Resource allocation in Cloud federation

Salma Rebai

THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité: Informatique et Télécommunications

Ecole doctorale: Informatique, Télécommunications et Electronique de Paris

Presentée par

Salma REBAI

Pour obtenir le grade de

DOCTEUR DE TELECOM SUDPARIS

Allocation et fédération des ressources informatiques dans le Cloud

---

Soutenue le 13 Mars 2017

devant le jury composé de:

Prof. Samir TOHMÉ	Rapporteur	Université de Versailles
Prof. Jalel BEN OTHMAN	Rapporteur	Université Paris 13
Prof. Marcelo DIAS DE AMORIM	Examinateur	UPMC – Paris 6
Prof. Véronique VÈQUE	Examinateur	Université Paris-Sud
Prof. Nadjib AIT SAADI	Examinateur	Université Paris-Est
Dr. José NÈTO	Examinateur	Telecom SudParis
Prof. Djamal ZEGHLACHE	Directeur de thèse	Telecom SudParis

Thèse n° 2017TELE0006

JOINT PHD THESIS BETWEEN TELECOM SUDPARIS AND UNIVERSITY OF PIERRE ET MARIE CURIE

Speciality: Informatics and Telecommunications

Doctoral School: Informatique, Télécommunications et Electronique de Paris

Presented by

Salma REBAI

To obtain the degree of

DOCTOR OF TELECOM SUDPARIS

Resource allocation in Cloud federation

Defended on 13 March 2017

Jury Members:

Prof. Samir TOHMÉ	Reporter	University of Versailles
Prof. Jalel BEN OTHMAN	Reporter	University of Paris 13
Prof. Marcelo DIAS DE AMORIM	Examiner	UPMC – Paris 6
Prof. Véronique VÈQUE	Examiner	University of Paris-Sud
Prof. Nadjib AIT SAADI	Examiner	University of Paris-Est
Dr. José NÈTO	Examiner	Telecom SudParis
Prof. Djamal ZEGHLACHE	Thesis Director	Telecom SudParis

Thesis n° 2017TELE0006

To my parents Fouzia and Zouhir,

I am particularly indebted for your endless love, unconditional trust and continuous support. Thanks for always believing in me and being at my side in everything I do!

To my dear husband Wael,

I am especially thankful for your understanding, encouragement, infinite support and sincere love. Thanks for everything!

To my sisters Imen and Amal, and my brother Rami,

Thank you for always standing by my side during difficult times and for the fun moments I have shared with you!

To all REBAI, ZOUAOUI and JRIBI family members,

Thanks for your love, kind support and encouragement!

Salma Rebai

Abstract

Cloud computing is a steadily maturing large-scale model for providing on-demand IT resources on a pay-as-you-go basis. This emerging paradigm has rapidly revolutionized the IT industry and enabled new service delivery trends, including infrastructure externalization to large third-party providers. The Cloud multi-tenancy architecture raises several management challenges for all stakeholders. Despite the increasing attention on this topic, most efforts have been focused on user-centric solutions, and unfortunately much less on the difficulties encountered by Cloud providers in improving their business.

In this context, Cloud Federation has been recently suggested as a key solution to the increasing and variable workloads. Providers having complementary resource requirements over time can collaborate and share their respective infrastructures, to dynamically adjust their hosting capacities in response to users' demands. However, joining a federation makes the resource allocation more complex, since providers have to also deal with cooperation decisions and workload distribution within the federation. This is of crucial importance for cloud providers from a profit standpoint and especially challenging in a federation involving multiple providers and distributed resources and applications.

This thesis addresses profit optimization through federating and allocating resources amongst multiple infrastructure providers. The work investigates the key challenges and opportunities related to revenue maximization in Cloud federation, and defines efficient strategies to govern providers' cooperation decisions. The goal is to provide algorithms to automate the selection of cost-effective distributed allocation plans that simultaneously satisfy user demand and networking requirements. We seek generic and robust models able to meet the new trends in Cloud services and handle both simple and complex requests, ranging from standalone VMs to composite services requiring the provisioning of distributed and connected resources.

