out the allocation taking into account overprovisioning, the following parameters must be provided:

**Max_Risk (H)** Maximum risk for the host, determines, as a percentage, the maximum resources to be used during machine placement. Setting a negative value for the parameter will lower the threshold of resources available for host allocation to ensure greater resistance to the variations from the expected workload. In this case, the over-provisioning window is still lower than the limit of the resources reserved to the host, so even if you stay within this window the virtual machines would not be slowed down (Figure 3.10). If the risk is assigned a positive value, the resulting compaction will generally be greater, as the space available on the machine increases. However, the risk of slowing down also increases, as the over-provisioning window is now overlapping resources reserved for the host (Figure 3.11).

**Max_Over_Time (H)** Maximum over-provisioning time, indicates the maximum time, during allocation, the resource workload can be found in the overprovisioning zone. The value is expressed as a percentage of the maximum allocation time. Note that setting a negative risk can be used to allocate the machines to balance the load rather than trying to optimize as few hosts as possible to solve the allocation problem.

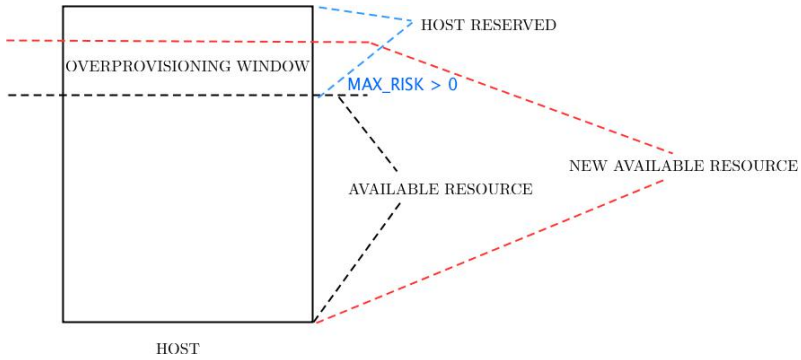



Figure 3.10: Setting a negative risk value ensures better noise resistance for machines. This approach can be used to ensure high responsiveness of applications.

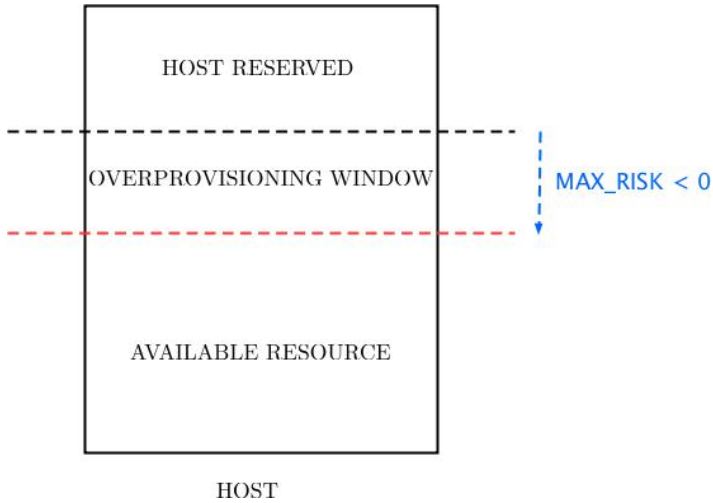



Figure 3.11: By setting a positive value for the risk, compaction increases, but the machine becomes more susceptible to delays caused by changes in the workload not foreseen during allocation.

# Chapter 4

# Icaro Project

This chapter gives a brief overview of the iCaro project describing its architecture and main components that are used by the simulator or pattern generation framework.

## 4.1 Introduction

The complexity of Cloud infrastructures is increasing every year, requiring new concepts and tools to face off topics such as process configuration and reconfiguration, automatic scaling, elastic computing and healthiness control [29]. Market analyzes show that Italian SMEs base their business on non-cloud services hosted on their local servers that are badly adapted to business evolution. They need to invest heavily in infrastructure and / or re-engineering processes and management software and SMEs can not handle this. Cloud can be a benefit for SMEs if it provides the BPaaS, cutting management costs, giving flexibility, favoring and accelerating business processes.

To encourage the adoption of cloud solutions for SMEs, the University of Florence (as DISIT Lab) has developed, in collaboration with several commercial parties, the iCaro platform.

The main objective of the iCaro project is to provide SMEs with a range of intelligent tools that can be adapted to existing cloud solutions and environments to provide greater flexibility to all possible configurations of the cloud environment itself. To mitigate all issues related to maintenance, configuration and management of infrastructure in typical cloud (IaaS, PaaS

and SaaS) solutions or non-cloud solutions, the iCaro project seeks to enclose them in a single BPaaS solution, ensuring that SMEs focus on their core business.

## 4.2 Architecture

The proposed iCaro Cloud Simulator (ICLOS) (Chpater 5) is integrated in the context of the iCaro Cloud platform for Smart Cloud management [29]. In this Section, an overview of the iCaro architecture is reported to highlight the relationships of ICLOS with other components. The iCaro Cloud architecture is reported in Figure 4.1 and includes six main areas:

**Cloud under management** on the left side the real cloud under management (including one or more datacenters) is depicted with its layers: IaaS, SaaS and PaaS. In pure simulation cases this part can be missing.

**Cloud administration area** including one or more commercial or open source Cloud Configuration Managers (CCMs) any kind of cloud brokers as those mentioned above, and Orchestrators (e.g., VCO of VMware, Microsoft cloud solution). In pure simulation cases this part can be missing.

**Supervisor and Monitor (SM)** collects monitoring data from real cloud resources, produces monitoring graphs and charts on demand, etc. Classical data are collected at level of IaaS (e.g., CPU, Memory, storage, network), at level of PaaS (e.g., operating system status), and SaaS (e.g., applicative metrics such as: number of users, number of accesses, number of deploy/download, etc.).

**Knowledge Base (KB)** can be invoked by any CCM or by the Orchestrator. The KB models the cloud knowledge in terms of structures, business configurations, SLA, resources, and corresponding actual values coming from the SM. KB also manages the monitoring tools performing the automated configuration of monitoring issues related to the new resources configured by the CCM. The KB may model real as well simulated cloud configurations, datacenters and conditions. The use of a KB makes easier interoperability among public and private clouds, and/or among different cloud segments managed by different cloud orchestrators or CCM;

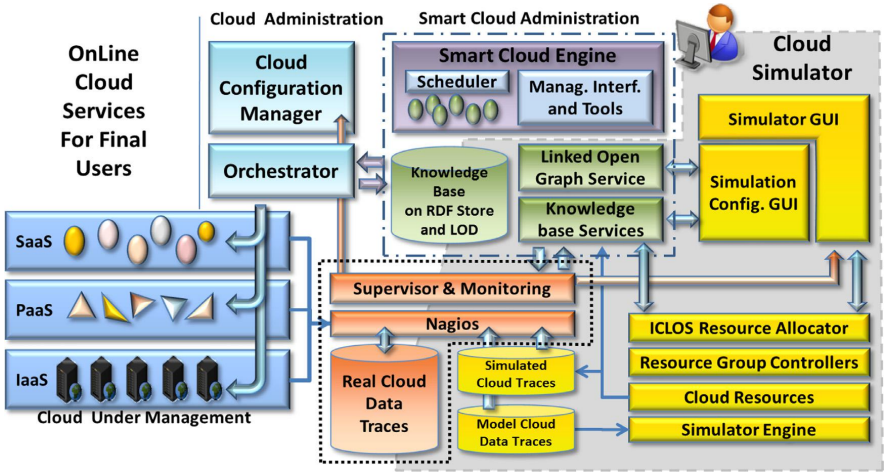



Figure 4.1: iCaro cloud architecture with cloud simulator

**Smart Cloud Engine (SCE)** exploits the Knowledge Base (KB) where the cloud under control, and the simulated clouds are modeled. The SCE allows the execution of making decision processes on a distributed and parallel architecture to assess cloud healthiness and reconfiguration strategies [29]

**ICLOS** simulates cloud conditions hourly, daily, weekly and yearly, taking into account real resource consumption and workload patterns and exploiting complex business configuration modeled into the KB.

## 4.3 Icaro Knowledge Base

The cloud Knowledge Base, KB, stores the general cloud configuration and the status of the cloud under control (as well as of simulated cloud configurations or mixt simulated over a basis of real conditions). The collected model includes services ranging from the data center infrastructure to SW application structure, as well as the applicative metrics definitions and values, referred to BC and SLA. A review on knowledge base usage in the context of cloud can be recovered in [28]. The variety of modeled resources in iCaro KB is higher if compared with the models adopted in the above mentioned

simulators. In ICLOS, the KB is adopted for modeling configuration, taking decision by comparing possible configurations with actions, and it is the very basis of any simulation. Therefore, the KB models: datacenters, hosts (i.e., HW servers), VMs, networks, net devices, SLAs, metrics, users, SW applications, operating systems, etc., and their generic and specific relationships such as: is-a (specialization), needs, hasPart, isPartOf, has SLA, hasNetworkAdapter, isTenantOf, etc. (see Figure 4.2). The model also allows to represent clusters as specific Business Configurations, racks as specific data centers elements, etc. The usage of a KB enables the reasoning on cloud structures and resources by inference on the basis of the specialization (is-a), aggregation, equivalence, subclassing, etc., relationships. On this basis, the KB helps to implement SCE strategies and simulations [29], allows formal verification and validation of resource cloud configuration, discovering and brokering services and resources, reasoning about cloud security, computing capability for horizontal or vertical scaling, thus elastic computing. The KB models and stores, not only the structure of cloud components (infrastructure, applications, and configurations), but also the values of metrics belonging to components and their temporal trends (collected by the monitoring tools with the needed sub-sampling) to be able to answer questions such as "Which host machines can allocate a new VM ?"or "Has the host machine H7897 been over used in the last week?", "Which VM is using most resources in the Host ?". However storing the full history of all metric values on the KB can be too expensive and unnecessary. Only high level metrics values are stored on the KB, while the low level metrics are stored in the monitoring service. The KB is feed by the SM with data regarding the monitored cloud resources (such action is based on NAGIOS monitoring tool). The KB may refer to multiple Nagios installations collecting data from different data centers or segments of data centers. KB stores both the application as a type and the application instances, and the latter can have specific constraints like the number of involved services (e.g., number of front-end web servers). Therefore, to avoid duplicating type/ instance relation (modeled in RDF) and to leverage on the modeling features available in OWL2 to express constraints (e.g., max/min cardinality) it has been decided to represent the application model as an OWL Class. Another need is the possibility to aggregate different applications, servers, VMs to build a complete BC (e.g., an ERP with a CRM) and also to model applications tenants to be able to put application tenants in BCs. The KB has to contain the SLAs associated

with application or application tenants or with a whole business configuration. The SLA has been modeled as a set of Boolean expressions relating high level metrics values belonging to a component with a reference value.

KB Services are provided as REST APIs for accessing, configuring, modeling, loading SLA, and manipulating any cloud element and metric value on an RDF Store (currently an OWLIM-SE instance). When a complex datum (e.g., a complex multitier business configuration) is provided to be stored on the KB via a RDF-XML description, it is first validated against completeness and consistency and then stored into the KB modeling the cloud. The KB provides a SPARQL endpoint allowing posing semantic queries for:

- the SCE assessment of the cloud element healthiness,
- the SCE decision criteria,
- the verification and validation of consistency and completeness of BC/SLA,
- any loading, storing and accessing the simulations, etc.

Every time the KB is configured with a new resource to be monitored (a new host, VM, service, connection, etc.), it automatically sends a corresponding command to the SM to set up the specific monitoring processes to enable any services and resources control (see Figure 4.3 ). Moreover, in order to make the formalization of semantic queries easier, a suitable graphical user interface based on Linked Open Graph has been used to access the KB and browse the semantic model [31]. An instance of the iCaro KB applied to the DISIT data center can be accessed by the Linked Open Graph tool on the real time RDF store of the iCaro cloud tools at http://log.disit.org.

## 4.4 Supervisor & Monitor

The SM is a cloud monitoring engine. It collects data from cloud resources, stores them for historical reasons, provides relevant data to the KB, and produces monitoring graphs and charts. For the low level monitoring, the SM specifically uses drivers to manage multiple Nagios instances (not discussed in this article). The SM collects monitored values from the cloud IaaS, PaaS and SaaS levels and high level metrics, HLM. The SLAs are typically based on HLM such as: the number of users registered on a social network, the number of downloads, the average number of connections, etc. All the collected data are stored in RRD (round-robin database) format. In this case,

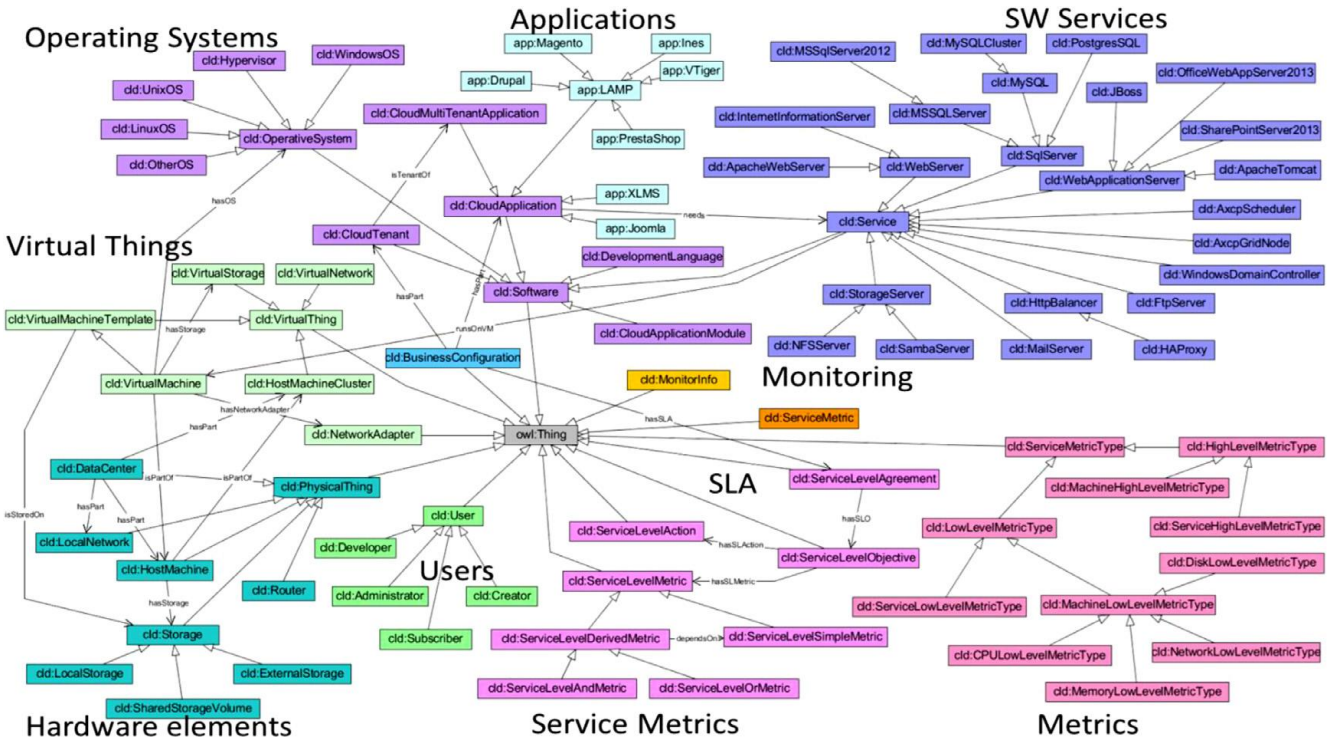



Figure 4.2: Ontology under Knowledge Base

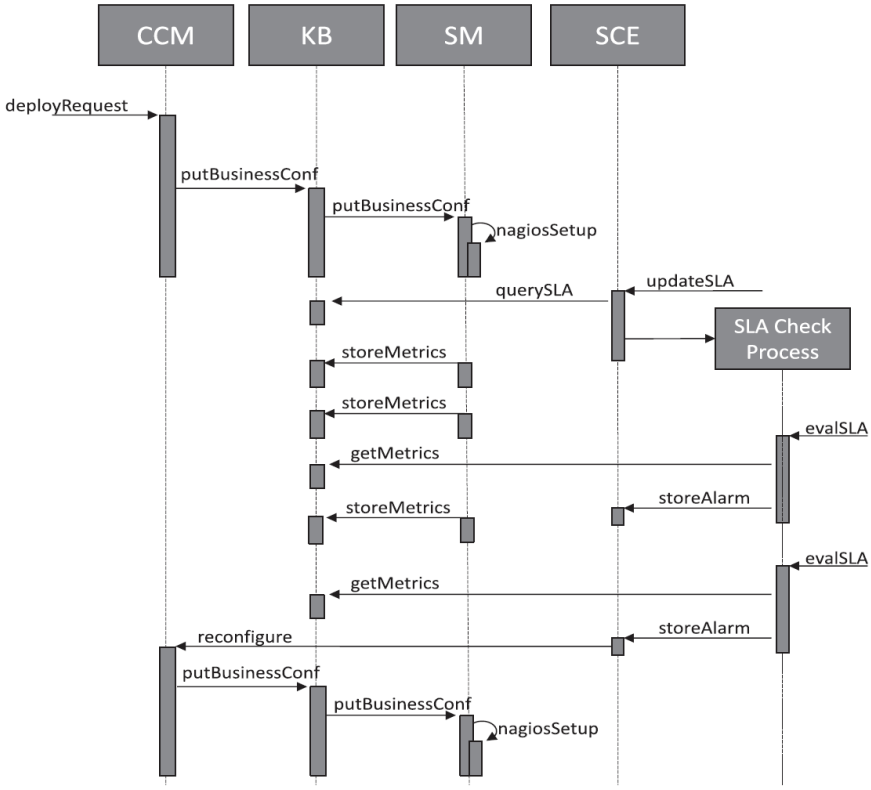



Figure 4.3: ICARO supervisor and monitor

Nagios has been chosen but a different low level monitoring tool could be used, too. The approach of delegating the monitoring processes configuration to the KB (see Figure 4.3 ) makes the work of the Orchestrator simpler, since each monitoring issue does not need to be programmed into the deploy workflow, thus reducing the error prone process, the distribution of passwords, etc. Besides, it allows to be sure that the SCE automatically adds all the monitoring processes allowing the SCE to have all the needed information to control the active BCs and SLAs. The SM is therefore automatically managed and configured by the KB. For all the collected data, the SM provides graphics and charts on demand to CCM (to be shown to customers), as well as to the user interface of the ICLOS.

## 4.5 Smart Cloud Engine

The SCE is an autonomous engine for the supervised control of cloud resources, for the automation and optimization of cloud services [29]. The SCE periodically checks the status of cloud resources in the cloud infrastructure (e.g., VM and application services) for each Business Configuration, BC, on the basis of the SLA. To this end, the SCE poses SPARQL queries to the KB modeling real or simulated clouds against additional rules with respect to those imposed in simulation. It can pose queries not only on the KB, but also on any other external database. The KB has the detailed model of the cloud since any new resource allocated on the cloud is registered into the KB by the cloud administration tools. The SCE executes a set of decision rules associated with cloud resources (e.g., Host, VM, services, switch, etc.) and SLA/BC. Each decision rule is typically composed by:

**An assessment condition** , when true it activates the actions. The assessment condition estimates the resource healthiness, verifies the contractual conditions of the SLA, etc. For example, if a BC is getting low in resources, according to the SLA a scale out strategy is planned.

**One or more actions** corresponding to the activation of strategies and procedures, for example for: scaling, reconfiguration, migration, cloning, balancing, etc. Actions can be configured to invoke remote calls (REST or WS or local calls) towards the CCM or the Orchestrator or other

Thus, thousands and thousands of SCE processes are executed per day, on a distributed scheduler. With the aim of detecting critical conditions

and taking decisions in real time, the SCE provides a distributed scheduler engine with cluster functionality allowing adding new scheduling nodes and defining jobs, for smart cloud management, without service downtime. The SCE can take decision about the cloud reconfiguration, address aspects of energy consumption, capacity planning, etc., with the aim of maintaining a high quality service according to the SLA, and to the general objectives of the cloud service provider in terms of energy, costs, etc. Thus the SCE can activate reconfigurations, in/out scaling, load balancing, moving, cloning, etc. The SCE presents a graphic user interface which includes: process definition and monitoring, decision configuration, connection to actions, etc.

# Chapter 5

# Icaro Cloud Simulator - ICLOS

This chapter discusses the functionality of Icaro Cloud Simulator. The first part describes the requirements that it must have to simulate scenarios that cannot be simulated with the other simulators already present in the literature. The internal and external architecture is then shown with the links between the simulator and other iCaro project tools, in particular the KB and the SM. Finally, it is described how workloads are realized starting from the patterns generated in Chapter 3 and the results of experiments carried out on the simulation and on the allocation of virtual machines are reported, as the latter grow.

## 5.1 Requirements

Real multimedia services, social networks, large web sites with Content Delivery Network (CDN), crowdsourcing solutions, and smart city solutions, typically they need to manage:

- Complex BCs as multitier architecture including several VMs, services, networks, services, processes;
- Real resource consumption patterns that may provide non-periodic behavior, as well as overlapped with periodic behavior at level of: hour, day, week, month and/or year. These factors can be due to the alternation of working hours, vacations, business orientation, seasonal commercial factors, and to possible unexpected events, like the arrival

of a storm, etc. The trends about resource consumption for CPU, memory, storage, network, etc. are related one another, and thus the real BC profiling has to be considered in terms of related patterns;

- Simulation for longer time windows by using workload partners describing days, weeks, months. Longer periods can be produce by replicating, while the modeling of long duration workload pattern strongly increase the simulation complexity;
- Simulation of multiple objectives, for example, the energy consumption on viable cloud allocations;
- Articulated SLA to avoid violation of SLA and to control major cost parameters, taking decision, informing the customer and administrators, etc., mainly connected to the Smart Cloud, SCE, features;
- Strategies activating elastic configuration processes for scaling on the front end, scaling on the database, scaling on the content ingestion of user generated content, scaling for computing suggestions, etc., also connected to the Smart Cloud, SCE, features;

For the most part such aspects are not addressed in a satisfactory manner by the simulators at the state of the art, see Chapter 2. The main goal of the ICLOS is to simulate the workload and cloud model in general and save them along days, weeks, months, etc. in the SM and KM. This allows to:

1. *model and simulate larger cloud and more complex configurations,*
2. *activate the SCE rules for further analysis.*

## 5.2 General Architecture

Figure 4.1 has presented the general architecture of the ICLOS. As depicted in Figure 5.1 the ICLOS consists of a number of subsystems. SM and the KB subsystems have been described respectively in Section 4.4 and in Section 4.3 with the aim of presenting their role for the general cloud management level.