In line with the thesis objectives, we first provide a survey of prior work on infrastructure resource provisioning in Cloud environments. The analysis mainly focuses on profit-driven allocation models in Cloud federations and the associated gaps and challenges with emphasis on pricing and networking issues. Then, we present a novel exact integer linear program (ILP), to assist IaaS providers in their cooperation decisions, through optimal "insourcing", "outsourcing" and local allocation operations. The different allocation decisions are treated jointly in a global optimization formulation that

splits resource request graphs across federation members while satisfying communication requirements between request subsets. In addition to the request topology, this partitioning takes into account the dynamic prices and quotas proposed by federation members as well as the costs of resources and their networking. The algorithm performance evaluation and the identified benefits confirm the relevance of resource federation in improving providers' profits and shed light into the most favorable conditions to join or build a federation. Finally, a new topology-aware allocation heuristic is proposed to improve convergence times with large-scale problem instances. The proposed approach uses a Gomory-Hu tree based clustering algorithm for request graphs partitioning, and a Best-Fit matching strategy for subgraphs placement and allocation. Combining both techniques captures the essence of the optimization problem and meets the objectives, while speeding up convergence to near-optimal solutions by several orders of magnitude.

keywords: *Cloud federation, profit optimization, distributed allocation, request splitting, linear integer programming, Graph decomposition, Gomory-Hu tree, Best-Fit matching.*

Acknowledgements

It is a pleasure to thank and convey my most profound gratitude to all those people who have contributed in one way or another to the achievement of this work.

I would like to express my deep gratitude and sincere thanks to my supervisor and thesis director, Prof. *Djamal ZEGHLACHE*, for welcoming me in his research group at Telecom SudParis and for his continuous support and guidance during my PhD study years.

I am very grateful to my reading committee members, Prof. *Samir TOHMÉ* and Prof. *Jalel BEN OTHMAN*, for accepting to judge this work. Thank you for your precious time, your interest, and your valuable feedback and suggestions to improve my dissertation work. My sincere thanks go also to the other members of my defense committee, Prof. *Véronique VÈQUE*, Prof. *Nadjib AIT SAADI*, Prof. *Marcelo DIAS DE AMORIM* and Dr. *José NÈTO*, for their interest and valuable comments and for being part of my thesis jury.

I extend heartfelt thanks to my friends and colleagues at Telecom SudParis for their encouragements and support and for the fun moments we spent together. A special acknowledgement is necessary for the administrative staff and especially the department's assistant for their continuous effort to facilitate administrative procedures. My warmest thanks go also to my colleagues at ESME Sudria for the excellent and truly enjoyable ambiance. I am very thankful for their encouragements and valuable advices whenever I was in need.

Last but not least, my endless and deepest appreciations go to my family members: my loving parents, my dearest husband, my caring brother and sisters, to whom I owe so much. Thanks for making my life beautiful and for supporting me throughout my thesis!

Contents

List of Figures

List of Tables

Glossary of Acronyms

API	**A**pplication **P**rogramming **I**nterface
CAGR	**C**ompound **A**nnual **G**rowth **R**ate
CPU	**C**entral **P**rocessing **U**nit
EC2	**E**lastic **C**ompute **C**loud
GH	**G**omory **H**u
GICTF	**G**lobal **I**nter-**C**loud **T**echnology **F**orum
IaaS	**I**nfrastructure-**a**s-**a**-**S**ervice
IDC	**I**nternational **D**ata **C**orporation
ILP	**I**nteger **L**inear **P**rogram
IT	**I**nformation **T**echnology
KVM	**K**ernel based **V**irtual **M**achine
LXC	**L**inu**X** **C**ontainers
NIC	**N**etwork **I**nterface **C**ontroller/**C**ard
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
OCCI	**O**pen **C**loud **C**omputing **I**nterface
OS	**O**perating **S**ystem
PaaS	**P**latform-**a**s-**a**-**S**ervice
PM	**P**hysical **M**achine
QoS	**Q**uality **o**f **S**ervice
SaaS	**S**oftware-**a**s-**a**-**S**ervice
SLA	**S**ervice **L**evel **A**greement
VA	**V**irtual **A**ppliance
VDC	**V**irtual **D**ata **C**enter
VM	**V**irtual **M**achine
VMM	**V**irtual **M**achine **M**onitor

Chapter 1

Introduction

Contents

1.1 Scientific Context

Cloud Computing [8–10] is a steadily maturing model for providing on-demand IT resources as a service over the Internet. This new computing paradigm, emerged initially as a solution for hosting large-scale online applications (e.g. social networking, web search and video gaming), has rapidly revolutionized the IT industry and enabled new trends of delivering, managing and consuming IT capabilities. With the rapid evolution of Internet and virtualization technologies and the support of Leader IT companies, the long-held dream of "Computing as utility" has finally come true and Cloud Computing has become one of the fastest growing fields in IT. According to a new forecast from Cisco [11], more than 86% of workloads will be processed in Cloud data centers by 2019. Likewise, IDC (International Data Corporation) predicts that spending on public Cloud services will exceed the $127 billion in 2018 compared to $56.6 billion spent in 2014, as shown in Figure 1.1. This represents a compound annual growth rate (CAGR) close to 23%, which is about six times the growth rate of the overall IT market [1].

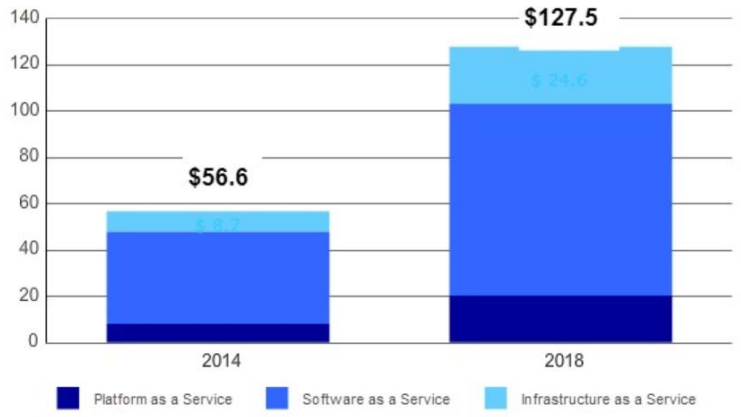


FIGURE 1.1: The IDC's forecasts on worldwide IT cloud services spending in billion dollars [1]

This increasing popularity of Cloud services is due to their flexibility in enabling access to resources and applications from anywhere and at anytime on a "pay-as-you-go" basis. This allows customers to avoid upfront investments for hardware acquisition and maintenance, while benefiting from increased resource availability and improved fault-tolerance capabilities. Among different Cloud delivery models, the Infrastructure as a Service (IaaS) allows users to outsource their infrastructures to third-party providers, offering on-demand access to an elastic pool of virtualized compute, network and storage resources. The IaaS services are typically delivered to users as Virtual Machine (VM) instances with different resource configurations and QoS guarantees. According to a recent forecast [1], IDC recognized the IaaS model as one of the fastest growing Cloud service categories, with an expected revenue of $24.6 billion in 2018 and a CAGR rate of 31% from 2014 to 2018 (Figure 1.1). This thesis is centered around this promising technology and addresses the related resource management challenges.

With the rapid growth of Cloud services, the definition of efficient management strategies has become a major concern for Cloud actors and has attracted significant attention in recent years. Most related works have been focused on user-centric solutions analyzing the functional and economic benefits of using Cloud services. Less attention, however, has been paid to the opportunities and challenges encountered by Cloud vendors to improve their profits and remain in business. This is of paramount importance for Cloud providers, who endlessly need efficient solutions to reduce their operational costs and maximize their revenues. Even if the multi-tenant cloud model enables providers to increase their hosting capacity by sharing their infrastructure among multiple users, they still need effective management policies to handle the complexity of Cloud systems and better meet user requirements. This includes the optimization of resource allocation and placement decisions, which is the focus of this thesis.

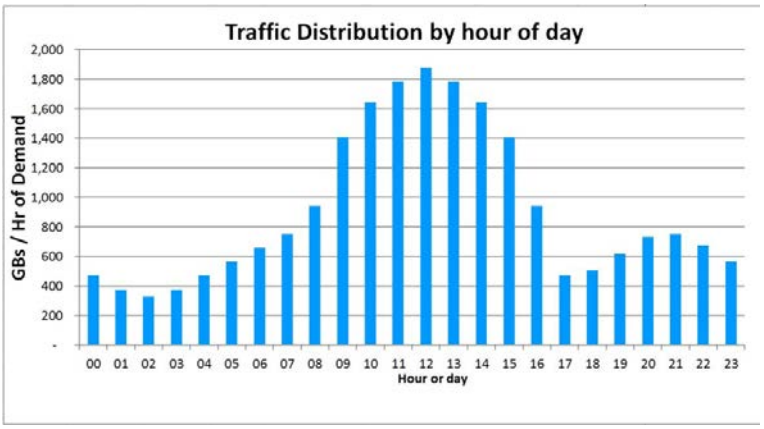


FIGURE 1.2: A daily demand distribution of a typical Internet application [2]