The elements of the ICLOS solution are described as follows:

**Simulator GUI** is the user interface of the ICLOS to:

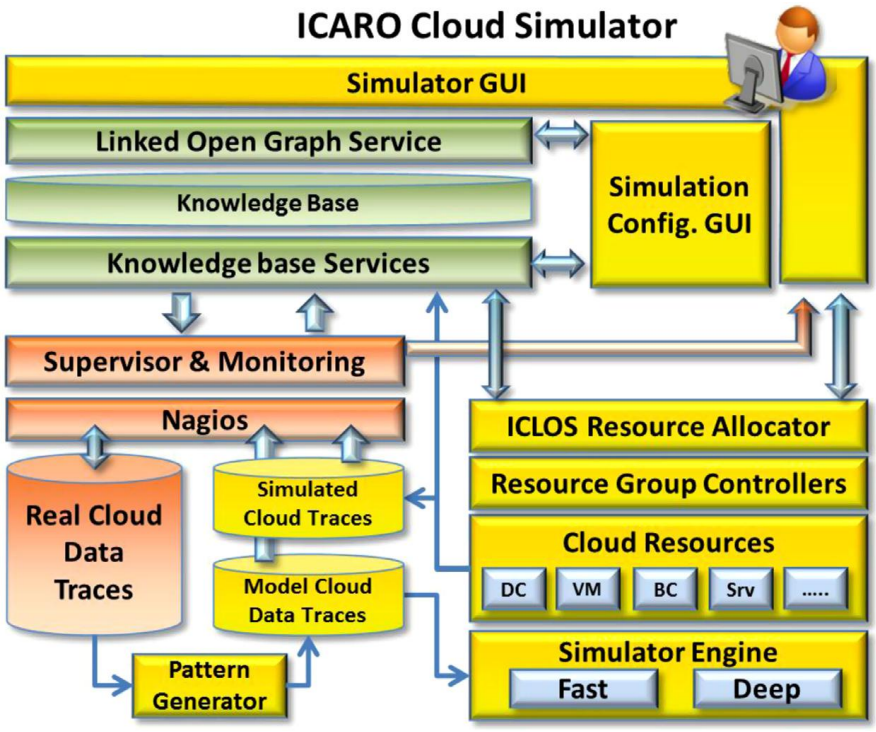



Figure 5.1: ICLOS architecture.

1. set up a new configuration to be simulated,
2. impose the configuration data,
3. obtain the simulation results in terms of resource consumption graphs and general assessment results.

**Simulation Configuration GUI** is a specific user interface to configure parameters of the resources involved into the configuration to be simulated. A configuration to be simulated is produced and stored into the KB by sending an XML file. The ICLOS starts from the KB to perform the simulation, and produces the result corresponding to the allocated resources into the Simulated Cloud Traces saved into RRD format.

**Pattern Generator** a set of tools to estimate patterns for resource work-

load, always considering the related CPU, memory, network, storage, etc. along days, hours, week, months, etc., of different VM, hosts and services of a BC (see Chapter 3).

**ICLOS Resource Allocator** on the basis of the configuration of resources it allows to allocate them into the cloud simulator memory.

**Resource Group Controller** it allows the management of the allocated resources addressing events and harmonizing the math models for computation.

**Cloud Resources** a collection of allocated resources according to the produced configuration. It may take into account multiple and incremental configurations. The resources that can be allocated in the simulator are in principle the ones being modeled by the KB (see Figure 4.2 ), while in reality only some of them are allocated and deployed as described hereafter.

**Simulator Engine** the simulation model can progress in estimating the output workload synchronously among all resources, time instant by time instant (deep mode), or it can compute the results on the basis of workload patterns associated to resources in the configuration phase and taken from the Model Cloud Data Traces in RRD format; thus, resulting in a faster simulation (Fast mode). The simulated values are the same requested by the simulator during the configuration and coherently defined by the SLA for each BC. The results of the simulation is again generated in the RRD format, thus allowing the visualization of results on SM and any further reuse in more complex simulations.

The ICLOS has been designed to model into the simulation the main KB classes and structures. In Figure 5.2, the main classes modeling layers IaaS, PaaS and SaaS aspects, the SLA and the group controllers are reported. According to the design pattern of Model View Control, a number of classes have been developed (not reported in Figure 5.2). They allow to view and model the inputting of data for each of the addressed cloud resources. On the other hand, their purpose is limited to the production of the XML file to feed the KB. The main goal of the ICLOS is to simulate the workload and cloud model in general and save them along days, weeks, months, etc. in the SM and KM. This allows to:

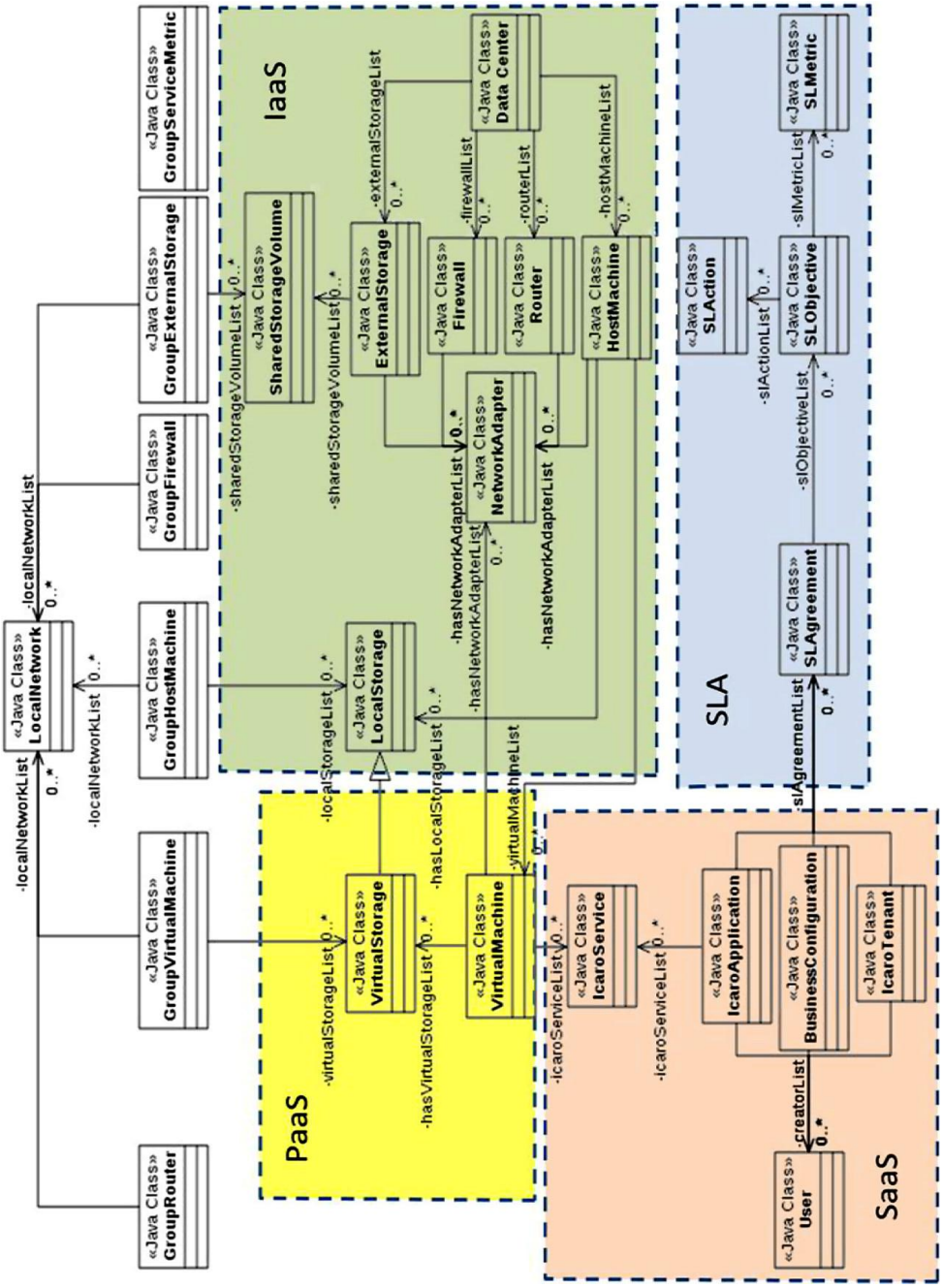



Figure 5.2: ICLOS architecture.

1. model and simulate larger cloud and more complex configurations,

2. activate the SCE rules for further analysis

## 5.3 Cloud workload

The problem of pattern production for cloud simulation has been addressed by Google Cloud Backend which performs a characterization according to their duration, CPU and memory requirements [67] The analysis of the data collected by the Performance Monitor may be used to perform a workload classification [96] [15]. Such workload patterns are exploited in cloud simulation in the ICLOS solution. In reality, the mere statistical characterization of VM or hosts on the basis of CPU and Memory workload is not enough to cope with complex BCs. The exploitation of SCE and Cloud Simulation based on real workload patterns derived from the monitoring log of the SM can be the path to setup a smarter cloud management engine [62].

The Pattern Generator (Chapter 3) perform a clustering analysis to identify the most probable workload patterns from real resource consumption trend of classical cloud resources and/or high level metrics such as: CPU, memory, network, storage, user activity, disk usage, etc. These trends can be computed per hour, day, week, month, etc., from real BCs including hosts, VM and services of a BC. The exploitation of SCE and Cloud Simulation based on real workload patterns derived from the monitoring log of the SM can be the path to setup a smarter cloud management engine [62].

The Pattern Generator tools exploit Real Cloud Data Traces in RRD format collected from Nagios/SM on the real cloud to perform cluster analysis and produce the most likely family of patterns for a given BC to be used into the ICLOS simulation phases. The family of patterns of each single BCs are coherently selected (associating coherent values among resources, avoiding of making simulations with CPU workload unrealistic with respect to the memory usage or disk access). Moreover, they are randomly selected among the most probable patterns (see Figure 5.3) to create the simulation workload. In addition, the same pattern is normalized and used to create different kinds of workloads, for example with 10%, 30%, 60%, 90% of load, and/or adding some random changes of limited value.

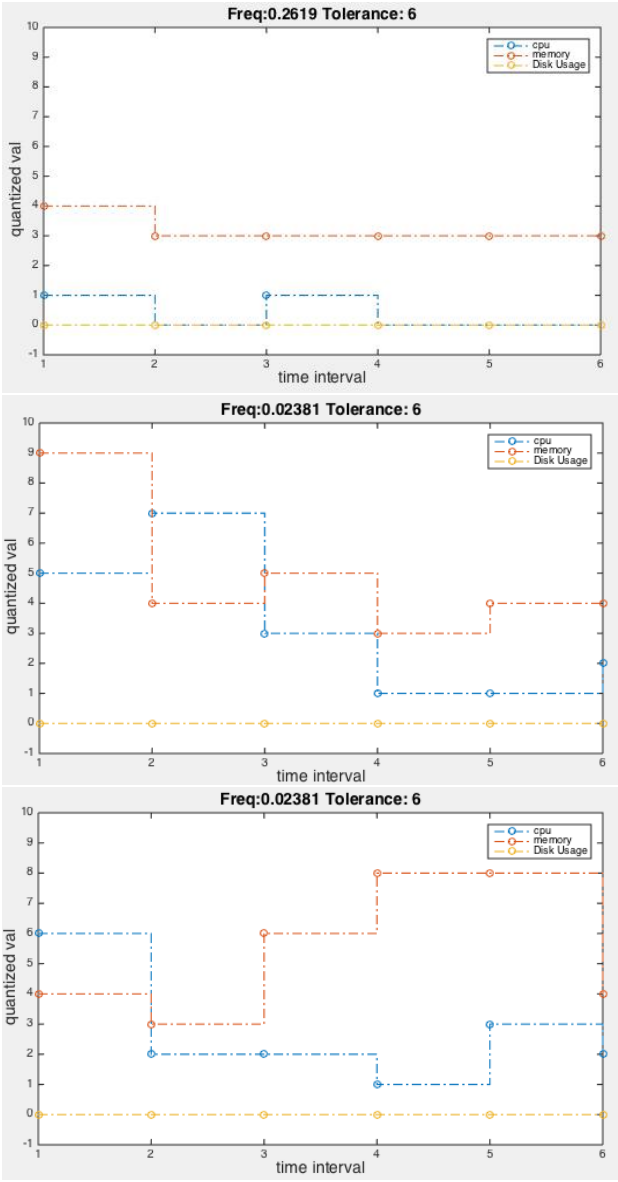



Figure 5.3: Example of three patterns created by Pattern Generator

## 5.4 Experimental results in simulating

In [119], the simulation of 1000 Hosts with 4 VM each, for a total of 40,000 VMs, the energy consumption model only, was performed in 3597s on an Intel Core i7 930 processor and 6GB of RAM. The estimation has been assessed by performing 5 repetitions and the simulations were done along 10 days, with a single data value every 10 min. The power consumption model has been modeled by using SPECpower benchmark [10]. For comparison purposes, a similar simulation has been performed with ICLOS. Thus, taking 1000 Hosts with 4 VM each, a total of 40,000 VMs has been simulated by computing the energy consumption model SPECpower benchmark [10] and using input values every 5 min, while generating output simulated values every 5 min. The simulation has been performed 5 times on Debian 64 bit, 6 Gbyte of memory, CPU 4 core, 2000Mhz, obtaining average time of 1985 s and a Std.Dev.=245.89. As a result, the ICLOS and DC simulators are comparable in terms of execution time. The simulation time cannot be easily compared with other simulators since in the case of ICLOS the simulations address longer time windows, and longer time lead also to spend more time in saving the output data resulting from the simulation of all the VM and Hosts on the hard-disk, with a sample every 5 min. Figure 5.4 reports the ICLOS simulation directly monitored into the SM tool which exploited NAGIOS libraries to access and render the RRD storages.

Moreover, Table 5.1 reports details of a number of simulations / configurations by considering: VM ranging from 1 to 3000, each of them with: CPU clocks per second equal to 2000 MHz, reserved CPU clocks per second equal to 800 MHz; RAM memory of 3 GB, reservation memory space of 1 GB; Hosts (cases 1 and 2) ranging from 1 to 10 (each of them with: 32 cores, 2500 MHz per core and 128 GB Ram); Hosts in cases 3 and 4 have been scaled up consequently. In ICLOS, the costs of Host computing simulation is included into the VM model, so that the simulation time and storage is linear with the number of VMs. The ICLOS simulations have been performed by using workload patterns of 1 week forward for resources (CPU, memory and disk) from the RRD of the SM with a measure every 5 min, thus simulating a whole week for the VM and hosts.

Therefore, the input workload patterns have a value every 5 min and they can be specifically assigned or randomly selected from a set of real patterns taken from ECLAP social network, Sii-Mobility smart city aggregator tools, etc. from the DISIT data center in XML format coming from RRD of

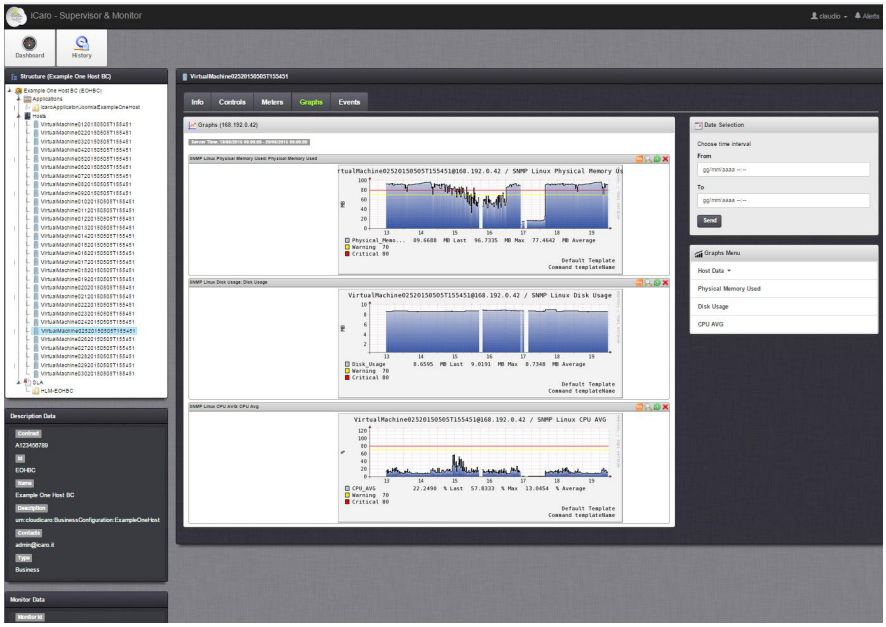



Figure 5.4: ICLOS simulation results on SM.

SM. Please note that the simulation of 1 week for 3000 VM/Hosts has been performed in about 80 min on a single server. The computing time can be allocated on multiple servers hosting the simulators, taking different segments of the cloud on KB to be simulated; since all computations are independent and produce results directly on the ICARO RRD/XML of the SM (the SM provides high level results to KB). Please note that, the registered numbers from simulations as reported in Table 5.1 have been obtained as mean value taken from 20 simulations with the same parameters. The simulations have been executed on a Debian 64 bit, 6 GB of memory, CPU 4 core, 2000 Mhz. ICARO Simulator has been developed in Java and runs on Tomcat. The Mean Total Time refers to the time needed to execute the whole simulation including the reading of the patterns (CPU, memory, storage) for the whole VM, the computation of the VM and Host load and any saving of the resulting data on SM in RRD format in a remote HD. The *Avg total time/#VM* grows marginally passing from 300 to 3000 VMs (at 1.60 s) with an increment of the 13% of the mean computational and saving cost per VM.

| **Parameters** | Case 1 | Case 2 | Case 3 | Case 4 |
|---|---|---|---|---|
| # Host | 1 | 10 | 1 | 1 |
| # VM per Host | 30 | 30 | 300 | 3000 |
| Total number of VM | 30 | 300 | 300 | 3000 |
| **Memory measures** | Case 1 | Case 2 | Case 3 | Case 4 |
| MB used for data output (.RRD) | 36.1 | 361.2 | 350.7 | 3503 |
| Avg MB used for data output | 1.20 | 1.20 | 1.17 | 1.17 |
| **Measured times and pc metrics** | Case 1 | Case 2 | Case 3 | Case 4 |
| Mean Total Time, in s | 38 | 385 | 422 | 4798 |
| Std Dev in Mean Total Time, in s | 1.32 | 16.85 | 19.12 | 228.47 |
| Avg total time/#VM | 1.25 | 1.28 | 1.40 | 1.60 |
| Mean Time Simulation, in s | 10 | 93 | 90 | 1061 |
| Avg Time Simulation/#VM | 0.330 | 0.311 | 0.301 | 0.298 |
| Mean Time Save RRD on SM, in s | 27 | 289 | 331 | 3723 |
| Avg Time Save RRD on SM, in s | 0.900 | 0.963 | 1.103 | 1.241 |

Table 5.1: ICLOS Simulations for power consumption assessment

This increment is mainly due to the cost of writing and sending the RRD of VM into the store of the SM (see Figure 5.4). The computational time to simulate the 10 Hosts with 30VMs for week (CPU, mem. and storage) is of about 93 s. On the other hand, the *Mean Total Time* reported in Table 5.1 also includes for each VM the access on HD to take the pattern, the XML parsing, the computation of simulation and the writing of the RRD/XML with the simulation. Provided that the simulation time is quite constant, it is almost useless to perform simulations with higher number of VM and Hosts, with a needed storage of about 1.2Mbyte of HD per each VM for a week. Each Host simulation is performed autonomously and thus also the RAM memory used by the simulator is almost constant, keeping its values under 120 Mbyte in all cases.

## 5.5 Experimental results in VMs allocation

The first case is reported in Table 5.2, where different BPAs have been used in order to identify the most probable number of hosts needed to allocate a number of VM (from 500 to 4000 including BCs).

| **Algorithms** | **Execution time (sec)** | **Host Number** | **Execution time (sec)** | **Host Number** |
|---|---|---|---|---|
| Number of VM | 494 | | 998 | |
| Dot Product | 49.36 | 37 | 169.89 | 75 |
| L2 Norm | 43.62 | 38 | 163.88 | 77 |
| FFD Sum | 2.95 | 37 | 14.84 | 75 |
| FFD Prod | 5.77 | 38 | 14.06 | 75 |
| Number of VM | 2000 | | 4000 | |
| Dot Product | 708.64 | 150 | 3169.60 | 300 |
| L2 Norm | 739.93 | 153 | 3457.49 | 305 |
| FFD Sum | 59.09 | 150 | 266.38 | 300 |
| FFD Prod | 47.96 | 150 | 231.36 | 300 |

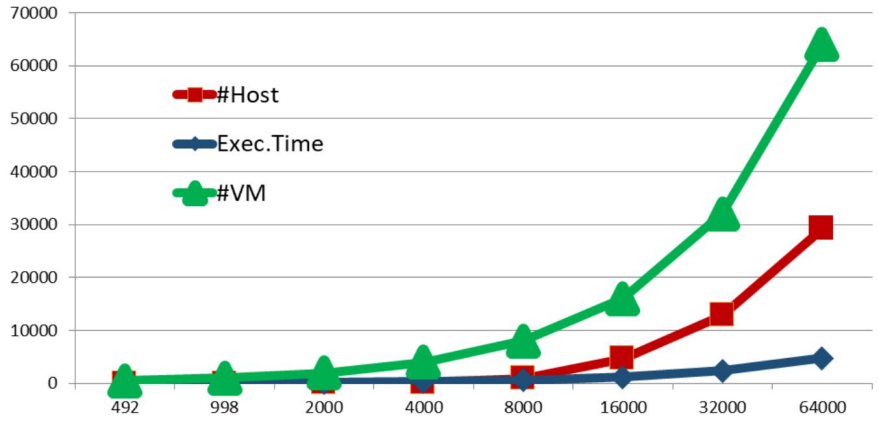
Table 5.2: ICLOS simulations for allocation by using different algorithms. The execution time refers to 20 executions of the allocation algorithm in simulation. The *host number* refers to the most probable number of hosts identified among the set of 20 simulations; in most cases, this number is the minimum number of hosts according to the goals of the adopted Bin Packing Algorithm (BPA).