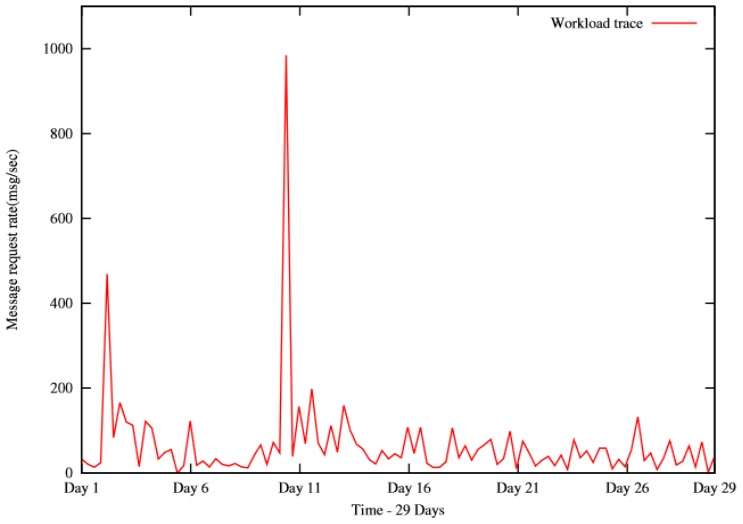
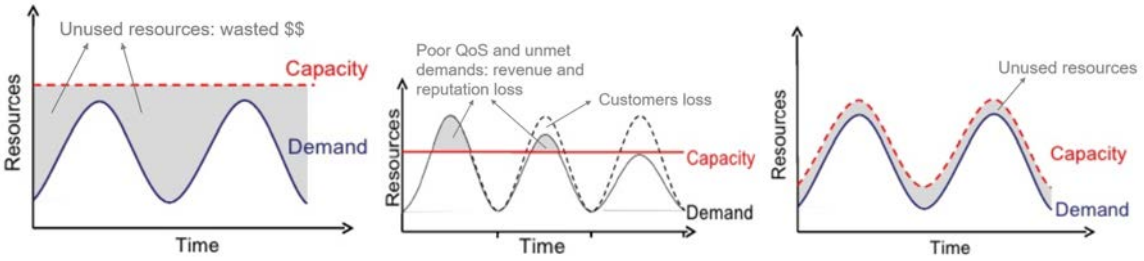


FIGURE 1.3: Google cluster workload traces of May 2011 [3]

The resource allocation problem is a recurring issue in distributed computing. The growing scale of Cloud computing and the increasing complexity of users' requirements introduce additional constraints and make allocation decisions more challenging with difficult tradeoffs between user satisfaction and profit maximization. In fact, IaaS providers are faced with stochastic request arrivals and departures, which generates highly heterogeneous and time-varying workloads. Moreover, the analysis of real workload traces has shown that user demands experience seasonal fluctuations with random bursts of up to 20 times the usual load, as illustrated in Figures 1.2 [2] and 1.3 [3]. Given these constraints, the long-term resource capacity planning becomes problematic [12]. Traditional allocation solutions based on static resource provisioning lead to poor performance and hinder providers from achieving expected profits. In fact, over-provisioning resources to meet potential demand peaks can result in significant costs and unused capacities as depicted in Figure 1.4-(a). In contrast, planning resources for only usual workloads

may lead to request rejection and QoS degradation in overload, which both reduce the provider's reputation and revenue as shown in Figure 1.4-(b). To avoid such issues, IaaS providers must be able to dynamically adjust their hosting capacity in response to demand fluctuations as in Figure 1.4-(c). This emphasizes the need for richer allocation mechanisms to help providers achieve better profits.



(a) Static capacity: over-provisioning

(b) Static capacity: under-provisioning

(c) Dynamic capacity adjustment

FIGURE 1.4: Static vs. Dynamic infrastructure resources provisioning.

To address these limitations, *"Cloud Federation"* has recently been introduced as a key solution to build efficient and profitable Cloud business. A Federation is a particular scenario of *inter-Clouds* [13, 14], where several providers can voluntarily form a partnership and share their resources to meet users' demands and requirements. This mutual resource sharing can improve the availability, cost-efficiency and QoS guarantees of Cloud services. This also enables new business opportunities through multi-site service provisioning. Such functional and financial benefits have motivated the evolution of the Cloud market from large "Monolithic" vendors to interoperable federations of small and medium providers, who cooperate to meet each other's resource needs (Business-2-Business).

The work carried out in this thesis is related to this context and is focusing on Cloud resource federation among multiple IaaS providers, with the aim to maximize their revenues. The goal is to provide novel and cost-effective allocation algorithms to optimize the cooperation decisions within a federation in response to market conditions. The rest of this chapter summarizes the main aspects of our research work, and is organized as follows. In section 1.2, we present the problem statement and motivations behind the work, the addressed research issues and the thesis objectives. Section 1.3 outlines the major scientific contributions of this dissertation and section 1.4 presents the organization of the thesis.

1.2 Research Problem and Objectives

1.2.1 Motivations and Problem Statement

The dynamic and uncertain nature of Cloud environments makes the resource allocation problem hard to solve. This difficulty increases with the sizes of service requests and hosting infrastructures. To address such problem, providers should be able to dynamically scale their hosting capacity in response to demand fluctuations. In fact, even if cloud computing promises on-demand access to "unlimited resources", there would always be an upper bound on hardware and network capacity within a data-center, which may lead to resource exhaustion and performance degradation during demand spikes. Moreover, as computational services are non-storable, unused resources generate a revenue loss that cannot be recovered later. To improve revenues, providers should optimize their resource utilization and achieve higher acceptance rates. However, existing allocation mechanisms are limited to static capacities and poor auto-scaling policies, which do not allow providers to efficiently manage unpredictable traffic bursts. Therefore, current solutions need to evolve beyond the simple allocation of local resources to offer flexible and seamless scalable hosting infrastructures.

To deal with these issues, *Cloud Federation* has been proposed as a key solution to random bursts in user demands. Providers having complementary resource requirements over time can collaborate and share their respective resources to dynamically adjust their hosting capacities in response to their workloads. Such collaboration empowers providers to overcome resource limitation and deliver advanced services with improved performance, availability and QoS guarantees. Figure 1.5 illustrates the cooperation aspects within a federation, namely the *"Insourcing"* and *"Outsourcing"* of virtual resources. During demand spikes, providers may *"Outsource"* part of their incoming loads to other members, by *"borrowing"* unused resources from foreign Clouds to get additional capacities. This gives providers the illusion of infinite resources and results in fewer requests rejection. In case of low demands, providers can avoid wasting resources by *"renting"* part of their idle capacities to serve *"Insourcing"* requests from other members. Beyond this collaboration, the federation members remain totally independent and may use different allocation and pricing strategies to operate their own infrastructures.

Joining a Federation brings many business opportunities for IaaS providers, including advanced service offerings, reliable multi-site deployment and service cost minimization. Among the different incentives of this emerging paradigm, we focus on its economical and financial benefits as solution to enhance providers' profitability. If used efficiently, Insourcing and Outsourcing resources can help providers alleviate the problem of load variability and meet specific requirements about geographic locations and access latency

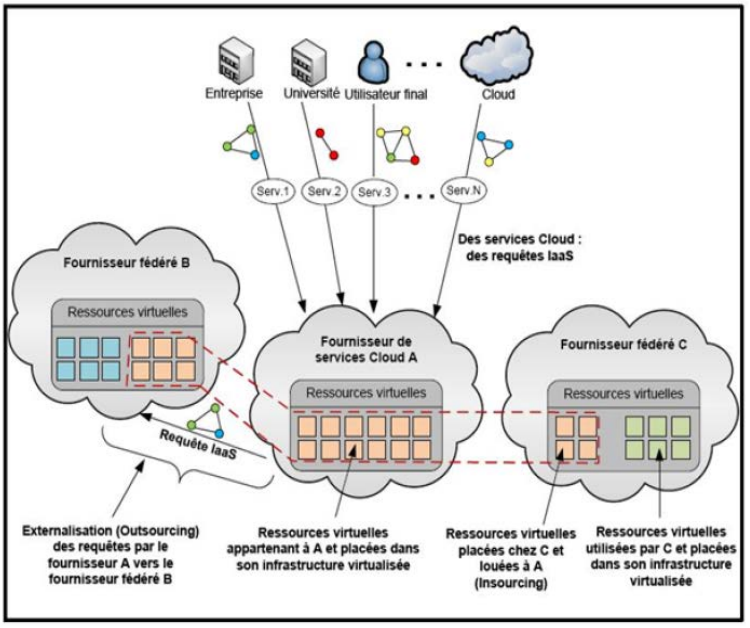


FIGURE 1.5: Insourcing and Outsourcing resources within a federation.