By combining the patterns available in different reports, it was possible to develop mixed test cases to simulate more complex BCs. To maintain consistent BCs it is necessary to assign patterns to VMs respecting the modularity of applications: if a test case consists of four machines, so the number of VMs that can be used to simulate the allocation on a data center will necessarily have to be a multiple of four. If this were not so, test cases would be used that do not actually correspond to real applications but only to parts of them.

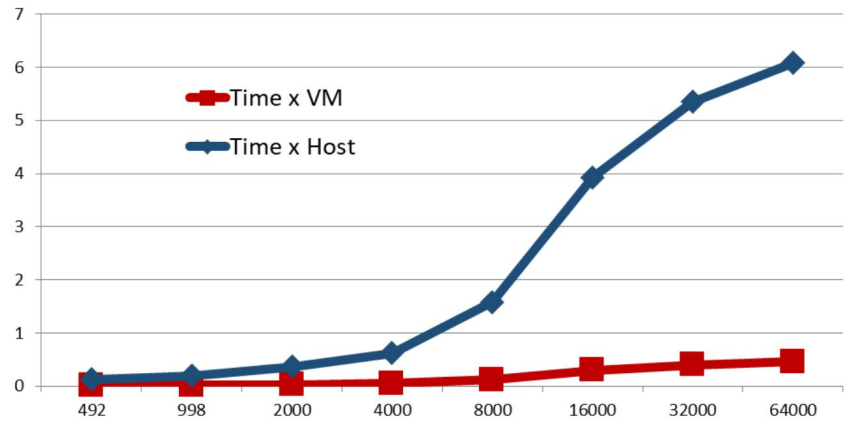
The BPAs try to compose the VM while respecting the possible configurations and composing the resources patterns, so as to always keep in mind the limits of the host capacity (Section 3.3). The algorithms selected have been already adopted for cloud resource allocation [101], and in particular the **FFD** by sum and by product weight, the **Dot product**, and the **L2 Norm** (For details see Section 2.2 of this work). When the patterns are complex, the Bin Packing goal is to find the compromise from the most probable number of minimum host for allocating a number of VMs belonging to a set of different BCs. The simulations have been performed using generated patterns from real cases and simulating one working day. As to the obtained data, it can be remarked that the FFD Prod algorithm provides good results with shorter execution time in almost all cases. The execution times have been estimated on a 24 CPU core host at 3.0 GHz with 64 GB of RAM on 20 simulations.

On the other hand, in most cases, 10 simulations could be enough to estimate the configurations to obtain the most probable number of needed hosts, as reported in Table 5.2. The patterns where referred to a distribution of 4 different BCs: a 4 tier architecture for data warehouse (5 VM as a balancer, 2 web server, one database and 6 computational nodes); a three tier solution of small social network, a simple two tier solution for a web server application, and a single tier solution with a web application. The simulations have been addressed starting from their real workload patterns and producing clusters simulation patterns for a 1 whole day of 24 h.

A second simulation experiment shows the execution time for VM allocation from 500 to 64000 VMs. The simulations have been addressed with the above described BC workload patterns for a day. Figure 5.5(a) reports the trends of the execution time in seconds, with respect to the number of VMs and the identified most probable number of hosts. Figure 5.5 (b) describes the execution time in seconds for simulating a VM and a Host respectively.




(a)




(b)

Figure 5.5: Simulations for allocating VMs by using FFD Prod algorithm from 500 to 64000 VMs: (a) trend of execution time and number of VMs and Hosts (20 simulations), (b) estimation of the executing time cost with respect to the number of VMs and Hosts.

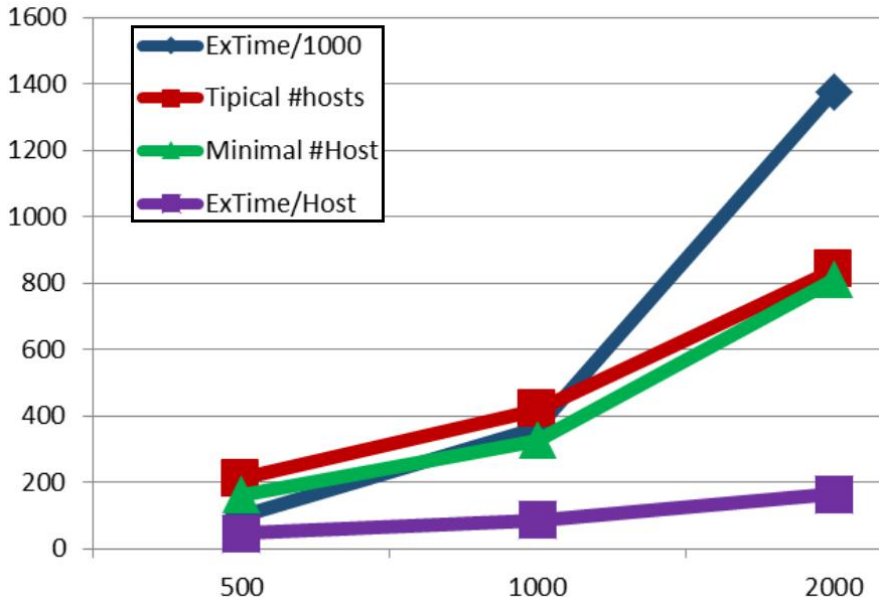



Figure 5.6: Simulations for allocating VM by using FFD Prod algorithm from 500 to 2000 VMs, workload patterns for 1 month according to the described mix of BC, 10 simulations for each estimation

In both cases, simulations with more than 32000 VMs tend to stabilize the execution time per VM and per Host.

In Figure 5.6, the trends of simulation execution time are reported for the case of workload of 1 month, for the same complex BC described in the first case of this section. The simulations have been addressed with complex workload patterns of 1 month according to the mixt of BCs described above. A huge complexity is added when long time durations are taken into account. In fact, the case of 2000 VMs for a day produced results of packing all into 150 hosts, while in this case of 1 month, the packing leads to 839 hosts, thus taking into account critical longer period behaviors for the VMs. In terms of execution time for the 2000 VMs, for a day costs 47 s (see Table 5.2), while for the case of 1 month pattern the averaged execution time for 20 simulations 2753.6 s that leads to about 91.8 s per day per simulation.

# Part II

# Part Two

# Chapter 6

# Smart City

In this chapter a brief overview is given on the Smart City concept, highlighting the challenges related to this idea and how these problems have been solved with the development of the Km4city framework for the aggregation of static and dynamic data and the creation of the Sii-Mobility architecture to make these data easily usable by those who are interested in having them, be they citizens or SMEs.

## 6.1 Concept

Open data as static data are not the main source of information in the city, neither the most valuable for the city users (citizens, tourists, commuters, operators, students, etc.). Most of the big data problems connected to smart city platforms are related to real time data as the public transports, vehicle and human mobility in city, events, parking, weather, wind, etc. A smart city architecture should be capable to take advantage of huge amount of big data coming from several domains, at different velocity for exploiting and analyzing them for computing integrated and multidomain information, making predictions, detecting anomalies for early warning and for producing suggestions and recommendations to city users and operators [23].

In the last years, many architectural solutions have been proposed with the aim of making data accessible, aggregated, usable, and exploitable, etc. [17], [55], [51], [47], and many of them failed in posing the basis for creating a smart city open environment for new and smart applications.

The major companies are proposing solutions to make city smarter, focusing on specific set of domains, such as IBM [4], [5] on services for citizens, business, transport, communication, water and energy; [89] on governmental, educational, e-health, safety, energy, transport and utilities; CISCO on people, things and data [1], etc. Most of these solutions present a multi-tier architecture ranging from 3 to 6 layers [17].

On the other hand, the number of tiers is partially relevant to the transformation of data in value for business, and thus to services for the city users, and in opportunities for the enterprises and city operators interested in creating innovative and effective services, while exploiting city data and information [55], [51], [47]. Also, the smart city ranking models are not suitable in putting in evidence these aspects, since they are mainly focused on counting the number of provided open datasets, smart services, solutions, or energy results [109], [70] [19].

In most cases, the effectiveness of a data service system for Smart City is enabled by the availability of private data owned and managed by City Operators addressing specific domains: mobility operator, energy providers, business services (health, water), telecom operators, tourist operators, university, etc. They are the city stakeholders providing data and services with different granularities and size. For example, in the city, we can have few energy operators with capillary house distribution, many public transport operators with thousands of vehicles/busses, some telecom operators deploying in the city from tens to hundred thousands or millions of sensors [22].

The data values (actual, predicted and/or detected) can be delivered to different operators and city users by some personal assistants on the basis of the user profile and role. For example, in order to provide information about what is or what would be around a current GPS position, the integration of geographic information and services is needed; while the integration of geolocalized services and the assessment of typical people flows may help the city in improving public services and transport, providing suggestions to the city users, and planning changes in the city [44]. For example, in [125] a solution for estimating the crowd density has been proposed exploiting mobile phones and Bluetooth; while in [95] a solution for monitoring people and vehicles in the city by exploiting multiple data sources has been proposed. In [91], [66], solutions based on Bluetooth server have been proposed for estimating the travel time. Thus, aggregated data can be exploited to implement a large number of services and applications by structuring the

Smart City Architecture and the corresponding Smart City APIs [19].

Despite of the above described large offer of different kinds and corresponding effort on creating smart city architecture and solutions, nothing has been done on making simple the creation of mobile and web Applications. The smart city developers, typically SME, researchers, students, and operators, still have to develop their applications by studying in deep the Smart City API, recovering the data models, reconciliating and aggregating data (to be repeated at each changing of the data model), creating applications exploiting low level Web Service and/or REST Call without the support of development tools for Apps [89], [4], [17], [2].

On the other hand, the world of mobile and web Apps is changing, the Apps are becoming more and more dynamic, pushing on HTML5 and on instant App, as the Android Instant Apps to run them without installation, and thus to use them at run time. This approach will create the need to a continuous renovation of Apps and the reduction of fidelized users. In conclusion, the production of web and mobile App has to be faster and cost effective [23].

In this research work, an innovative tool for smart city web and mobile Apps development is generated (see Chapter 7): it contains a set of open source Apps for shortening the development (e.g., starting from scratch on new kind of Apps, as well as developing modules that can be loaded dynamically from a an already published App) [79].

The proposed development tool is based on Smart City API described in [22] which in turn are based on Km4City ontology [26] and RDF storage. The proposed development tools have been developed realized in the context of Sii-Mobility Smart City national project on mobility and transport of Italian Ministry of Industry and Research, and presently also used as development tool and model in REPLICATE H2020, and RESOLUTE H2020 projects of the European Commission.

## 6.2 Knowledge Model 4 City - Km4City

Km4City provides a unique point of access for interoperable data of a city metropolitan area via web and mobile applications. Km4City provides a set of scalable and efficient tools fro data ingestion, management, aggregation, indexing and for producing in short time web and mobile applications have been realized and make accessible. Km4City is a comprehensive and open

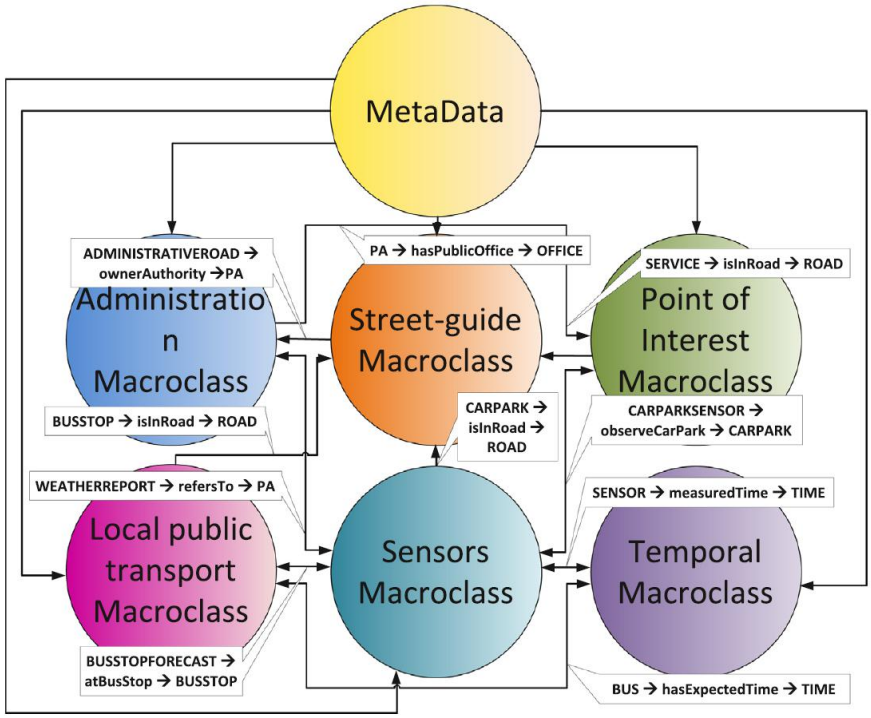



Figure 6.1: Ontology macro-classes and their connections

ontology for smart cities covering domains of: weather, cultural heritage, smart sensors, public structures, mobility, city parking, services, transportation, events, geographic locations, health, etc. [20].

In order to create a knowledge model for Smart City services, a large number of data sets have been analyzed to see in detail each single data elements of each single data set with the aim of modeling and establishing the needed relationships among elements, thus making a general data set semantically interoperable at model level (e.g., associating the street names with toponimous coding, resolving ambiguities) [26].

The analysis of the above mentioned data sets allowed us to create the km4City integrated ontological model presenting 7 main areas of macroclasses as depicted in Figure 6.1, and described as follows.

**Administration** includes classes related to the structuring of the general

public administrations, namely PA, and its specifications, Municipality, Province and Region; also includes the class Resolution, which represents the ordinance resolutions issued by each administration that may change the traffic stream.

**Street-guide** formed by entities as Road, Node, RoadElement, AdministrativeRoad, Milestone, StreetNumber, RoadLink, Junction, Entry, and EntryRule Maneuver, it is used to represent the entire road system of Tuscany, including the permitted maneuvers and the rules of access to the RTZ. The street model is very complex since it may model from single streets to areas, different kinds of crosses and superhighways, etc. In this case, Ontology for Transport Network (OTN) vocabulary has been exploited to model traffic [7] that is more or less a direct encoding of Geographic Data Files (GDF) in OWL.

**Point of interest (POI)** includes all services, activities, which may be useful to the citizen and who may have the need to *search-for* and to *arrive-at*. The classification of individual services and activities is based on main and secondary categories planned at regional level. In addition, this macrosegment of the ontology may take advantage of reusing Good Relation model of the commercial offers: in fact, the ontological model Km4City allows connecting Service instances to the corresponding instances of Location belonging to GoodRelations model [68].

**Local public transport** includes the data related to major LPT (Local Public Transport, in Italian: TPL, Transport Public Local) companies scheduled times, the rail graph, and data relating to real time passage at bus stops. Therefore, this macroclass is formed by classes PublicTransportLine, Ride, Route, AVMRecord, RouteSection, BusStop-Foreast, Lot, BusStop, RouteLink, RouteJunction. (where AVM means Automatic Vehicle Monitoring).

**Sensors** macroclass concerns data from sensors: ambient, weather, traffic flow, pollution, etc. Currently, data collected by various sensors installed along some streets of Florence and surrounding areas, and those relating to free places in the main car parks of the region, have been integrated in the ontology. Some of the sensors can be located on moving vehicles such as those on busses, car sharing, bike sharing, and on citizens' mobiles, etc.

**Temporal** macroclass that puts concepts related to time (time intervals and instants) into the ontology, so that associate a timeline to the events recorded and is possible to make forecasts. It takes advantage from time ontologies such as OWL-time [100].

**Metadata** This group of entities represents the collection of metadata associated with the data sets, and their status conditions. If they have been ingested and integrated into the RDF store index, data of ingestion and update, licenses information, versioning, etc. In the case of problems with a certain set of triples or attributes, it is possible to recover the data sets that have generated them, when and how.

## 6.3 Sii-Mobility Architecture

The reference architecture of Sii-Mobility is depicted in Figure 6.2. The solutions allows to collect data coming from different kind of sources (open data, private data, real time data), domains (mobility, environment, energy, culture, e-health, weather, etc.), and protocols. The architecture is based on a semantic aggregation of data and services according to the Km4City ontological model. Data providers as City Operators and Data Brokers offer data which are collected by the Smart city in pull by using Extract Transform and Load (ETL) processes scheduled on the Big Data processing back office based on a Distributed Smart City Engine Scheduler (DISCES) tool developed for Sii-Mobility and made open source. Among the data collected those provided in Open Data from the municipalities, Tuscany region (Observatory of mobility), LAMMA weather agency, ARPAT environmental agency, etc., and several private data coming from City Operators: mobility, energy, health, cultural heritage, services, tourism, wine and food services, education, wellness, etc. Data Brokers collect and manage real time data coming from sensors (IoT), and from vehicular kits (On board Device) which are developed for monitoring and informing car, bus and bike drivers, etc.

Once the data are collected the back office perform several processes for improving data quality, re-conciliating data and converting data into triples for the RDF store of the KB [27], implemented by using a Virtuoso triple store. DISCES is allocating processes on several virtual machines allocated on the cloud according to their schedule and requests arriving from the Decision Makers, Developers and Data Analytics (typically 3.5-5 thousand of jobs per day, collecting multiple data per job, for example all the busses on a

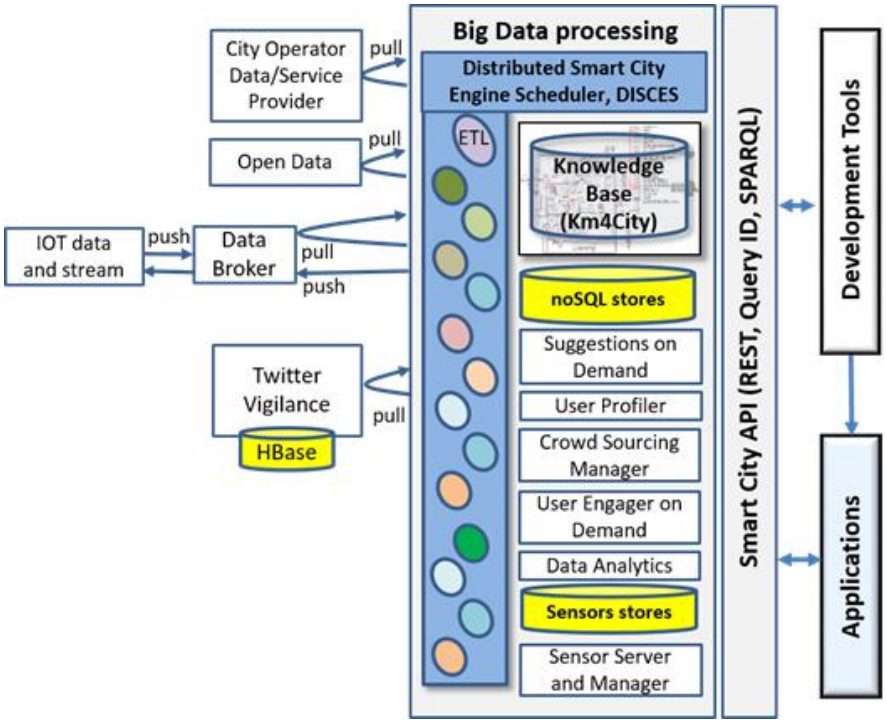



Figure 6.2: Sii-Mobility Architecture.

line according to DATEX II protocol [3]). The processes for data collection can be scheduled according to several different policies to cope with Open Data (to verify if they change sporadically), quasi real time data (changing a few times per day) to real time data (changing every few seconds, such as the position of the Bus, or the position of the City Users) and taking into account all the permissions access connected to each different piece of information managed in the Km4City Knowledge base.

For semantic aggregation of data and service it has been decided to exploit and improve the Km4City Ontology (http://www.km4city.org ) [26], [27], adding a number of details regarding mobility and transport, sensors, environment, with respect to former model. Now Km4City is modeling multiple domain aspects related to mobility, services, Wi-Fi, cultural services, energy, structure (streets, civic numbers, green areas, sensors, busses, etc.).

The above data collected is exploited by a number of scheduled data analytics processes to compute: user behavior and mobility, recommendations, suggestions and personal assistant messages according to the city and city operator strategies.

In order to be capable of providing contextual information web and mobile Apps provide data to the Sensor Server and Manager. The data collected from Apps (mainly mobiles) are related to many different aspects: the position of the city users, preferences (user profiles), requests to the Smart City API, searching queries, action performed on mobile, velocity, accelerations, etc. [80]. All these kinds of data are useful to understand the user behavior, and thus, to engage the users generating ad-hoc suggestions and recommendations.

In the architecture proposed, in addition to the RDF store for the knowledge base, presents several noSQL stores (namely: HBase and Mongo) for storing tabular data as those arriving from sensors and user profiles, and to make versioning of collected data that have to be passed into the RDF store for reasoning. This approach allows to have the needed tabular data accessible for Data Analytics processes such as those performed for the: estimations of recommendations, engagements, traffic flow predictions, parking forecast, clustering of sensor data behavior, and anomaly detection. When needed, federated queries can be performed among RDF and tabular stores. The resulted architecture provided several services via Smart City API to Development Tools or to the City Users Tools (Applications).

# Chapter 7

# Mobile Application Developer Kit

This chapter describes the Mobile Application Developer Kit in all its basic features. This Kit has been developed to simplify the creation of Mobile Applications using the Sii-Mobility platform and the Km4city framework described in Chapter 6. In particular, it explains in detail the operation of the service that allows to provide useful tips to users for experiencing their city at its best. On two applications developed with this kit, *Florence where, what.... Km4city* and *Tuscany where, what.... Km4city*, some usability reports have been made in the last part of the chapter.

## 7.1 Architecture

The main purpose of the Km4City Architecture is to enrich and aggregate the data, thanks to the KM4City Semantic Model, and then make the data available for other purposes, depending on the permission access of each different kind of data.

The work for a developer, who want to access and reuse the Km4City aggregated and qualified data is complex. Thanks to the geo-referenced info of each data stored in the Km4City knowledge base, a set of services and tools based on a visual approach has been realized, starting from those useful for the developers. These tools, have been developed in addition to

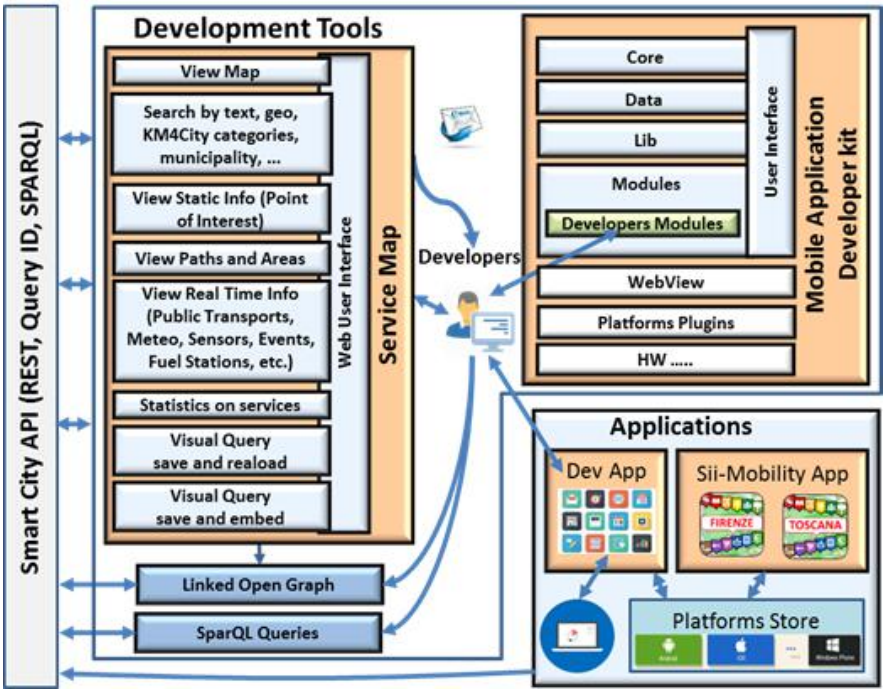



Figure 7.1: ServiceMap Development Tools.

the API system to simplify the work to the developers, with an easy visual connections on data, to simplify the use of the Smart City API, to give a set of guidelines to develop new web and mobile applications. The main developers' tools created to solve these kinds of problems are:

- ServiceMap [9]
- Mobile Application Developer Kit [79]

The set of Development Tools also includes other assisting tools to work with RDF stores and SPARQL queries, when and if needed.

In Figure 7.1, the interaction among the main developers' tools is depicted. The actions performed by the developers to create their application are:

1. Make searches on the ServiceMap and visualize the resulting data on the Map.

2. Save the search done and receive an e-mail in which they can find the API to be used in their App to reproduce the same experience had in the ServiceMap (the same data).

3. Use the API to develop their own apps, in any forms (i.e., web app or mobile app, hybrid or not) or use the API and the Mobile Application Development Kit to create new modules that can be integrated on the already published Mobile Apps.

The main aim of the Application Developer Kit (ADK) consists in making simple and fast the production of new applications. This is possible thanks to the ServiceMap development tool and to the proposed architecture of the web and mobile applications. The main requirements that have drown our design have been the strong need of:

**Modularity** to have a structure which allows the modularization of functionalities in order to make the code development more simple, and distributed among several teams.

**Dynamicity** to make possible the addition of modules inside the application at run time (when the application is installed in the device of the final users), in order to speed up the deploy of:

- the new functionality in the hands of the final users,
- the integration of the new functionalities in the core part of the application.

**Personalization** to adapt the user experience (menu, user interface, functionalities) on the basis of the final users actions and profile; also giving to the user the possibility of resetting and changing the profile.

**Alerting** to enable the App to receive alerts, notifications and messages possibly when the App is in background or off, so that to inform the user about critical situations in the city and also for personal assistance, suggestions, etc.

**Multiplatform** to realize Apps that can be installed and performed on more than one platform, avoiding substantial changes of the code.

The proposed architecture (see Figure 7.2) of the ADK and thus of the final App have been designed to enable every developer to create his own module, that can be dynamically loaded inside the application.

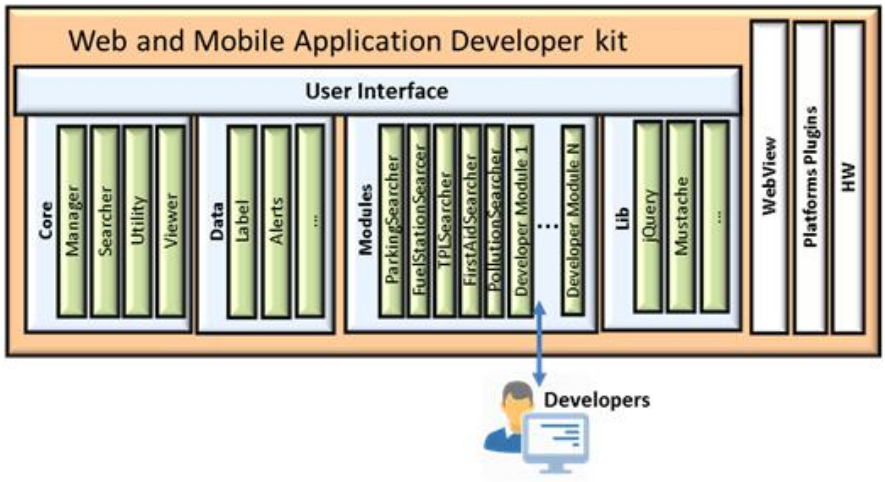



Figure 7.2: Web and Mobile Application developer kit.

## 7.2 Modularity and Dynamicity

The modular structure of ADK (see Figure 7.2) is used to prevent the change of files inside the application with another code or data, from new developers. New modules have to be placed inside the modules folder of the ADK, and the ADK has to be loaded at the proper time scripts and templates. New data have to be also combined with data already present in the basic version of the ADK (i.e., Labels and Alerts in Data block). The modularity structure avoids developers to modify the other parts of the ADK; they can use the functionalities made available for the modules and submitted from other scripts (i.e., take data from GPS position, visualize services on the map and other functionalities offered by scripts contained in Core block). As shown in Figure 7.2, Modules Block contains modules (already developed and integrated on Sii-Mobility application) concern the services that offer data in real time:

**Parking Searcher** it finds the closer and the freest car parks in the city and shows the number of free parks.

**Fuel Station Searcher** it finds the closest and cheapest Fuel Stations and shows the favorite fuel kind and the price.

**TPL Searcher** it finds the closer public transportations stops and shows the next ride time, or all the list;

**First Aid Searcher** it finds the hospitals and shows the emergency room situation, the triage status;

**Pollution Searcher** it finds the closest environmental and pollution sensors and shows the last data collected by the sensor(s).

Each module is substantially a mini-application, in a certain sense, since each of them exploits the general access to the Smart City API, the registration of the user ID, the access to sensors of the device, etc., and thus may receive from the services on servers (see Figure 6.2). The produced modules may be dynamically integrated inside the official application or provided into the App from the store. In the first case, every time the App is executed search for the availability of new modules to load them from the server site, also updating the former versions when needed. The loader directly locates the new and updated function and miniapps on the menu, and when needed new buttons are connected to the new feature that will appear and in the desired place according to the server manifest of the module.

## 7.3 Personalization and Profiling

Users of the Apps are profiled and their profile is communicated to the server. Profile are presently kept totally anonymous and classified as citizens, students, commuters or tourists. Thus their interaction with the interface is anonymously recorded, to know the most sought services of the different profiles and modify the main menu. This allows us to tune the service and improve the user experience providing different menu arrangement of the functionalities according to the user profile. Thus the menus offered for each profile is modified according to the statistics calculated on the collected data, by defining the positions the most researched categories, buttons, etc., and in some cases, putting off/on some functionalities. For example, the triage monitoring may be of interest for citizens and operators and less for tourists, the parking status is interesting for user using the private car and less for those moving with the public busses or in bike, etc. The priority on the creation of the menu profiles is given in first place to the updated menu which can be found on the server. If the server is unreachable, the last saved

menu for that specific profile is loaded; if the menu has not been saved, the loaded menu is the menu released with the application update, and so on.

## 7.4 Alerting and Tracking

The users on the App need to be informed on what is going on the city. On the other hand, most of the Mobile platforms have different approaches for providing asynchronous notifications in push to the device when the App is not in foreground, is not executed by the user. On this regard, Android allows to execute processes in background so that it can be used to collect changes with a polling approach, iOS provide a service called APN for central management of notifications in push and does not allow to execute processes in background, a similar approach is also provided for Windows Phone. In addition to the alerting, the movements of the device should be also communicated to the server in order to get new context based suggestions and alerting, such as: please take care about the weather forecast in your area, alarms of civil protection, environmental status, closer car park, etc. Some of these innovative and smart features are produced as suggestions, engagements from personal assistant [21], predictions produced by the data analytics modules (e.g., parking, arrival of busses, etc.), alerts (e.g., civil protections, changes in the traffic, events).

### 7.4.1 Tracking Service

The mobile application is deployed across multiple operating systems and different data collection strategies are performed, based on the possibility of running services in the background inside the various devices. Currently, the data collection service works as a background service that is always active for those who install the application on a device with Android operating system. In other operating systems, no dedicated background service has been developed and data is collected during use of the foreground application. Both services (foreground and background) follow the same workflow of operations (see Figure 7.3), although they are formally different being written in two different languages: Java for the background service of Android and Javascript for other operating systems.

The main collected data are those related to:

**positions and movements** through raw information on latitude, longi-

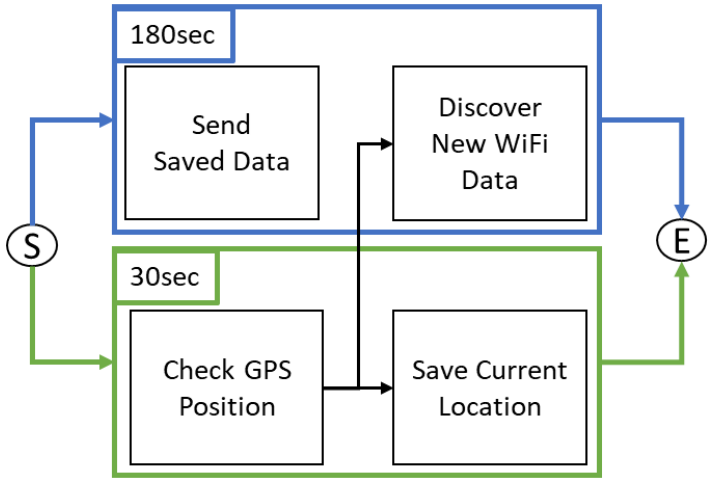



Figure 7.3: High level workflow of operations performed by tracker

tude, speed, altitude and accuracy of these measurements provided by the device's GPS (or other location provider) [21].

**terminal charateristics** as operating system, application version, device mobile

**user charateristics** as language and profile

Other data collected are related to the disposition, in particular on the centre of Florence, of public WiFi in order to have an overview of the correct distribution of these free Wi-Fi access points [30].

The data collection protocol mainly revolves around the possibility of retrieving the GPS position on a given device at a given time (for a detailed view see Figure 7.4).

The protocol checks whether a location provider is active on the device: preferring the GPS over others (the location ca be obtained from the network or using a mixed strategy). If no provider is active, it means that the user has disabled location detection so only a *status* record is retrieved. This record indicates that the service is still running but it is not possible to do anything else on that device.

If at least one of the two providers is active, there may be two more different situations:

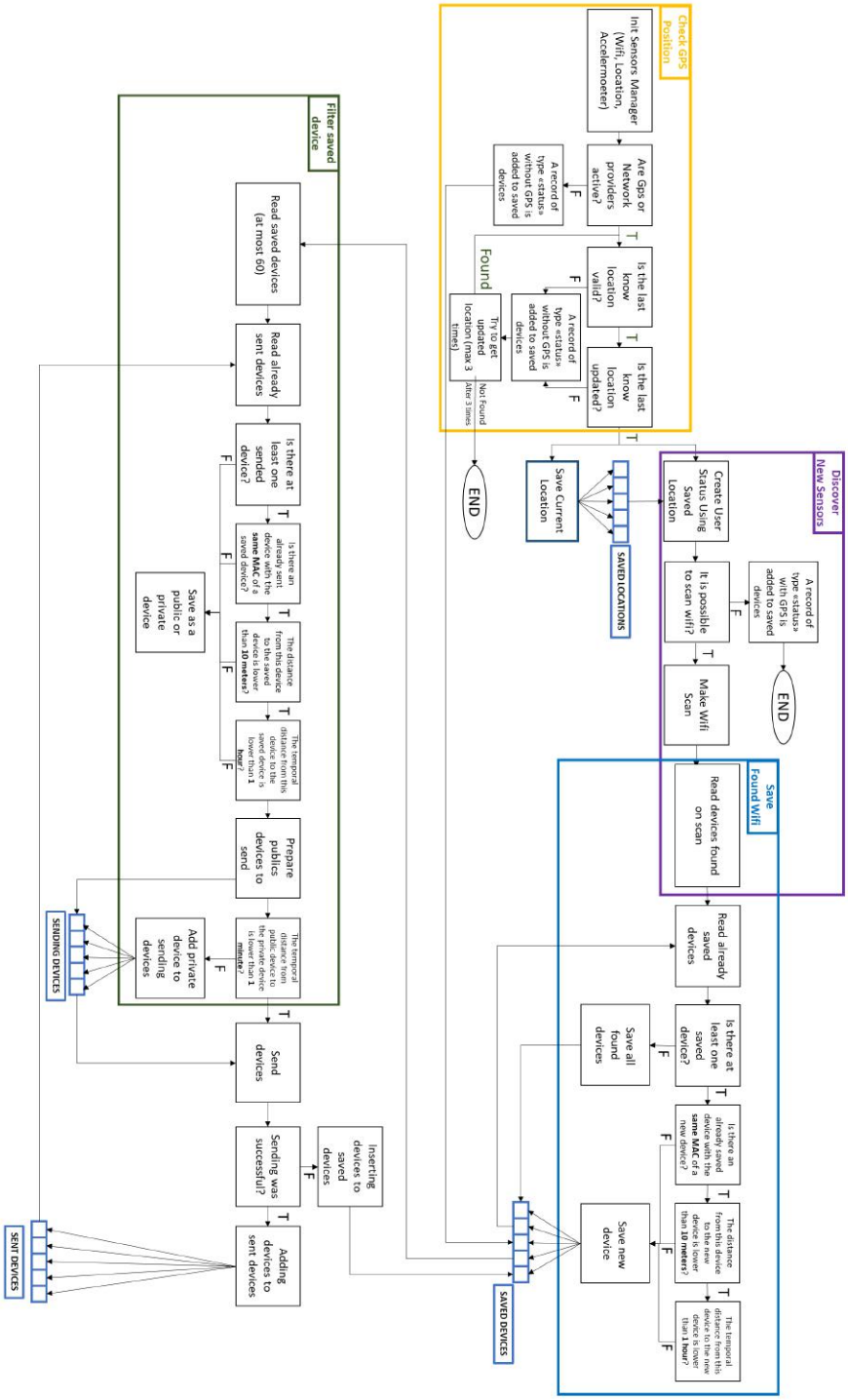



Figure 7.4: Detailed workflow of operations performed by tracker

**Not Available** this provider is not able to detect the location of the device at that time (for example, the user is inside a building where the GPS signal fails to retrieve information from satellites)

**Not Updated** the location detected is not up to date (the device has retrieved the position but then entered inside a building: the position is still the last detected but it has passed too long)

In both cases, a position update is requested from the service to the active provider at that time: if the update is not possible, a *status* record is retrieved.

Assuming that an updated location can be retrieved, it is possible to scan Wi-Fi and beacons in the proximity of the device and send them to the server.

If the airplane mode is active no other operation is performed and a *status* record is sent (different from others this record contains location information). If the airplane mode is not active, then scanning can be performed: it checks whether the Wi-Fi module of the device can be activated and if it is possible it is activated and scanning is carried out to collect data near the device. If the Wi-Fi module has been deactivated, the device will return to its initial state in order not to interfere with the user's choices. Once the list of Wi-Fi devices in the closeness of the device has been obtained, 3 filters are performed in sequence:

1. data from Wi-Fi, having the same MAC as those already saved, are deleted if is temporally and spatially close to those already saved.

2. removes as much as possible the private Wi-Fi detected during the scan: public and free Wi-Fi (with SSID FirenzeWifi and UnifiWifi) data are immediately saved . If the data of public Wi-Fi do not cover a wide spectrum of space and time, then private Wi-Fi are also kept in memory, as they are able to make up for the "holes" in the recovery of information that would have happened with public Wi-Fi.

3. the remaining data are compared with the data already sent to the server and also in this case for data with the same MAC are deleted those spatially and temporally close to others already sent.

Once the data to be sent has been saved, they are sent periodically by the service in the background: if the sending is successful, the data sent will

be deleted from those saved otherwise they will be queued again and sent in the next period.

If the transmission is successful, a successful data entry message is returned. Eventually, this message as a payload containing a message or more messages generated by the Engager for the user [21]. These messages are saved in temporary files located in the folders where the application can be accessed and once the application is started, they are read and shown to the user as soon as possible.

In Figure 7.3, the protocol's operations performed by the background service are shown. The block called *Save Current Location*, every time the GPS position is available, stores a list of data and, whenever the data are sent to the server, these positions are used to understand how the user is moving at a given time [30].

Both background service and foreground service perform the same operations. It is important to notice that the foreground service is more precise in sending data, since when the application is open, it is possible to make synchronously all the calls for preparing and sending available data. For the background scenario, different policies has been adopted by various Android's versions, both with regard to the official versions and those developed by device manufacturers. Some of them trying to limit the use of resources (CPU, memory and network) by the services that run in the background and increase the period of clock execution of such services up to 5 minutes from theoretically 30 seconds shown in Figure 7.3.

## 7.5 Multiplatform

In order to satisfy the first requirement of multiplatform, the Application Developer Kit and the architecture have been based on Apache Cordova framework. It allows us to realize hybrid applications on multiple platforms. The applications developed with Apache Cordova, shell consist of user-interface implemented with HTML, CSS and Javascript, and of plugins which allow to use the specific hardware functionalities from the different platforms (i.e., battery-status, camera, device orientation) (Platforms Plugins in Figure 7.2), through a Javascript interface. The Sii-Mobility App development has shown the powerful of this framework that permit in a very short time (i.e., few days or one week) to publish the application on different platforms stores, and to have a unique code for all platforms. It is possible

to start from the ADK source code released as AGPL, available on GitHub Disit Lab, for all developers that would like to realize new modules for official applications on the basis of data available through Smart City API Km4City and/or from other sources and APIs.

## 7.6 Usability test

This section discusses the results of the Sii-Mobility App usability assessment day organised on Wednesday 1 March 2017 by the Center for Generative Communication (CfGC) and Disit Lab of the University of Florence.

### 7.6.1 Objective

The aim of the usability tests was to:

- check whether the sample of testers was able to easily use the different functions offered by the App;
- analyse the perception that users have of this application, in order to identify its strengths and critical points;
- highlight the main critical issues that need to be resolved.

Among the identified users, subjects who already used mobile apps and owned an Android device (smartphone or tablet) were selected. They were also asked to download the App a few days before the test, in order to start experimenting with its use.

The methodology used in this specific test involved a series of actions for users to identify the strengths and critical points of the mobile application [98]. Users had to take 12 actions. As far as the test is concerned, each action has been designed to introduce the various functionalities of the App to the testers, giving researchers the opportunity to analyze the level of understanding and usability of the available functions and the criticality found in each of them.

### 7.6.2 Organization

Users have to fill a *start questionnaire*, to evaluate the perception of the App downloaded in the previous days. Then they carry out a test based on 12 actions. Finally they carry out an *outgoing questionnaire* to allow

researchers to acquire further elements of knowledge and verify if, after the use of some functions in the test, their perception of the App has changed compared to the initial phase.

The test sheet distributed to users after the completion of the questionnaire was conceived by the CfGC as a succession of 12 pages with:

- a detailed description of the 12 actions to be carried out (1 action per page);
- a series of choices to be made with an x to indicate the degree of difficulty of the action to be taken;
- a space to describe the experience of solving the action;
- a space to record the main critical issues.

The team of researchers provided specific timeframes for each action, derived from simulations carried out previously: in the time available, users had to carry out the action, write down on the test card the steps taken and the critical points that emerged. Researchers, in turn, noted on an analysis sheet what they observed during the user experience. All test phases lasted about 2 hours in total.

### 7.6.3 Sample

The Sii-Mobility App usability evaluation day, which took place in Florence on Wednesday 1 March 2017 by Center for Generative Communication (CfGC) and DISIT Lab of University of Florence, was attended by 21 people. The 21 users who took part in the test were distributed as follows:

- 10 citizens,
- 6 students,
- 3 tourist,
- 2 commuters.

The sample of users was equally divided by gender (11 males and 10 females) as well as each individual category. As far as average age is concerned, however, the categories were divided as follows:

- citizens: 38 years
- students: 26 years
- tourists: 26 years
- commuters: 27 years

### 7.6.4 Test Descripton

The user test included 12 actions, with variable timeframes for each action, depending on the difficulty of each one. Each action was related to a specific functionality of the App. The list of actions is as follows:

1. **General Settings** Open the App and watch the main screen in front of you and the different features. Go to "Settings" and bring the "maximum distance" to 500 meters.
2. **Cycle Paths** Find cycle paths near you.
3. **Distributors** Find out how much gasoline costs in the stations around you.
4. **Nearby Point Of Interest** Try to find out if there are interesting places to see around you. After selecting "Library of Technological Sciences" take a photo, give a vote and leave a comment. Add this place to your favorites.
5. **Weather** Try to discover the next Friday's weather.
6. **Public Transport** Starting from where you are now, try to get to Santa Maria Novella station by public transport via line 4. Find out when the next bus will be passing and which route it will take.
7. **Ticket** Try buying a bus ticket to reach Santa Maria Novella station.
8. **Parking** Search for the nearest car park where you are now. Check if there are any free places.
9. **Events** Try to find out if there are exhibitions or other events around you. Select what you think you are closest to you and look at what it is about. After writing the event name on the usability test sheet, try to read the message in full screen. Share the event via WhatsApp (or another communication channel) with a user.

10. **Civil Protection** Look at whether there are civil protection alerts.

11. **Far Points of Interest** Find out what are the interesting places around Piazza della Signoria or in another area other than the one where you are now.

12. **Personalization** Try to customize the main page of the App according to the features that are most useful for you. Describe on the test sheet what features you deleted and why.

### 7.6.5 Test Results

In most cases, actions have been resolved by all users in the time available (see Figure 7.5). In 9 out of 12 actions there are users who have not solved what they were asked.

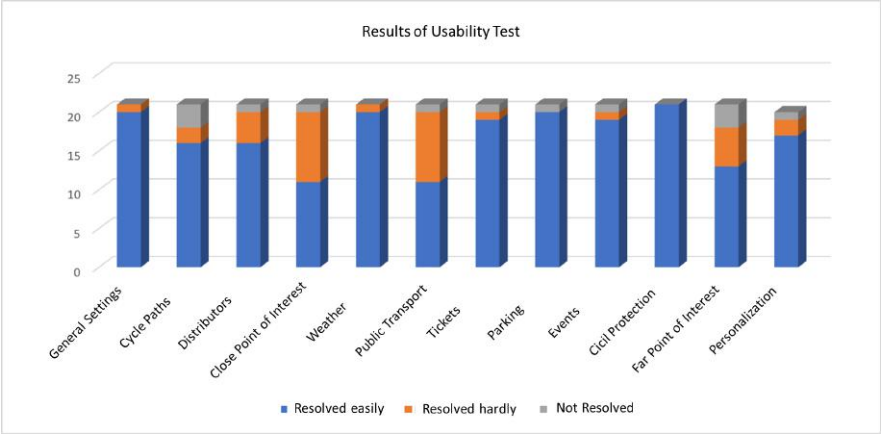



Figure 7.5: Usability Test Results.

However, only in two cases (action *2 - Cycle Paths* and action *11 - Far Points of Interest*) there are three users who have not been able to complete the action, while in 7 actions there is only one user who has not solved the proposed tasks. The most successful actions resolved with success are, in order, the *10 - Civil Protection*, resolved without difficulty by all users; action *5 - Weather* and action *1 - General Settings*, resolved without difficulty by 20 users and by a user with difficulties; action *8 - Parking*, resolved without

difficulty by 20 users and not solved by a user; action *7 - Ticket* and action *9 - Events*, resolved by solutions.

The most problematic actions were the *2 - Cycle paths* and *3 - Distributors*, solved without difficulty by 16 users, with difficulties of 2 and 4 users respectively and not solved by 3 and 1 users. Always critical at first glance is the action *11 - Far Points of Interest*, which has been solved without difficulty by only 13 users, with difficulties by 5 users and 3 users have not been able to solve it.

Finally, the most critical cases are represented by actions *4 - Nearby Points of Interest* and *6 - Public transport*, both resolved without difficulty by 11 users, with difficulties for 9 users and not solved by 1 user.

### 7.6.6 Use of App

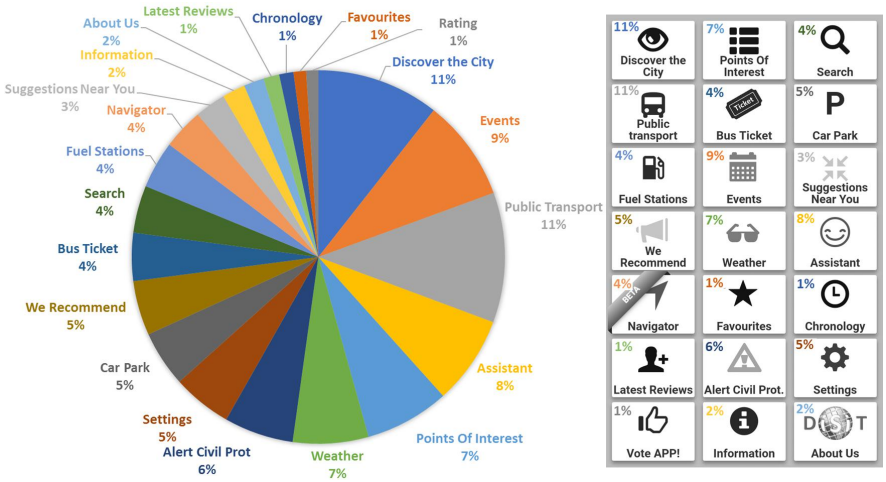



Figure 7.6: Statistics on clicks of the main menu.

In Figure 7.6, a statistic on users' clicks on the buttons of the main menu is shown. The percentages of clicks have been inserted on the buttons to see the spatial distribution: the clicks are concentrated mainly at the top left because the users search the mainly functionality on this area. Secondly, the clicks are focused on the central buttons and on the alert button: on these buttons during the use of the application are added badges with number that indicates how many elements are found once the user open that feature (i.e.,

the number of events planned for that day, the number of messages sent by the assistant, etc.). Then the notifications appearing on buttons provokes curiosity about users who are inclined to click to see what shows them.

In Figure 7.7, a statistic on the categories requested by users is depicted. The most searched categories are those relating to buttons that display directly to user the services sought (buttons on the principal menu or buttons aside of the map, when it is shown). Very popular is also the function Around you that is shown to the user in a popup over the GPS markers and over manual position marker. Search for categories, represented in the graph by the labels "Tourist Menu" and "Citizen Menu" is not much used even though allows the user a more targeted choice of services that should be searched.

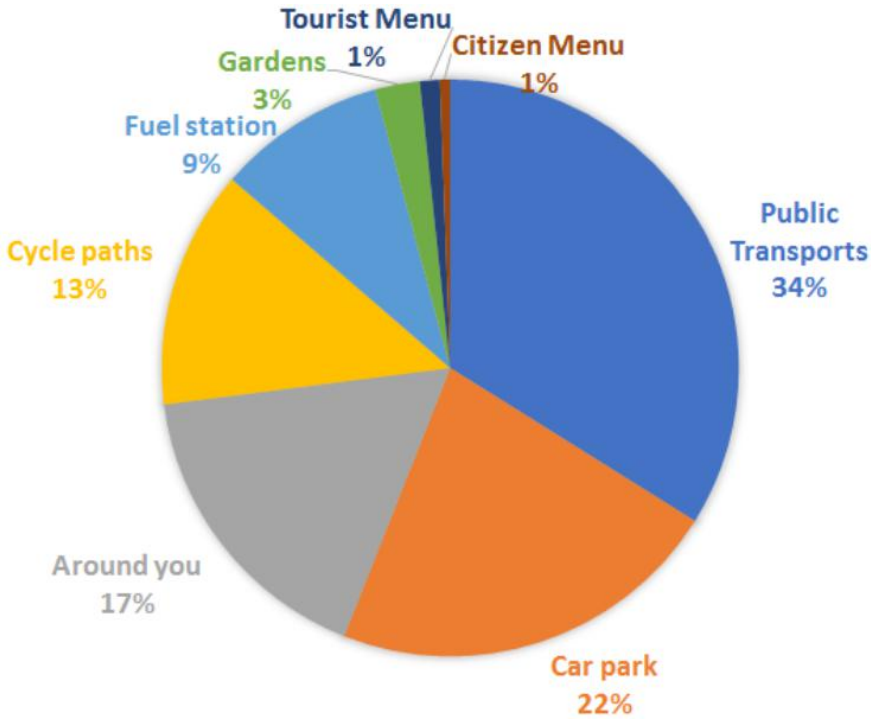



Figure 7.7: Statistics on more researched categories.

# Chapter 8

# Predicting Free Parking for ParkingSearcher Module

In this chapter, a set of metrics has been identified in order to predict the number of free parking slots in city garages with gates. With this aim, three different predictive techniques have been considered, compared and different models produced. The comparison has been performed on the basis of data collected in a dozen of garages in Florence area. The resulting solution demonstrated that a Bayesian Regularized Neural Network exploiting historical data, weather condition and traffic flow data can be a robust approach for reliable and fast prediction of free slots. The solution is deployed as a feature on Smart City Apps in the Florence area for sustainable mobility.

## 8.1 Introduction

Prediction of free parking spaces is a complex non-linear process whose dynamic changes involving multiple kinds of factors. In addition, to cover the whole Tuscany means to address the computation for about 200 garages. Those parking facilities provide several different working conditions. Some of them are dedicated to a specific facility (football stadium, hospital), others on multipurpose (station, fair, etc.), and others on periphery. Variability and performance are one of the problems to be addressed, together with the precision in critical time slots when the parking is getting full.

Drivers are wasting a considerable amount of time trying to find a vacant parking lot, especially during the peak hours, depending on the specific urban areas. Car drivers in dense city quarters usually spend between 3.5 to 14 minutes in search a slot [110]. Consequently, cruising for free parking spaces may depend from a peculiar number of different situations: different travel motives, the garage proximity from the destination, the price differences between garages, the driver's lack of familiarity with the urban area, etc. Looking for parking slots may not only be the reason of annoyance and frustration, it is in fact believed to have a harmful impact on the efficiency of the transportation system within the urban tissue, thus on sustainability.

Thanks to the today's technologies, it is possible to collect real-time parking information – i.e., capacity, garage prices, number of empty parking slots in the silos or in the area. Recent researches have highlighted the relevance of other data sources such as the garage proximity traffic flow information and the information related to the weather conditions. The prediction model proposed has been created by exploiting open data and real time data by using the Sii-Mobility (see Chapter 6) solution based on Km4City in the Florence area, Italy, for its corresponding Smart City solution.

In the context of monitoring and predicting the parking garage status, on Sii-Mobility more than 200 garages are monitored in whole Tuscany (an area of 3.5 inhabitants), and among them, about 12 are in Florence city. The status of each of those garages is updated every 15 minutes, while the goal consists in providing in advance, of 30 minutes and 1 hour, guesses about the parking status for each parking garage and provide such data within the App to make them usable by users in real time. In this manner, the car drivers will have time enough to decide to park in different parking area and/or to abandon the idea of taking the private car to reach the same destination by using a more sustainable solution such as the public transportation.

## 8.2 State of the Art

The parking activity of a driver is influenced by multiple factors, i.e. the walking distance to destination, driving and waiting time, parking fees, service level, parking size, safety [16], [81], parking price, availability and accessibility [103]. In particular, if it is known, the number of free parking spaces is an important attribute in the parking decision-making process of a driver, and its past experience on that. So that, drivers that possess information of

parking availabilities are 45% more successful in their decisions than those without knowledge about this information when arriving at their parking facilities [38].

In more details, parking slots can be searched on the street or in some parking silos of garage with gate. In terms of prediction models, there is a substantial difference between parking garages and street-parking. In parking garages, it is possible to count the number of free slots considering the tickets released to drivers at the entrance gate. In street-parking, could be necessary the set up systems to detect the occupancy with the use of sensors. For this reason some authors focused their research on the street-parking prediction, while others focused on the free parking slots prediction inside garages.

[118] have developed a theoretical model for parking demand forecast based on the capacity of network in central commercial district attempting to integrate the requests of both silos and street parking.

[46] tackles the street-parking problem in San Francisco by predicting the occupancy rate (defined as the number of occupied parking spots in the total spots) of parking lots in a zone given a future time and geolocation. He works on aggregated parking lots to explore the estimation error reduction pattern in occupancy prediction and to investigate different travel behavior at different region.

[46] discretizes the days into 24 intervals, and performed the principal component analysis on time series to discover occupancy feature. Thus, four different predictive approaches (ARIMA, Linear Regression, Support Vector Regression, and Feed Forward Neural Network) have been tried to investigate the relationships among prediction errors and aggregation levels. The comparison shown that feed forward neural network was the best model for prediction, presenting a prediction error 1 hour ahead of about 3.57%. In this case, only well-defined and stationary cases have been addressed using historical data and no additional contextual data. On the same line, [117] proposed an unsupervised clustering (Neural-Gas Network [93]), of the data to identify the similar street-parking behavior over 24 hours using a small data sample with a temporal resolution of 15 minutes. This approach for street parking prediction highlighted the strong variability without presenting a prediction model.

[74] have improved the solution [134] method (based on wavelet neural network) to predict the availability of a parking lot minute, in an interval

time of 15 hours (from 6:00 AM to 10:00 PM), using a three-days training set and one day as test set. The predicting capability obtained has been in the range of 3-10%. While suddenly declare that in critical cases (where the slots are close to zero) the error rapidly increases. We would stress that is precisely in those cases that the precision has to be high.

[124] have proposed a two-step methodology for the street-parking occupancy prediction based on sensor data. The first step consists in a real-time occupancy prediction scheme based on recurrent artificial neural networks. The second module is a static approach based on survival analysis for estimating the probability of finding available parking space with relation to traffic volume, the type of the day and the time period. The resulting error for predicting at 30 minutes is the range of 4.3%.

[42] have proposed a mathematical model based on queueing theory and Markov chain, to predict parking slots occupancy based on information exchanged among vehicles, which are connected to an ad-hoc network, presenting errors in the range of 8% after 30 minutes (1800s). In the same thematic area, [115], [128] have developed an "intelligent" parking system. In particular, [115] proposed a parking space inventory control system based on a combination of fuzzy logic and integer programming techniques making "on line" decisions whether to accept or reject a new driver's request for parking.

On the other hand, addressing the prediction of free slots in parking garages/silos is a completely different problem. In this case, the number of offered slots is typically higher in density, clearly reported at the entrance gate of the garage, and thus they are a strong attraction for drivers that may arrive all together. Moreover, they are typically located closer to commercial centers, hospital, railways stations, theaters, and multipurpose areas, etc. Therefore, the prediction of free parking slots in garages is not an easy task. Some of them may have stable stationary behavior over time (since serve stable facilities, such as peripheral hospitals), thus making their prediction easier. Others are affected by several factors that make the prediction of free slots over time much more difficult, especially in the critical situation in which the parking becomes full.

Considering that the data coming from Sii-Mobility are relative to the free slots in the garages/silos, more attention has been paid to those articles dealing with predictions within the garages.

In [102], a distinction between different sources of data, i.e., parking data,

user data, publicly data, has been proposed. They emphasize the importance of publicly data to make an independent and sustainable system to support the search of parking spaces on the street using a neural network model. According to [129], and [102], the capacity of garage depends on road traffic flow, weather events, road condition, etc., and the best prediction of available spaces is a combination of short time and historical information. In [102], the prediction capability derived by using neural network presented an MSE (mean square error) of 16% without addressing the critical situations of parking silos with non-stationary attitudes. [129] also use a neural network model to predict the available spaces in the city of Beijing but the prediction results are not well reported. The authors keep saying that their prediction results shown that when sample size is larger error is smaller, and that large sample size costs much time and affects real-time requirement, high lighting problems of performance.

## 8.3 Data Description

The data used for the prediction of the number of free parking slots in garages have been collected by the Sii-Mobility infrastructure, in the period from January 5, 2017 to March 26, 2017. For each car park (they are garages or silos controlled by a gate), the number of free slots has been captured every 15 minutes with their time stamp. The considered garages are located in three different areas of Florence: close to hospitals, in downtown (near to the main touristic area) and in peripheral areas, as exchange parking for who decides to leave their cars and to switch to public transportation.

As stated in the state of the art section, in terms of distribution of free spaces in the parking area, there is a substantial difference between a parking garage and a street-parking. In the context of street-parking, it could be necessary to make a clustering in order to understand the free space distribution of an area, to reduce the number of possible predictive models, aggregating the street-parking areas with the same behavior. Instead, taking into account garages, the distribution between each of them is very different, and the parking spaces are already concentrated in a specific area. For this reason a clustering is not viable: each garage can be considered as a different case, and may have a peculiar set of variables to obtain the needed precision.

Instead, taking into account garages, the distribution between each of them is very different, and the parking spaces are already concentrated in a

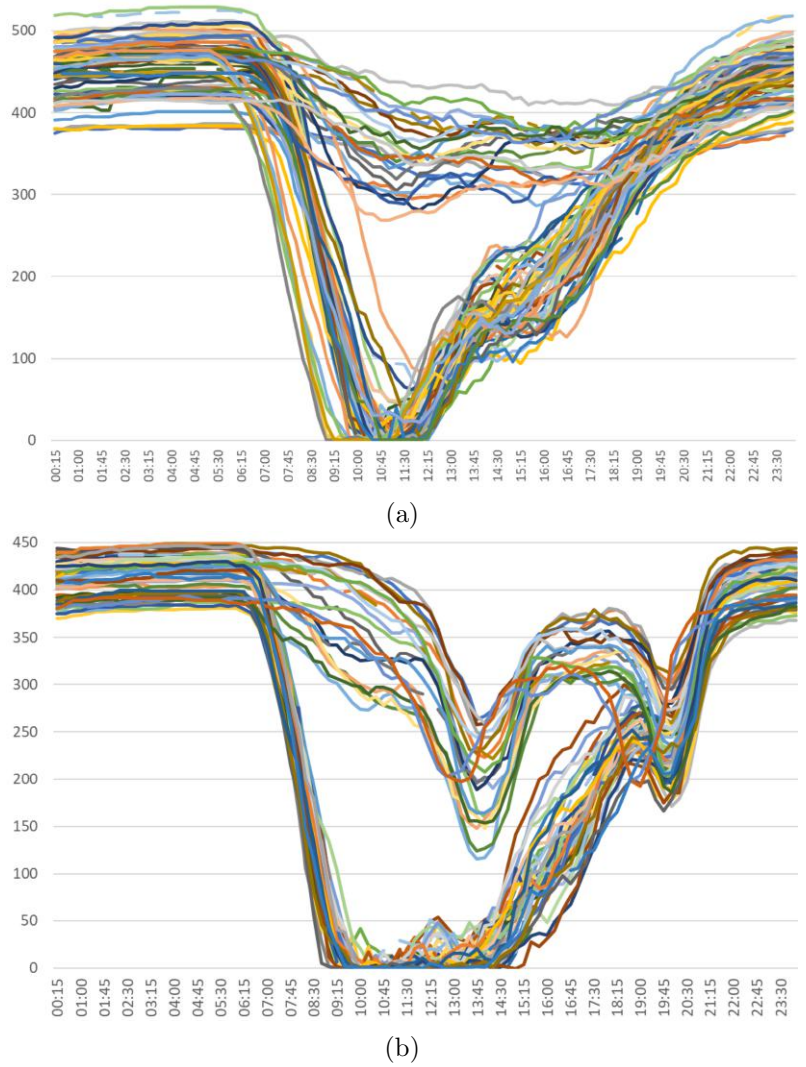
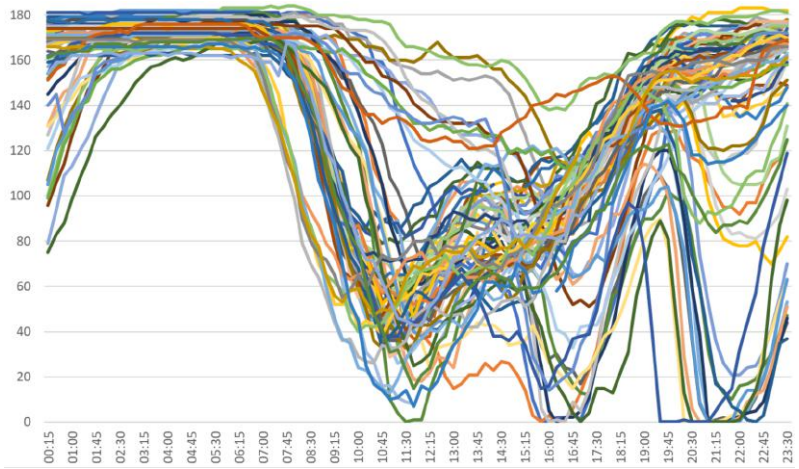


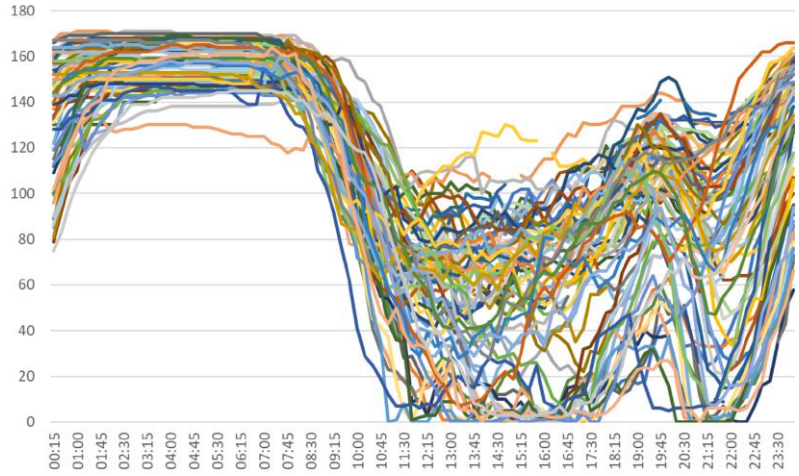

Figure 8.1: Typical daily trends of free slots every 15 minutes in parks (a) Pieraccini Meyer and (b) Careggi.

specific area.

Thus, one of the difficulties consists in identifying a common model that can be capable to produce suitable and precise predictions in almost all cases,




(a)




(b)

Figure 8.2: Typical daily trends of free slots every 15 minutes in parks (a) Beccaria and (b) S.Lorenzo.

and in particular, when the free parking slots are close to zero, that is the main critical condition, in which the drivers have to be alarmed in advance.

In Figures 8.1 and 8.2 and 8.3, a number of typical daily trends of free slots for some parking garages are reported.

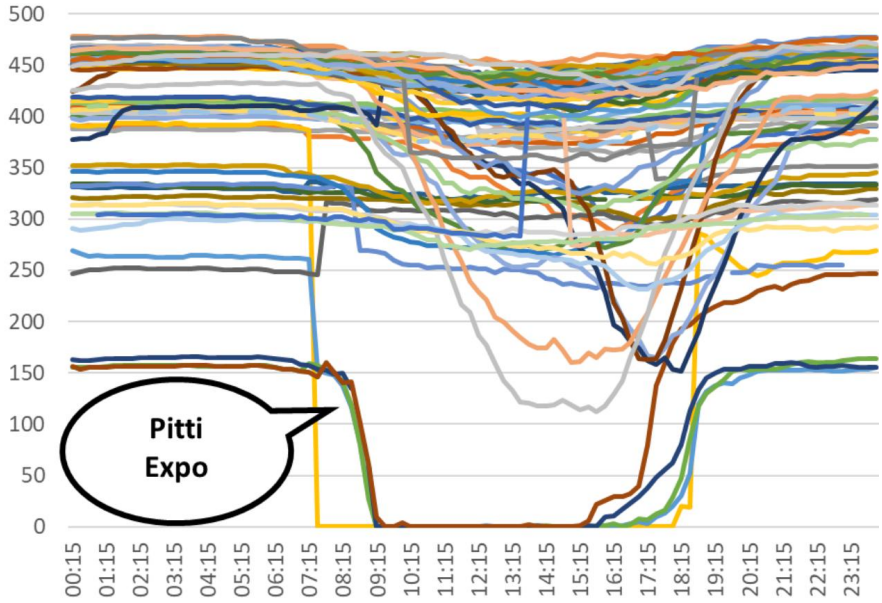
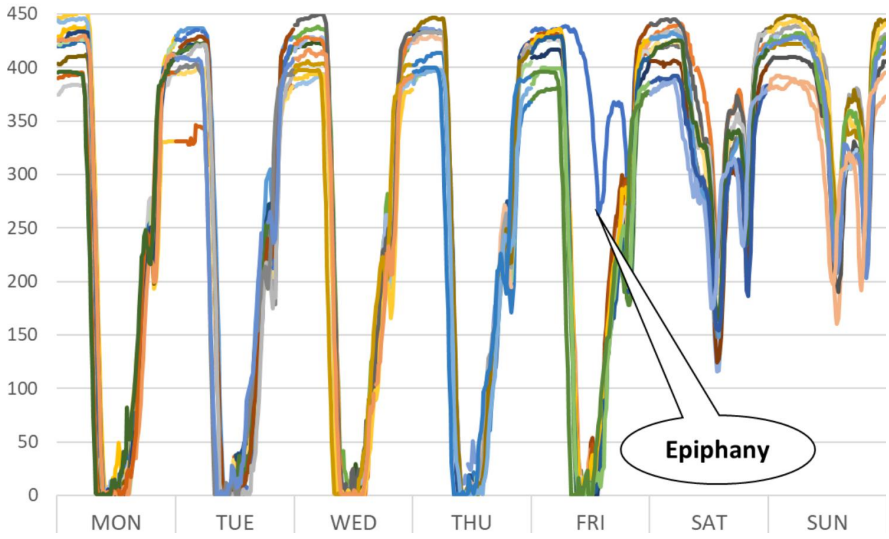



Figure 8.3: Typical daily trends of free slots every 15 minutes in park Stazione Fortezza Fiera.

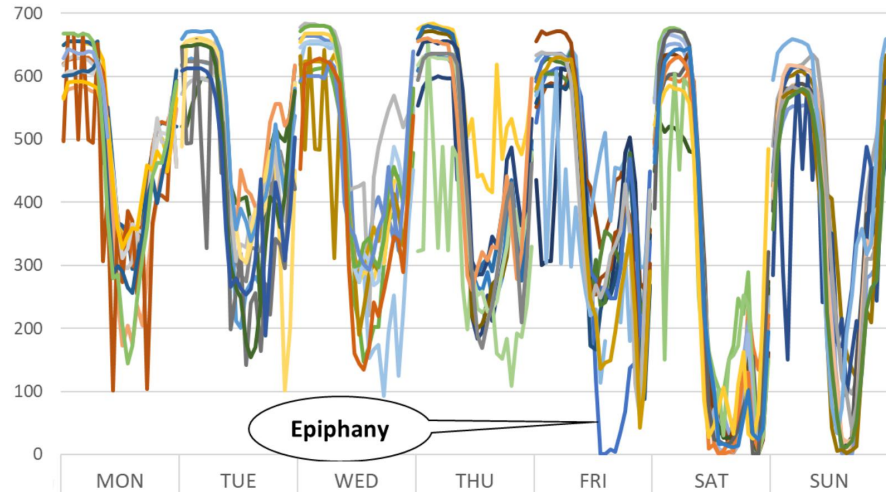
According to Figures 8.1(a) and 8.1(b), the workdays are readily recognizable with respect to the weekends (numerically higher since they 5 days of working over the week). This strong difference is present in cases 8.1(a) and 8.1(b) which are both located close to hospital: Careggi and Pieraccini Meyer car parks, respectively.

On the contrary, in the cases of parking areas S. Lorenzo and Beccaria (see Figures 8.2(a) and 8.2(b)) the number of free slots seems to be independent on the workdays, and the daily trend is not repeated regularly. Case of Figure 8.2(b) represents an area of Florence's nightlife and restaurants: it is evident that in some cases, sporadically, the parking garage is full at lunch/dinner time and after dinner time. Moreover, in Figure 8.3a more chaotic case is reported: the garage is located between the main train station and the fair area of Florence. In this case, the presence of large expositions can make the difference as the presence of *Pitti Fashion Expo* from the 9th to 13th of January 2017.

Moreover, For Careggi and S.Lorenzo car parks (case Figures 8.1(a) and




(a)




(b)

Figure 8.4: Typical weekly trends of free slots every 15 minutes in parks (a) Careggi and (b) S.Lorenzo.

8.2(b)), the weekly curves are shown in Figures 8.4(a) and 8.4(b) respectively. It should be noted that in Figure 8.4(a) the trend is more defined and the noise is lower than in Figure 8.4(b). In both of cases, it is possible to observe that festivity that fall during the week (the 6th of January, Epiphany vacation) had a trend totally similar to the weekend. According to [74], it is reasonable that changes in patterns between workdays and weekends can be due to different travel purposes: people that travel for work on workdays, and for entertainment on weekends. In this case, we add that the people have also a different trend for the passing time windows that the hospitals set up to allow visit patients and friends.

## 8.4 Features

Some features have been identified and are reported in Table 8.1; in which they are classified in three main groups. Each measurement containing all features is carried out within a 15-minute period.

**Baseline features** refer to the measures related to the free parking slots recorded every 15 minutes and including: number of free spaces, date and time, day of the week, festivity. These variables are used to consider the seasonality of the data that may have strongly different trends – i.e., the work days with respect to the weekend, etc. If the parking spaces have the same trend during the same day and time between different weeks, two other features have been included in the model: difference between the actual and previous number of free space at the same time, recorded one week before ($POD$); the difference between the actual number of parking spaces and the next one at the same time, recorded one week before ($SOD$). At a specific observation of a specific date and time corresponds the $POD$ and $SOD$ of the previous week (see Figure 4).

**Weather feature** (i.e., temperature, humidity and rainfall) are collected every 15 minutes. According to our analysis, the significant values are those related to the hour before. In order to predict the number of free spaces in a garage at 3 pm, the weather features related to 2 pm on the same day are relevant. In fact, the weather conditions typically influence the decisions of taking the car or the public transportation. For example, the expected behavior of citizens when it's raining is to

drive the car instead of the motorcycle. By doing so, more parking lots will be occupied. In this case, also the weather forecast may influence the decisions and are accessible on the Km4City smart city knowledge base in real time. On the other hand, according to our experiments, they are less significant with respect to the real weather features.

**Traffic sensors feature** (i.e., vehicle flow, concentration, average time and average speed) refers to the values of the previous hour recorded by the sensors leading from the path to the garage's area. Traffic sensors for each garage may be one or more, and they should be chosen considering the direction of travel and the most likely route to reach the garage. Note that, there are four features per sensor. The assessment of the traffic sensors is used as a detector for identifying the occurrence of relevant events such as those of Figure 8.3. As an alternative, a specific solution for interpreting events could be performed. On Sii-Mobility, also the list of the major city events and their GPS coordinates in the city are available. .

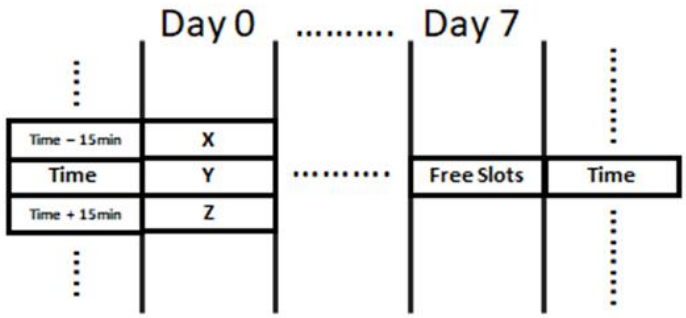



Figure 8.5: Construction of *POD* and *SOD* features described in Table 8.1

## 8.5 Forecasts Techniques

In the general framework, three different approaches were tested, i.e., Bayesian Regularized Neural Network (BRNN), Support Vector Regression (SVR) and ARIMA model applied on features presented above.

| Data Group | Feature | Description |
|---|---|---|
| **Baseline** | Free Parking Slots | Real number of free slots recorded every 15 minutes |
| | Time | Hours and minutes |
| | Month | Month of the year (1-12) |
| | Day | Day of the month (1-31) |
| | Day Week | Day of the week (0-6) |
| | Weekend | 0 for working days, 1 else |
| | Previous observation's difference (POD) | Difference between the number of free spaces at time i and number of free spaces at time ($i - 15$ minutes) recorded in the previous week |
| | Subsequent observation's difference (SOD) | Difference between the number of free spaces at time i, and the number of free spaces at time ($i + 15$ minutes) recorded in the previous week |
| **Weather** | Temperature | City temperature measured one hour earlier than Time, in °C |
| | Humidity | City humidity, measured one hour earlier than Time, in % |
| | Rainfall | City rain, measured one hour earlier than Time, in mm |
| **Traffic Sensors** | Average Vehicle Speed | Average speed of vehicles on the road closest to the parking, over one-hour period (km/h) |
| | Vehicle Flow | Number of vehicles passed closest to the parking, over one-hour period |
| | Average Vehicle Time | Average of distance between vehicles, over one-hour period |
| | Vehicle Concentration | Number of vehicles per kilometer, over one-hour period |

Table 8.1: Categories and description of features

### 8.5.1 Artificial Neural Networks with Bayesian Regularization

The Artificial Neural Network (ANN) is a very popular technique which relies on supervised learning. ANNs [34], [106], [120] proposed powerful nonlinear regression techniques inspired by theories about how the brain works. The primary application of neural networks involves the development of predictive models to forecast future values of a particular response variable from a given set of independent variables, and they are particularly useful in problems with a complex relationship between input and output. The outcome is modeled by an intermediary set of unobserved variables (hidden neurons), that are linear combinations of the original predictors. The connection between neurons in each layer is termed a link. This link is stored as a weighted value which provides a measure of the connection between two nodes [61], [65].

The supervised learning step changes these weights in order to reduce the chosen error function, generally mean squared error, in order to optimize the network for use on unknown samples. ANNs tend to overfit, which leads to a fitting of the noise and a loss of generalization of the network. On the other hand, Bayesian Regularized ANNs (BRANNs) attempt to overcome the overfitting problem by incorporating Bayes' modeling into the regularization scheme [36]. In general, the potential overfitting increases when a neural network grows in size through additional hidden layer neurons. BRANNs avoid overfitting because the regularization pushes unnecessary weights towards zero, effectively eliminating them. The BRANN method is more robust, parsimonious, and efficient than a classical ANN, and the network weights are more significant in modeling the phenomena [36].

The BRANN model fits a three-layer neural network as described in [87] and [56]. The layer weights the network, which is initialized by the Nguyen-Widrow initialization method [97], and thus, the model is given by:

$$y_i = g(x_i) + e_i$$

$$y_i = \sum_{k=1}^{s} w_k g_k \left( b_k + \sum_{j=1}^{p} x_{ij} \beta_j^{[k]} \right) + e_i, i = 1, \ldots, n$$

where:

- $e_i$ $N(0, \sigma_e^2)$;

- $s$ is the number of neurons
- $w_k$ is the weight of the $k-th$ neuron, $k = 1, \ldots, s$;
- $b_k$ is a bias for the $k-th$ neuron, $k = 1, \ldots, s$;
- $\beta_j^{[k]}$ is the weight of the $j-th$ input to the net, $j = 1, \ldots, p$;
- $g_k()$ is the activation function: in this case

$$g_k(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

The objective function consists of minimizing $F = \alpha E_W + \beta E_D$, where $E_W$ is the sum of squares of network parameters (weight and bias), $E_D$ is the error sum of squares, $\alpha$ and $\beta$ are the objective function parameters.

### 8.5.2 Support Vector Regression

The SV (Support Vector) algorithm is a nonlinear generalization of the generalized Portrait algorithm developed in Russia in the sixties and further developed for decades [123]. This theory characterizes properties of learning machines which enable them to effectively generalize the unseen data, and excellent performances have been obtained in regression and time series prediction applications [52]. SVR model with linear kernel has been adopted in this paper as a predictive method. The idea of SVR is based on the computation of a linear regression function

$$f(x) = w^T x + b$$

to a given data set

$$(x_i, y_i)_{i=1}^N$$

in a high dimensional feature space where the input data are mapped via a nonlinear function. Instead of minimizing the observed training error, SVR attempts to minimize the generalization error bound so as to achieve generalized performance. This generalization error bound is the combination of the training error and a regularization term that controls the complexity of the hypothesis space [111].

### 8.5.3 ARIMA models

The autoregressive integrate moving average approach (ARIMA) has been adopted as alternative solution to set up accurate predictive models. The autoregressive part (AR) of the model creates the basis of the prediction, which can be improved by a moving average modelling for errors made in previous time instants of prediction (MA). The order of ARIMA modelling is defined by the parameters (p, d, q): p is the order of autoregressive model; d is the degree of differencing, and q is the order of the moving average part, respectively; and by the corresponding seasonal counterparts (P, D, Q). The predictive model can been developed by using Box-Jenkings methodology for ARIMA modelling [35].

## 8.6 Evaluate Tecniques with MASE

According to the literature, most of the researchers use MAPE or MSE to calculate the prediction error. This choice of error measurements is due to the fact that on the particular issue of street-parking predictions, critical cases where the parking slots are close to zero are not present or not analyzed. Measures based on percentage errors (e.g., MAPE) have the disadvantage of being infinite or undefined if the observed value is equal to zero during the period of interest, and having extreme values when any observation is close to zero. On the other hand, if parking garages issue is analyzed, the possibility of reaching critical cases is a problem to be taken into account: most of the garages depicted in Figures 8.1, 8.2 and 8.3, every day for several hours are full (the number of free parking slots is equal to zero). For this reason, we have chosen the Mean Absolute Scaled Error (MASE) [71], that is one the most widely used metrics for the assessment of prediction accuracy as an alternative to the percentage errors. Mean Absolute Scaled Error is calculated as follows

$$MASE = mean(|q_t|)$$

and

$$q_t = \frac{obs_t - pred_t}{\frac{1}{n-1}\sum_{i=2}^{n}|obs_i - obs_{i-1}|}$$

where:

- $obs_t$ = observation at time t,
- $pred_t$ = prediction at time t,
- $n$ is the number of the values predicted over all test sets (96 daily observations per 7 days)

Note that, MASE is clearly independent of the scale of the data. In comparing models, the best is the one presenting the smaller MASE, and thus the corresponding predictions can be considered the best among those compared.

## 8.7 Experiments and Results

In the general framework, three different approaches were tested – i.e., BRANN, SVR and ARIMA model – applied on the features presented above. In detail, the number of input neurons in BRANN model corresponds to the number of the features reported in Table 8.1. Note that, all features are considered as an individual neuron, except Time that has 96 neurons, one for each slot of 15m ("00:00", "00:15",..., "23:45"). A single output neuron represents the predicted value. The model fits a three-layer neural network with three intermediate neurons – i.e., the number of neurons that correspond to the lowest error rate [132]. The processing time comparison, among the above-considered models, is also relevant and it is reported in Table 8.2 for a single car parking garage. From Table 8.2, it is evident that all the approaches are capable to produce predictions every hour for the next hour in advance with a quite low average estimation. On one hand, the ARIMA approach needs to re-compute the training every hour to produce satisfactorily significant predictions, which is a very large cost of about 9s for each car park. On the other hand, BRANN and SVR allow being "trained" once a day, providing predictive models with 96 values in advance with quite precise results. For this reason, the ARIMA solution has been discharged as performed by other researchers in the literature, as reported in Section 1. Note that, the identified ARIMA was $(5, 1, 2)x(1, 0, 1)$ and allowed to perform short-term predictions with a MASE of about 1.2.

The aim is not only to find a satisfactory solution to make predictions that are computationally viable and capable to adapt for the several cases but also to produce satisfactory results in terms of precision in the critical

| **Training** | **Forecasts Techniques** | | |
|---|---|---|---|
| | **BRNN** | **SVR** | **ARIMA** |
| Average training processing time, in s | 76.3 | 9.1 | 9.2 |
| Re-Training frequency | Daily | Daily | Hourly |
| Training period | 3 Months | 3 Months | 3 Months |
| **Estimation** | **BRNN** | **SVR** | **ARIMA** |
| Average estimation processing time, in s | 0.0031 | 0.0052 | 0.0015 |
| Estimation frequency | Hourly | Hourly | Hourly |
| Estimation length | 1 Hour | 1 Hour | 1 Hour |

Table 8.2: Comparison among models processing time in training and estimation, for a single garage

cases discussed before. Therefore, the comparison has been focused in that direction and by considering BRANN and SVR based on the whole set of car parks in Florence. In Table 3, some examples related to the reference cases of Figure 8.1, 8.2 are reported. In Table 8.3, MASE over the predicted week and the specific MASE relating to morning, afternoon, evening and night, have been reported for each of the predicted numbers of free parking lots. The comparison puts is evidence that the most reliable results are produced by BRANN approach, especially in critical time slots, where the car parking garages risk to be full. This fact is also highlighted by the best MASE for BRANN in all reference cases.

Furthermore, a comparison in terms of R-squared, RMSE, and MASE of the BRNN approach has been computed considering four combinations of the different categories of data (reported on in Table 8.1): baseline features; baseline and weather features; baseline and traffic sensors; baseline, weather, and traffic sensors features together.

According to results reported in Table 8.4, the differences among the different cases are not very relevant. The results suggest that the best choice in terms of precision still is the use of model exploiting the baseline only. On the other hand, extending the assessment to all parking garages the results are substantially different.

In fact, Figure 8.6 reports the comparison of the 4 models compared in

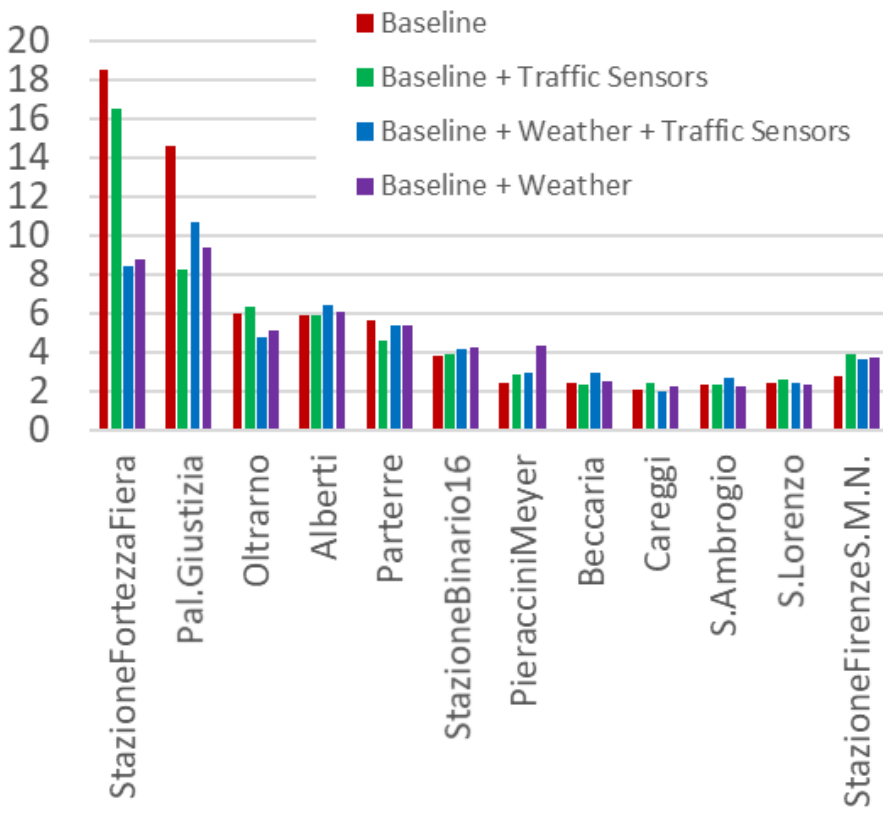



Figure 8.6: Comparison among the predictive models for the 12 garages in Florence in terms of MASE assessed in the last two weeks of predictions.

| Comparison Error | Forecasts Techniques | |
|---|---|---|
| | **BRNN** | **SVR** |
| **Careggi** | | |
| MASE Night | 34.85 | **16.29** |
| MASE Morning | **0.76** | 1.42 |
| MASE Afternoon | **1.89** | 4.34 |
| MASE Evening | 1.99 | **1.51** |
| MASE Daily | **1.87** | 2.34 |
| **Pieraccini Meyer** | | |
| MASE Night | **6.08** | 12.83 |
| MASE Morning | **0.86** | 1.27 |
| MASE Afternoon | **1.87** | 2.91 |
| MASE Evening | **1.36** | 1.57 |
| MASE Daily | **1.37** | 2.06 |
| **San Lorenzo** | | |
| MASE Night | **10.33** | 11.81 |
| MASE Morning | 2.13 | **1.91** |
| MASE Afternoon | **2.70** | 3.15 |
| MASE Evening | **2.15** | 3.09 |
| MASE Daily | **2.72** | 3.21 |
| **Beccaria** | | |
| MASE Night | 9.32 | **7.80** |
| MASE Morning | **0.95** | 1.25 |
| MASE Afternoon | 2.49 | **2.14** |
| MASE Evening | **2.96** | 4.75 |
| MASE Daily | **2.13** | 2.67 |

Table 8.3: Comparison among predictive models estimated considering only the baseline features. MASE is estimated on a testing period of 1 week after the 27 of March for: Careggi, Pieraccini Meyer, S.Lorenzo, and Beccaria car parks.

Table 8.4 extended to all garages on the basis of the estimation of MASE for the last week. The comparison puts in evidence that in most cases in which the daily trend for free slots is regular (such as cases (a) and (b) of Figure 8.1, Careggi or Pieraccini Meyer), the 4 models proposed produce

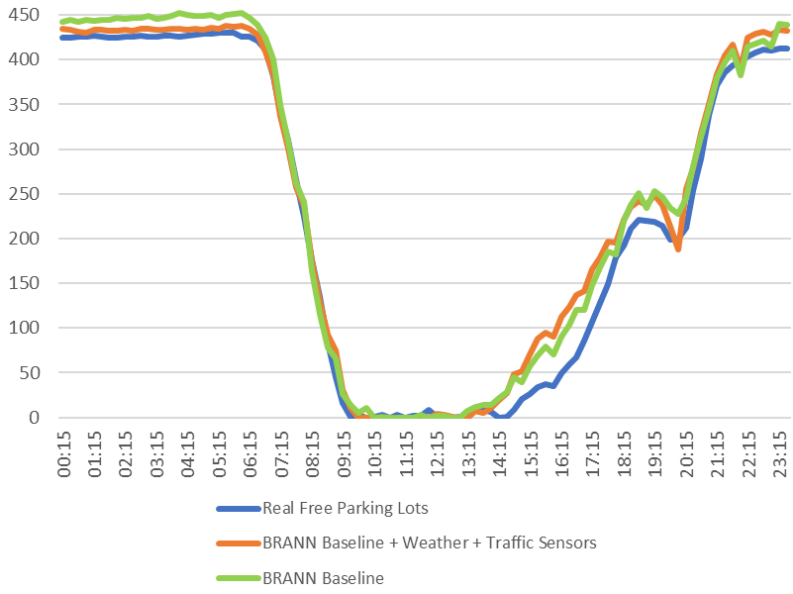
| **Model Features** | **BRANN Model Results** | | |
|---|---|---|---|
| | **R-squared** | **RMSE** | **MASE** |
| **Careggi** | | | |
| Baseline | **0.974** | **24** | 1.87 |
| Baseline + Weather | 0.975 | **24** | **1.75** |
| Baseline + Traffic sensors | 0.975 | **24** | 2.04 |
| Baseline + Weather + Traffic Sensors | 0.975 | **24** | 1.87 |
| **Beccaria** | | | |
| Baseline | **0.888** | 16 | **2.13** |
| Baseline + Weather | 0.890 | **15** | 2.15 |
| Baseline + Traffic sensors | 0.892 | 16 | 2.24 |
| Baseline + Weather + Traffic Sensors | 0.895 | 16 | 2.33 |

Table 8.4: BRNN training model results in terms of R-squared, RMSE and the estimated prediction error MASE for Careggi and Beccaria car parks

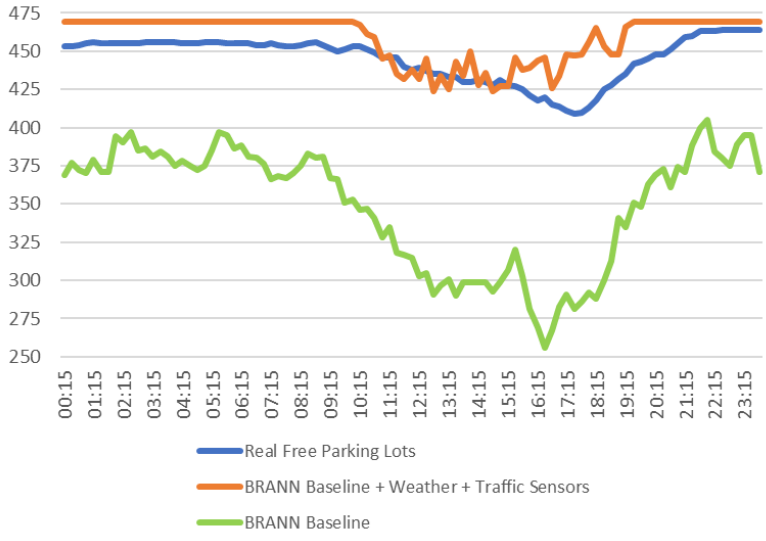
results with no relevant results among each other. On the contrary, when critical cases such as case of Figure 8.3 is present (Stazione Fortezza Fiera, Palazzo di Giustizia), a strongly non-stationary trend is present that can be substantially corrected by adding weather features and traffic sensors.

In detail, Figure 6 depicts the comparison of: (i) the real daily trend; (ii) the prediction using baseline features only; (iii) the prediction using the combination of baseline, weather and traffic sensors features, for Careggi and Stazione Fortezza Fiera car parks. Note that, the addition of weather and traffic sensors features decreases the difference between the real values and the predictions in the Stazione Fortezza Fiera car park.

In order to validate the above considerations, in Figure 8.8 the features listed in Table 8.1 are reported in order of importance for the BRANN full model prediction – i.e., the model with all the categories of covariates: the relationship between each predictor and the outcome is evaluated. A loess smoother is fitted between the outcome and the predictor, while the $R^2$ statistic is calculated for this model against the null model (intercept only). The histogram suggests that variable Time (of the baseline) is the most relevant for predicting the number of free slots for all garages. The second one, in terms of relevance, results to be Average Vehicle Time of the traffic sensors features. According to these results, traffic variables are of primary




(a)




(b)

Figure 8.7: Comparison between the actual trend of free parking lots and the predicted trend on the basis of BRANN using baseline features and all features for (a) Careggi, (b) Stazione Fortezza Fiera car park for 24 hours.

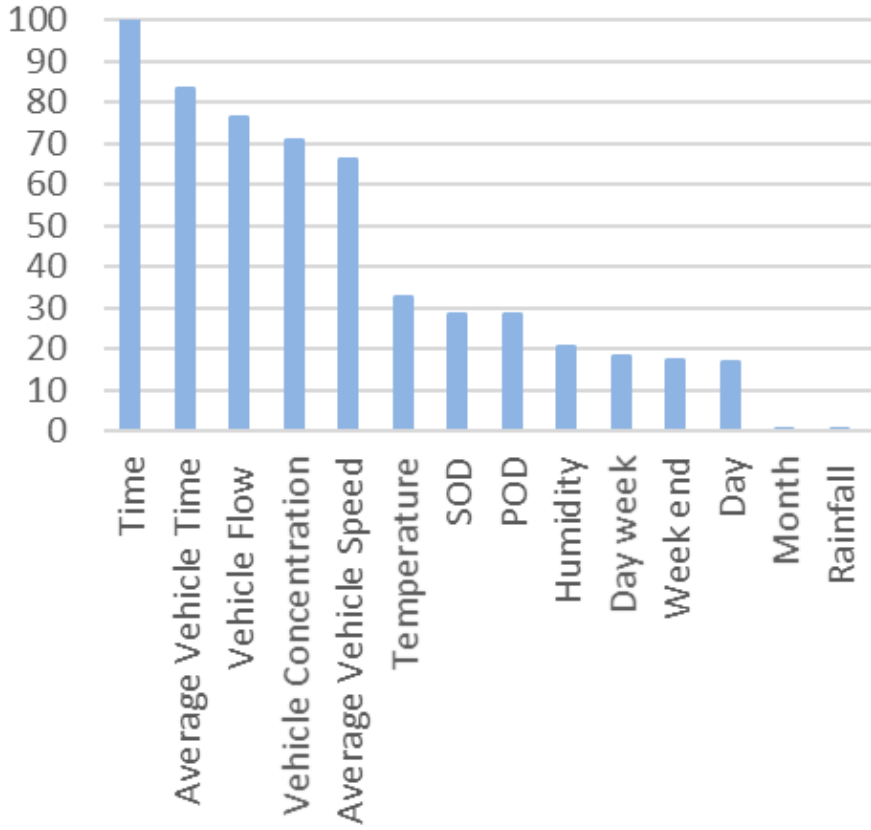



Figure 8.8: Variables Importance of the BRNN full model.

importance, as already mentioned in [129]. These statements seem to be not coherent with the finding of [102], where traffic was found to be of secondary importance. This may be due to the fact that, considering garages, it is easier to choose traffic sensors related to the car park analyzed – i.e., those sensors on the streets leading from the path to the garage. Contrarily, in street-parking prediction, only the general traffic situation may be related. Note that, the chose of sensors can be done only in the case in which the data are publicly available, as emphasized in [102]. In fact, authors who develop IPSs on garages seem to focus only on free parking data for forecasting [115], [128], [39]. In our case, having more public data available, it has been possible to integrate these data with those of the other features usually used in the predictions made on road parking lots. To clarify, the proposed model uses the real number of free parking lots and the real data recorded by traffic sensors leading from the same path to the garage, every 15 minutes. Finally, our results cannot directly comparable in terms of prediction error, because we have analyzed critical cases not reported in the literature before: when the parking slots are close to zero, measures based on MAPE and MSE have the disadvantage of being infinite or undefined. For this reason, MASE is the best choice, resulting to be 1.75 in the best case.

# Conclusions

The research activity is carried out at the DISIT laboratory (Distributed Data Intelligence and Technologies) of the DINFO department (Department of Information Engineering) of the University of Florence.

The activities performed during the PhD have focused on the study of the following topics:

1. iCaro cloud simulator exploiting knowledge base (Part I)
2. A Smart City Development kit for designing Web and Mobile Apps (Part II)
3. Predicting free parking slots on critical and regular services exploiting a range of open data (Part II - Chapter 8)

**iCaro cloud simulator exploiting knowledge base**

This research activity was carried out within the iCaro project in collaboration with Computer Gross Italia, Liberologico and CircleCap.

iCaro is a project that aims to produce prototypes of innovative technological solutions to solve the difficulties that Italian Small Medium Enterprises (SMEs) encounter by basing their business on non-cloud services hosted on their local servers that are poorly adapted to business evolution, guaranteeing a gradual integrated access to cloud services such as Business Platform As A Service (BPaaS) and offering flexibility and scalability, cost reduction, maximum confidentiality and data security.

One of the parameters that mainly affects the costs of the entire cloud platform is the number of physical machines active: the higher this number, the more the data center will consume energy and the more it will need maintenance. In order to make an optimal allocation, account must be taken

of both the reduction in the number of active physical machines and the contracts signed with customers, which contain constraints to be respected for the resources offered.

Since it is not possible to work directly on the physical production machines, a simulator is needed to find an allocation as close to the optimal solution as possible, given the Virtual Machines (VMs) currently present within the cloud platform. Moreover, the simulator can help to decide in which physical machine a new VM will have to be allocated in order not to move too much from the previously found allocation (since every VM move has a cost).

The analysis of the state of the art leads to the identification of the three main problems that the simulators analyzed have:

- No tools are provided to generate the workloads to be simulated in a simple and intuitive way for non programmers
- Not allowing a high level description of the workloads which must be described by low-level information such as: time to execute and main memory used at level of CPU
- Do not allow a dynamic modification of the simulated data center structure by adding new resources during simulation.

For these reasons, a framework has been developed to characterize the trend of any temporal signal and to characterize the trend of several correlated temporal signals: with this instrument it is therefore possible to characterize the workload of a VM for a certain period of time and for all the resources that are deemed necessary. In order to be able to use the models generated by the framework in an easy way and to perform simulations of cloud platforms functioning, a webapp has been created which, in addition, allows to interact with the other components of the iCaro project and in particular with:

**KB (Knwoledge Base)** ontological database containing the description and associations between entities that can be found in a cloud platform

**SM (Supervisor & Monitor)** a monitoring tool that uses NAGIOS (application for computer monitoring) to collect point data on which to calculate high level metrics to be entered in the KB.

This integration is necessary to massively add new entities to the KB, analyze the metrics entered by SM within the KB and collect real host data recoverable from SM.

Through the webapp, using the templates generated by the framework, it is possible to simulate the workload of a data center and find the best allocation for it using the heuristics of solving the Vector Bin Packing problem: bin can be considered as the physical machines to be *filled* with VMs and each resource is a dimension that needs to be considered during the allocation phase.

Finally, the interface and simulation logic have become completely independent, so that it is easier to subdivide the workload generated by the simulations between several processes that can be executed in different machines.

**A Smart City Development kit for designing Web and Mobile Apps**

This research activity is carried out within the projects:

**Sii-Mobility** National Smart City http://www.sii-mobility.org for the study of mobility and transport aspects: for the evaluation of service quality, for the study of events; and

**RESOLUTE H2020** http://www.resolute-eu.org for resilience aspects, data collection related to mobility, transport system, flows of people in the city and risk assessment.

These projects use the model and tools developed by Km4City (Knowledge Model For City): a framework, created within the DISIT laboratory, which provides a single access point for interoperable city data across the web or mobile platforms. Km4City covers mobility and transport, energy, banking, car parks, commerce, culture, cycling, green areas, health, tourism and much more. Km4City today processes more than 1 million new data in real time per day, making them accessible in an aggregated way and producing suggestions, trajectories, destination origin maps, answers to searches, predictions, decision support, etc.

A hybrid application has been developed (Tuscany where, what... Km4City) that uses the model and tools made available by the Km4City framework and has been created features that allow to have a good user experience and to analyze the use of the application by users. In addition, a background

service has been created that collects data on public Wi-Fi in the closeness of users and logs anonymously information about the location and movements of the devices that have installed the application. Research has focused on making this service as robust as possible in the background and is continuously refined to minimize battery consumption, bandwidth used for sending data and data logging errors. Data were collected from each individual device and it was possible, thanks to this data, to develop two components to better support the daily life of the user:

**Recommender** suggests to the user if there are services nearby that can be of interest to the user based on previous searches by the user, his profile and his GPS location.

**Engager** Ask the user to perform actions within the application, for example, if he wants to write a review of a particular service, send customized surveys or warns, for example, that you have parked the car in an area where you do not have permission to park. It is based on the location of the GPS and the permanence of the device in a certain area.

Correct operation of the service in the background is essential to allow these components to perform their calculations on the correct data and that is why it is developed to make it as robust as possible. By analyzing these data, it has been possible to modify the menus that allow the users to search for the desired points of interest in order to show each type of user (citizens, tourists, commuters and students) a menu that is more relevant to their category. An architecture and a creation of these menus has been developed that allows remote editing without updating the application, so that each user has the updated version of the menu at all times.

With the experience gained in the design and development of the application, the research focused on creating a *Mobile Application Development Kit*, based on the application architecture, allowing other developers to create their own module that can be integrated without problems within the application already developed. Through this *Mobile Application Development Kit* it is possible to realize in a short time, without being aware of how a hybrid application can be developed, its own application containing its own modules that are developed in HTML, JS and CSS.

**Predicting free parking slots on critical and regular services exploiting a range of open data**

One of the most important challenges for a Smart City is to make itself more liveable for its citizens and this can be done, for example, by reducing the pollution that is generated by those who live there. Think about how many times, in large cities, it can happen to turn around the streets, before finding a free place to park: during the time of searching for a free place the car continues to pollute and this pollution could be reduced simply by finding a way to park faster. It can be perceived how useful it could be for a driver looking for parking, to have real-time information on the availability of free parking spaces in the nearby car parks.

The data in real time, however, is limited if we consider commuters or people approaching larger centers to entertain themselves in the evening or to have services that are not provided in the suburbs (think of the large hospitals that are usually found in the capital cities). These people would like to know in advance what the parking will be with more free parking spaces in the vicinity of your destination.

To help these people and, consequently, reduce the pollution generated during the search for parking, a machine learning algorithm has been developed, based on data relating to controlled access parking lots (large paid parking lots with bar), which carries out daily training of a *Bayesian Regularized Neural Network* (BRNN) and on this training are generated predictions (every 15 minutes) of how many free spaces will be available, in a given parking lot, at a distance of an hour.

In addition to the BRNN, two other models based on ARIMA and SVR were studied, but they were discarded respectively due to the excessive time taken to carry out training and prediction and the greater prediction error than the neural network.

# Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.

## International Journals

1. **C. Badii**, P. Bellini, D. Cenni, A. Difino, P. Nesi , M. Paolucci. "Analysis and assessment of a knowledge based smart city architecture providing service APIs", *Future Generation Computer Systems*, Volume 75, October 2017, Pages 14-29. [DOI: 10.1016/j.future.2017.05.001]
2. **C. Badii**, P. Bellini, I. Bruno, D. Cenni, R. Mariucci, P. Nesi. "Icaro cloud simulator exploiting knowledge base", *Simulation Modelling Practice and Theory*, Volume 62, March 2016, Pages 1-13 [DOI:10.1016/j.simpat.2015.12.001]

### Submitted

1. **C. Badii**, P. Nesi, I. Paoli. "Predicting free parking slots on critical and regular services exploiting a range of open data", *IEEE Access*, 2018. (Submitted)
2. **C. Badii**, A. Difino, P. Nesi, I. Paoli, M. Paolucci. "Automated Classification of Users' Transportation Modality in Real Conditions", *IEEE IEEE Intelligent Transportation Systems Transactions*, 2017. (Submitted)

## International Conferences and Workshops

1. **C. Badii**, P. Bellini, D. Cenni, A. Difino, M. Paolucci, and P. Nesi, "User engagement engine for smart city strategies", in Smart Computing (SMARTCOMP), 2017 IEEE International Conference on. IEEE, 2017, pp. 1-7. [10.1109/SMARTCOMP.2017.7947059]
2. **C. Badii**, P. Bellini, P. Nesi, M. Paolucci, "A Smart City Development kit for designing Web and Mobile Apps", IEEE international Conference on

Smart City and Innovation, 2017, San Francisco. With 2017 IEEE Smart World Congress.

3. **C. Badii**, P. Bellini, D. Cenni, G. Martelli, M. Paolucci, and P. Nesi, "Km4city smart city api: an integrated support for mobility services", in Smart Computing (SMARTCOMP), 2016 IEEE International Conference on. IEEE, 2016, pp. 1-8. [10.1109/SMARTCOMP.2016.7501702]
4. **C. Badii**, P. Bellini, I. Bruno, M. Di Claudio, G. Martelli, P. Nesi, N. Rauch, "Km4City as Smart City Semantic Model and Tools", Open Challenge of 14th International Semantic Web Conference (ISWC2015) Bethlehem, PA, USA, October 11-12, 2015, Press by SPRINGER

## National Conferences

1. **C. Badii**, P. Bellini, D. Cenni, P. Nesi, M. Paolucci, "From Data to Sensing Users by exploiting Km4City Services", CINI Annual Conference on ICT for Smart Cities & Communities (I-CiTies2016) Benevento, Italy, September 29-30, 2016.
2. **C. Badii**, E. Bellini, P. Bellini, D. Cenni, A. Difino, P. Nesi, M. Paolucci,"Km4City: una soluzione aperta per erogare servizi Smart City", Conferenza GARR 2016, Firenze, Italy, 30 November - 2 December, 2016

## Technical Reports

1. **C.Badii**, D. Cenni, "Manuale d'uso Modulo SCE", ICARO-DE-4-35-1-v2-0
2. **C.Badii**, "Kit di sviluppo per applicazioni fisse e mobili", Sii-Mobility-DE3-20
3. **C.Badii**, I.Zaza, A. Di Fino, M. Paolucci, "Moduli di connessione con SII per mobile, web e applicazioni fisse", Sii-Mobility-DE3-18a-ModConSIIMobWebApp

# Acronyms

| | |
|---|---|
| **ACO** | Ant Colony Optimization |
| **ADK** | Application Development Kit |
| **AGPL** | Affero General Public License |
| **API** | Application Programming Interface |
| **AP-TAS** | Asymptotic Polynomial Time Approximation Scheme |
| **ARIMA** | AutoRegressive Integrated Moving Average |
| **BC** | Business Configuration |
| **BPaaS** | Business Platform as a Service |
| **BPP** | Bin Packing Problem |
| **BPA** | Bin Packing Algorithms |
| **BRNN** | Bayesian Regularized Neural Network |
| **CC** | Cloud Computing |
| **CCM** | Cloud Configuration Manager |
| **CDN** | Content Delivery Network |
| **CoC** | Criterion of Clustering |
| **CPU** | Central Processing Unit |
| **CRM** | Customer Relationship Management |
| **CSS** | Cascading Style Sheets |

| | |
|---|---|
| **d-DVP** | d-Dimensional Bin Packing |
| **DINFO** | Department of Information Engineering of University of Florence |
| **DISCES** | Distributed Smart City EngineDISCES |
| **DISIT** | Distributed Systems and Internet Technologies Lab & Distributed Data Intelligence and Technologies Lab of DINFO |
| **EERM** | Economically Enhanced Resource Manager |
| **ERP** | Enterprise resource planning |
| **ETL** | Extract Transform and Load |
| **FFD** | First Fit Decreasing |
| **GDF** | Geographic Data Files |
| **GPS** | Global Position System |
| **GUI** | Graphical User Interface |
| **HLM** | High Level Metrics |
| **HPC** | High Performance Computing |
| **HTML** | Hypertext Markup Language |
| **IaaS** | Infastructure as a Service |
| **ICLOS** | iCaro Cloud Simulator |
| **IP** | Internet Protocol |
| **JS** | JavaScript |
| **JSON** | JavaScript Object Notation, a lightweight data-interchange format easy for humans to read and write and, for machines, to parse and generate |
| **KB** | Knowledge Base |
| **Km4City** | Knowledge Model For City |

| | |
|---|---|
| **KP** | Knapsack Problem |
| **LP** | Linear Programming |
| **MAC** | Media Access Control |
| **MASE** | Mean Absolute Scaled Error |
| **MIT** | Massachusetts Institute of Technology |
| **MKP** | Multi-choice Knapsack Problem |
| **MMKP** | Multi-dimensional Multi-choice Knapsack Problem |
| **MMMKP** | Multi-dimensional Multi-choice Multi-Knapsack Problem |
| **NAGIOS** | Nagios Ain't Gonna Insist On Sainthood |
| **NIST** | National Institute of Standards and Technology |
| **NoSQL** | No Structured Query Language |
| **NP** | Nondeterministic Polynomial Time |
| **OS** | Operating System |
| **OWL** | Web Ontology Language |
| **PaaS** | Platform as a Service |
| **QoS** | Quality of Service |
| **RAM** | Random Access Memory |
| **RDF** | Resource Description Framework |
| **RDFS** | Resource Description Framework Schema |
| **REST** | Representational State Transfer |
| **RM** | Resource Manager |
| **RMSE** | Root Mean Square Error |
| **RRD** | Round Robin Database |
| **SaaS** | Software as a Service |

| | |
|---|---|
| **SCE** | Smart Cloud Engine |
| **SE** | Simulation Error |
| **SLA** | Service Level Agreement |
| **SM** | Supervisor & Monitor |
| **SME** | Small Medium Enterprise |
| **SPARQL** | SPARQL Protocol and RDF Query SQL Structured Query Language |
| **SSAP** | Static Server Allocation Problem |
| **SSID** | Service Set Identifier |
| **SV** | Support Vector |
| **SVM** | Support Vector Regression |
| **TS** | Time Series |
| **TCP** | Transmission Control Protocol |
| **VBP** | Vector Bin Packing |
| **VM** | Virtual Machine |
| **W3C** | World Wide Web Consortium |
| **WS** | Web Service |
| **XML** | Extensible Markup Language |

# List of Figures


2.1 Allocate VMs for high priority applications. . . . . 37

3.1 File exported with daytime, weektime and monthtime measurements . . . . . 46
3.2 Partial representation of the resource description file memory for a generic machine . . . . . 47
3.3 Possible patterns of a web server machine . . . . . 51
3.4 Possible patterns of a elaboration machine . . . . . 52
3.5 Trend of simulation error as tolerance increases . . . . . 57
3.6 Trend of variance as tolerance increases . . . . . 58
3.7 Single patterns results in the value model average. In red the mean value, in black the cluster . . . . . 58
3.8 Error and variance with respect to tolerance . . . . . 59
3.9 Simulated daily workload for memory and CPU of a single VM 61
3.10 Setting a negative risk value ensures better noise resistance for machines. This approach can be used to ensure high responsiveness of applications. . . . . 64
3.11 By setting a positive value for the risk, compaction increases, but the machine becomes more susceptible to delays caused by changes in the workload not foreseen during allocation. . . 64

4.1 iCaro cloud architecture with cloud simulator . . . . . 67
4.2 Ontology under Knowledge Base . . . . . 70
4.3 ICARO supervisor and monitor . . . . . 71

5.1 ICLOS architecture. . . . . 77
5.2 ICLOS architecture. . . . . 79
5.3 Example of three patterns created by Pattern Generator . . . 81

5.4 ICLOS simulation results on SM. . . . . 83
5.5 Simulations for allocating VMs by using FFD Prod algorithm from 500 to 64000 VMs: (a) trend of execution time and number of VMs and Hosts (20 simulations), (b) estimation of the executing time cost with respect to the number of VMs and Hosts. . . . . 87
5.6 Simulations for allocating VM by using FFD Prod algorithm from 500 to 2000 VMs, workload patterns for 1 month according to the described mix of BC, 10 simulations for each estimation . . . . . 88

6.1 Ontology macro-classes and their connections . . . . . 94
6.2 Sii-Mobility Architecture. . . . . 97

7.1 ServiceMap Development Tools. . . . . 100
7.2 Web and Mobile Application developer kit. . . . . 102
7.3 High level workflow of operations performed by tracker . . . . 105
7.4 Detailed workflow of operations performed by tracker . . . . . 106
7.5 Usability Test Results. . . . . 112
7.6 Statistics on clicks of the main menu. . . . . 113
7.7 Statistics on more researched categories. . . . . 114

8.1 Typical daily trends of free slots every 15 minutes in parks (a) Pieraccini Meyer and (b) Careggi. . . . . 120
8.2 Typical daily trends of free slots every 15 minutes in parks (a) Beccaria and (b) S.Lorenzo. . . . . 121
8.3 Typical daily trends of free slots every 15 minutes in park Stazione Fortezza Fiera. . . . . 122
8.4 Typical weekly trends of free slots every 15 minutes in parks (a) Careggi and (b) S.Lorenzo. . . . . 123
8.5 Construction of *POD* and *SOD* features described in Table 8.1 125
8.6 Comparison among the predictive models for the 12 garages in Florence in terms of MASE assessed in the last two weeks of predictions. . . . . 132
8.7 Comparison between the actual trend of free parking lots and the predicted trend on the basis of BRANN using baseline features and all features for (a) Careggi, (b) Stazione Fortezza Fiera car park for 24 hours. . . . . 135

8.8 Variables Importance of the BRNN full model. . . . . 136

# List of Tables


2.1 Comparison between software cloud simulators previously developed (First Set) . . . . . 24

2.2 Comparison between software cloud simulators previously developed (Second Set) . . . . . 26

3.1 Representation of a time-series object . . . . . 45

3.2 Possible patterns of a web server machine . . . . . 50

3.3 In column (A) the number of average clusters per group. In column (B) the number of average clusters per assembly when tolerance is set to zero. In column (C) the number of average clusters per group when tolerance is set using the minimum variance criterion. . . . . 60

5.1 ICLOS Simulations for power consumption assessment . . . . 84

5.2 ICLOS simulations for allocation by using different algorithms. The execution time refers to 20 executions of the allocation algorithm in simulation. The *host number* refers to the most probable number of hosts identified among the set of 20 simulations; in most cases, this number is the minimum number of hosts according to the goals of the adopted Bin Packing Algorithm (BPA). . . . . 85

8.1 Categories and description of features . . . . . 126

8.2 Comparison among models processing time in training and estimation, for a single garage . . . . . 131

8.3 Comparison among predictive models estimated considering only the baseline features. MASE is estimated on a testing period of 1 week after the 27 of March for: Careggi, Pieraccini Meyer, S.Lorenzo, and Beccaria car parks. . . . . 133
8.4 BRNN training model results in terms of R-squared, RMSE and the estimated prediction error MASE for Careggi and Beccaria car parks . . . . . 134

# Bibliography


[1] Cisco. [Online]. Available: http://www.cisco.com/c/dam/en_us/about/ac79/docs/ps/motm/Smart-City-Framework.pdf,

[2] Ckan. [Online]. Available: http://www.ckan.org

[3] Datex ii. [Online]. Available: http://www.datex2.eu/sites/www.datex2.eu/files/Datex_Brochure_2011.pdf

[4] Ibm institute for business value, how smart is your city? helping cities measure progress,. [Online]. Available: http://www.ibm.com/smarterplanet/global/files/uk__en_uk__cities__ibm_sp_pov_smartcity.pdf

[5] Ibm smart city. [Online]. Available: http://www-935.ibm.com/services/us/gbs/bus/html/smarter-cities.html

[6] Nagios. [Online]. Available: http://www.nagios.org

[7] Ontology of trasportation networks, deliverable a1-d4, project rewerse, 2005. [Online]. Available: http://rewerse.net/deliverables/m18/a1-d4.pdf

[8] Rrd. [Online]. Available: https://oss.oetiker.ch/rrdtool/tut/rrd-beginners.en.html

[9] Servicemap. [Online]. Available: http://servicemap.disit.org

[10] Standard performance evaluation corporation. [Online]. Available: http://www.spec.org

[11] M. A Vouk, "Cloud computing–issues, research and implementations," *CIT. Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[12] M. Aggarwal, "Introduction of cloud computing and survey of simulation software for cloud," *Research Journal of Science & IT Management, RJSITM*, vol. 3, no. 01, 2013.

[13] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 866–872.

[14] M. M. Akbar, E. G. Manning, G. C. Shoja, S. Shelford, and T. Hossain, "A distributed heuristic solution using arbitration for the mmmkp," in *Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107*. Australian Computer Society, Inc., 2010, pp. 31–40.

[15] M. Amoretti, F. Zanichelli, and G. Conte, "Efficient autonomic cloud computing using online discrete event simulation," *Journal of Parallel and Distributed Computing*, vol. 73, no. 6, pp. 767–776, 2013.

[16] S. An, B. Han, and J. Wang, "Study of the mode of real-time and dynamic parking guidance and information systems based on fuzzy clustering analysis," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 5. IEEE, 2004, pp. 2790–2794.

[17] L. Anthopoulos and P. Fitsilis, "Exploring architectural and organizational features in smart cities," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 190–195.

[18] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[19] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, and M. Paolucci, "Analysis and assessment of a knowledge based smart city architecture providing service apis," *Future Generation Computer Systems*, vol. 75, pp. 14–29, 2017.

[20] C. Badii, P. Bellini, I. Bruno, M. DiClaudio, G. Martelli, P. Nesi, and N. Rauch, "Km4city as smart city semantic model and tools (submitted for open track)," 2011.

[21] C. Badii, P. Bellini, D. Cenni, A. Difino, M. Paolucci, and P. Nesi, "User engagement engine for smart city strategies," in *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

[22] C. Badii, P. Bellini, D. Cenni, G. Martelli, M. Paolucci, and P. Nesi, "Km4city smart city api: an integrated support for mobility services," in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[23] C. Badii, P. Bellini, P. Nesi, and M. Paolucci, "A smart city development kit for designing web and mobile apps," in *First International Conference on Smart City and Innovation, 2017*. IEEE, 2017.

[24] N. Bansal, A. Caprara, and M. Sviridenko, "Improved approximation algorithms for multidimensional bin packing problems," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. IEEE, 2006, pp. 697–708.

[25] A. Bashar, "Modeling and simulation frameworks for cloud computing environment: A critical evaluation," in *International Conference on Cloud Computing and Services Science*, 2014, pp. 1–6.

[26] P. Bellini, M. Benigni, R. Billero, P. Nesi, and N. Rauch, "Km4city ontology building vs data harvesting and cleaning for smart-city services," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 827–839, 2014.

[27] P. Bellini, I. Bruno, P. Nesi, and N. Rauch, "Graph databases methodology and tool supporting index/store versioning," *Journal of Visual Languages & Computing*, vol. 31, pp. 222–229, 2015.

[28] P. Bellini, D. Cenni, and P. Nesi, "Cloud knowledge modeling and management," *Encyclopedia of Cloud Computing*, pp. 640–651, 2015.

[29] ——, "A knowledge base driven solution for smart cloud management," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on.* IEEE, 2015, pp. 1069–1072.

[30] P. Bellini, D. Cenni, P. Nesi, and I. Paoli, "Wi-fi based city users' behaviour analysis for smart city," *Journal of Visual Languages & Computing*, vol. 42, pp. 31–45, 2017.

[31] P. Bellini, P. Nesi, and A. Venturi, "Linked open graph: browsing multiple sparql entry points to build your own lod views," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 703–716, 2014.

[32] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[33] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on.* IEEE, 2010, pp. 577–578.

[34] C. M. Bishop, *Neural networks for pattern recognition.* Oxford university press, 1995.

[35] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[36] F. R. Burden and D. A. Winkler, "Robust qsar models using bayesian regularized neural networks," *Journal of medicinal chemistry*, vol. 42, no. 16, pp. 3183–3187, 1999.

[37] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on.* IEEE, 2009, pp. 1–11.

[38] F. Caicedo, "The use of space availability information in "parc" systems to reduce search times in parking facilities," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 1, pp. 56–68, 2009.

[39] F. Caicedo, C. Blazquez, and P. Miranda, "Prediction of parking space availability in real time," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7281–7290, 2012.

[40] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

[41] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[42] M. Caliskan, A. Barthels, B. Scheuermann, and M. Mauve, "Predicting parking lot occupancy in vehicular ad hoc networks," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th.* IEEE, 2007, pp. 277–281.

[43] P. Campegiani and F. L. Presti, "A general model for virtual machines resources allocation in multi-tier distributed systems," in *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on.* IEEE, 2009, pp. 162–167.

[44] P. Castillo, A. Fernández-Ares, P. García-Fernández, P. García-Sánchez, M. Arenas, A. Mora, V. Rivas, J. Asensio, G. Romero, and J. Merelo, "Studying individualized transit indicators using a new low-cost information system," *Handbook of Research on Embedded Systems Design, Advances in Systems Analysis, Software Engineering, and High Performance Computing*, pp. 388–407, 2014.

[45] C. Chekuri and S. Khanna, "On multidimensional packing problems," *SIAM journal on computing*, vol. 33, no. 4, pp. 837–851, 2004.

[46] X. Chen, "Parking occupancy prediction and pattern analysis," *Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS229-2014*, 2014.

[47] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *System Science (HICSS), 2012 45th Hawaii International Conference on.* IEEE, 2012, pp. 2289–2297.

[48] W. Cirne and F. Berman, "A comprehensive model of the supercomputer workload," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on.* IEEE, 2001, pp. 140–148.

[49] T. Cucinotta and A. Santogidis, "Cloudnetsim-simulation of real-time cloud computing applications," in *Proceedings of the 4th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, 2013.

[50] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on.* Ieee, 2010, pp. 27–33.

[51] A. Domingo, B. Bellalta, M. Palacin, M. Oliver, and E. Almirall, "Public open sensor data: Revolutionizing smart cities," *IEEE Technology and Society Magazine*, vol. 32, no. 4, pp. 50–56, 2013.

[52] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.

[53] M. Dyer, W. Riha, and J. Walker, "A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem," *Journal of Computational and Applied Mathematics*, vol. 58, no. 1, pp. 43–54, 1995.

[54] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[55] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, and P. Pucci, "Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors," in *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on.* IEEE, 2010, pp. 281–286.

[56] F. D. Foresee and M. T. Hagan, "Gauss-newton approximation to bayesian learning," in *Neural networks, 1997., international conference on*, vol. 3. IEEE, 1997, pp. 1930–1935.

[57] M. Gabay and S. Zaourar, "Variable size vector bin packing heuristics-application to the machine reassignment problem," 2013.

[58] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on.* IEEE, 2010, pp. 87–92.

[59] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of operations research*, vol. 1, no. 2, pp. 117–129, 1976.

[60] S. K. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on.* IEEE, 2011, pp. 105–113.

[61] D. G. Garson, "Interpreting neural network connection weights," 1991.

[62] C. Germain-Renaud and O. F. Rana, "The convergence of clouds, grids, and autonomics," *IEEE Internet Computing*, vol. 13, no. 6, pp. 9–9, 2009.

[63] T. Ghasemi and M. Razzazi, "Development of core to solve the multidimensional multiple-choice knapsack problem," *Computers & Industrial Engineering*, vol. 60, no. 2, pp. 349–360, 2011.

[64] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on.* IEEE, 2007, pp. 171–180.

[65] A. Goh, "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143–151, 1995.

[66] A. Haghani, M. Hamedi, K. Sadabadi, S. Young, and P. Tarnoff, "Data collection of freeway travel time ground truth with bluetooth sensors," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2160, pp. 60–68, 2010.

[67] J. L. Hellerstein. Google cluster data. [Online]. Available: :http://googleresearch.blogspot.com/2010/01/google-cluster-data.html

[68] M. Hepp, "Goodrelations: An ontology for describing products and services offers on the web," *Knowledge Engineering: Practice and Patterns*, pp. 329–346, 2008.

[69] M. Hifi, M. Michrafy, and A. Sbihi, "A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem," *Computational Optimization and Applications*, vol. 33, no. 2, pp. 271–285, 2006.

[70] U. Housing and M. S. OTB, "Smart cities ranking of european medium-sized cities."

[71] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[72] S. Iqbal, M. Bari, and M. Rahman, "Solving the multi-dimensional multi-choice knapsack problem with the help of ants," *Swarm Intelligence*, pp. 312–323, 2010.

[73] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*. Springer Science & Business Media, 2011.

[74] Y. Ji, D. Tang, P. Blythe, W. Guo, and W. Wang, "Short-term forecasting of available parking space using wavelet neural network model," *IET Intelligent Transport Systems*, vol. 9, no. 2, pp. 202–209, 2014.

[75] A. D. JoSEP, R. Katz, A. Konwinski, L. Gunho, D. PAttERSon, and A. RABKin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[76] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 1287–1294.

[77] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[78] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing*, vol. 12, no. 1, pp. 1–15, 2009.

[79] D. Lab. Sii-mobility app kit: App module development. [Online]. Available: http://www.disit.org/6992

[80] ——. Sii-mobility km4city: Smart city api. [Online]. Available: http://www.disit.org/6991

[81] W. H. Lam, Z.-C. Li, H.-J. Huang, and S. Wong, "Modeling time-dependent travel choice problems in road networks with multiple user classes and multiple parking facilities," *Transportation Research Part B: Methodological*, vol. 40, no. 5, pp. 368–395, 2006.

[82] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009, pp. 1–9.

[83] J. Liu, F. Zhao, X. Liu, and W. He, "Challenges towards elastic power management in internet data centers," in *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*. IEEE, 2009, pp. 65–72.

[84] M. Macías, J. O. Fitó, and J. Guitart, "Rule-based sla management for revenue maximisation in cloud computing markets," in *Network and Service Management (CNSM), 2010 International Conference on*. IEEE, 2010, pp. 354–357.

[85] M. Macıas, J. Guitart, and J. Girona, "Influence of reputation in revenue of grid service providers," in *Proceedings of the 2nd International Workshop on High Performance Grid Middleware (HiPerGRID 2008)*, 2008, pp. 9–16.

[86] M. Macías, O. Rana, G. Smith, J. Guitart, and J. Torres, "Maximizing revenue in grid markets using an economically enhanced resource manager,"

*Concurrency and Computation: Practice and Experience*, vol. 22, no. 14, pp. 1990–2011, 2010.

[87] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[88] R. Malhotra and P. Jain, "Study and comparison of cloudsim simulators in the cloud computing," *The SIJ Transactions on Computer Science Engineering & its Applications*, 2013.

[89] A.-L. Market and C. I. team, "Getting smart about smart cities understanding the market opportunity in the cities of tomorrow."

[90] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing–the business perspective," *Decision support systems*, vol. 51, no. 1, pp. 176–189, 2011.

[91] M. Martchouk, F. Mannering, and D. Bullock, "Analysis of freeway travel time variability using bluetooth detection," *Journal of Transportation Engineering*, vol. 137, no. 10, pp. 697–704, 2010.

[92] S. Martello, "Knapsack problems," *Algorithms and Computer Implementation*, 1990.

[93] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'neural-gas' network for vector quantization and its application to time-series prediction," *IEEE transactions on neural networks*, vol. 4, no. 4, pp. 558–569, 1993.

[94] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[95] F. Mendez, "System and method for monitoring people and/or vehicles in urban environments," *US Patent Application*, no. 2011/0128, p. 127, 2013.

[96] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[97] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," 1990.

[98] J. Nielsen and R. Budiu, *Mobile usability*. MITP-Verlags GmbH & Co. KG, 2013.

[99] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.

[100] F. Pan and J. R. Hobbs, "Temporal aggregates in owl-time." in *FLAIRS conference*, vol. 5, 2005, pp. 560–565.

[101] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," *research. microsoft. com*, 2011.

[102] C. Pflügler, T. Köhn, M. Schreieck, M. Wiesche, and H. Krcmar, "Predicting the availability of parking spaces with publicly available data." in *GI-Jahrestagung*, 2016, pp. 361–374.

[103] Z. S. Qian and R. Rajagopal, "Optimal parking pricing in general networks with provision of occupancy information," *Procedia-Social and Behavioral Sciences*, vol. 80, pp. 779–805, 2013.

[104] S. G. Ranu Pandey, "Comparative study of simulation tools in cloud computing environment," *Int. J. Sci. Eng. Res.*, vol. 5, no. 5, 2014.

[105] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems." *NCM*, vol. 9, pp. 44–51, 2009.

[106] B. D. Ripley, "Statistical ideas for selecting network architectures," in *Neural networks: Artificial intelligence and industrial applications*. Springer, 1995, pp. 183–190.

[107] G. Sakellari and G. Loukas, "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing," *Simulation Modelling Practice and Theory*, vol. 39, pp. 92–103, 2013.

[108] H. Schwetman, "Csim18-the simulation engine," in *Simulation Conference, 1996. Proceedings. Winter*. IEEE, 1996, pp. 517–521.

[109] J. M. Shapiro, "Smart cities: quality of life, productivity, and the growth effects of human capital," *The review of economics and statistics*, vol. 88, no. 2, pp. 324–335, 2006.

[110] D. C. Shoup, "Cruising for parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.

[111] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[112] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on services computing*, vol. 3, no. 4, pp. 266–278, 2010.

[113] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10. San Diego, California, 2008, pp. 1–5.

[114] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.

[115] D. Teodorović and P. Lučić, "Intelligent parking systems," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1666–1681, 2006.

[116] W. Tian, M. Xu, A. Chen, G. Li, X. Wang, and Y. Chen, "Open-source simulators for cloud computing: Comparative study and challenging issues," *Simulation Modelling Practice and Theory*, vol. 58, pp. 239–254, 2015.

[117] T. Tiedemann, T. Vögele, M. M. Krell, J. H. Metzen, and F. Kirchner, "Concept of a data thread based parking space occupancy prediction in a berlin pilot region." in *AAAI Workshop: AI for Transportation*, 2015.

[118] C. Tiexin, T. Miaomiao, and M. Ze, "The model of parking demand forecast for the urban ccd," *Energy Procedia*, vol. 16, pp. 1393–1400, 2012.

[119] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 385–392.

[120] M. Titterington, "Neural networks," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 1–8, 2010. [Online]. Available: http://dx.doi.org/10.1002/wics.50

[121] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures," in *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*. IEEE, 2013, pp. 108–112.

[122] L. Tucker, "Introduction to cloud computing for startups and developers, sun microsystems," 2009.

[123] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation and signal processing," in *Advances in neural information processing systems*, 1997, pp. 281–287.

[124] E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos, and M. G. Karlaftis, "Exploiting new sensor technologies for real-time parking prediction in urban areas," in *Transportation Research Board 93rd Annual Meeting Compendium of Papers*, 2014, pp. 14–1673.

[125] J. Weppner and P. Lukowicz, "Bluetooth based collaborative crowd density estimation with mobile phones," in *Pervasive computing and communications (PerCom), 2013 IEEE international conference on*. IEEE, 2013, pp. 193–200.

[126] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications*

*(AINA), 2010 24th IEEE International Conference on.* IEEE, 2010, pp. 446–452.

[127] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler, "More than bin packing: Dynamic resource allocation strategies in cloud data centers," *Information Systems*, vol. 52, pp. 83–95, 2015.

[128] G. Yan, W. Yang, D. B. Rawat, and S. Olariu, "Smartparking: A secure and intelligent parking system," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 18–30, 2011.

[129] Z. Yang, H. Liu, and X. Wang, "The research on the key technologies for improving efficiency of parking guidance system," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 2. IEEE, 2003, pp. 1177–1182.

[130] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in *Green Computing COnference, 2010 International*, pp. 357–364.

[131] F. Zhang, J. Wu, and Z. Lu, "Psrps: A workload pattern sensitive resource provisioning scheme for cloud systems," in *Services Computing (SCC), 2013 IEEE International Conference on.* IEEE, 2013, pp. 344–351.

[132] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[133] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[134] W.-X. Zhu and E.-X. Chi, "Analysis of generalized optimal current lattice model for traffic flow," *International Journal of Modern Physics C*, vol. 19, no. 05, pp. 727–739, 2008.