of users' applications. However, being part of a federation raises new resource allocation challenges since providers have to also deal with cooperation level optimization (workload distribution, insourcing and outsourcing operations). The increasing number of actors and the diversity of service offerings within the federation make the allocation task particularly complex to handle, since the number of metrics and key performance indicators can be high. The definition of efficient resource allocation and sharing strategies is and will remain a real challenge for a while.

This issue has recently attracted significant attention from the research community. Prior works have mainly focused on VM placement and servers consolidation in Single-Cloud environments, but unfortunately much less on distributed multi-Cloud scenarios. However, with the progress and popularity of Cloud offerings, customers are becoming more demanding in terms of quality and range of services, which is hard to be satisfied by isolated Clouds. To fulfil complex requirements, providers are inclined to collaborate and form partnerships for mutual benefits and resource sharing. More attention should be given to federated Clouds to meet these new business trends. Related state-of-the-art solutions have been centered on the definition of platforms and architectures for interoperability and interactions between providers, but much less on the problem of workload management within a federation. This is of crucial importance for cloud providers from a business value and profit standpoint and especially challenging in a federation involving multiple providers and heterogeneous distributed resources. Innovative allocation algorithms and techniques are required to help providers address current barriers and

support large-scale applications with advanced QoS requirements.

Our work focuses on this optimization problem of federating and optimally allocating distributed resources amongst multiple infrastructure providers, with respect to profit maximization. The problem consists in finding, for each incoming service request, the optimal resource aggregation that leads to the best cost/performance tradeoff from the users and system point of view. The selected allocation plan should achieve the maximum profit according to the federation offerings, while satisfying users' demands and requirements.

Analyzing the field of infrastructure resource provisioning from one or multiple providers, we noticed that current research handles primarily the allocation of individual VMs to consumers and ignores the internal structure of requested services. This leads to suboptimal solutions and service performance degradation, especially in case of multi-tier applications involving distributed and networked resources. Allocation mechanisms must evolve to support composite services and meet stringent networking requirements. This issue is at the center of this thesis research, which aims to address complex service requests requiring the provisioning of multiple connected VMs according to a specific network topology. To our knowledge, previous work on profit-driven allocation models does not incorporate networking costs between the federation members. Making the system aware of the communication requirements between service components and the costs of their networking, may significantly improve performance.

Resource pricing is another important aspect that should be considered in our study, since it directly affects the efficiency of the allocation strategy and achieved profits. Current Cloud market is mainly based on fixed pricing for service billing. Nevertheless, recent studies have revealed that traditional flat-rate pricing can lead to ineffective performances due to the mismatch between demand fluctuations and resource availability. To improve their business, several providers are resorting to new pricing strategies based on dynamic price adjustment according to supply and demand conditions. To obtain accurate results, we believe it is important to respect this variety of pricing schemes when solving the resource allocation problem.

Adopting the previously cited aspects makes the profit optimization in Cloud federations more challenging. Finding the optimal distributed resource allocation plan becomes more complicated since providers need to involve networking QoS parameters and pricing information in the selection procedure. Smart placement solutions are required to automate the resource assignment for tenants' applications according to requests requirements and federation conditions. This thesis addresses the problem with its different facets and dimensions to provide a generic allocation approach.

1.2.2 Research Questions and Objectives

In line with the scope of the thesis, we have identified the following research questions that have obviously driven the investigations of this doctoral work:

- How to support the heterogeneity of users' demands? The user expressed requirements in terms of computing resources and network topology have to be embedded in the model to achieve better performance and optimal request partitioning across providers. We seek generic allocation models able to support both basic standalone virtual machines and complex services with several elementary components.
- How should providers exploit available resources to optimally distribute their incoming load across the federation? This question addresses the provider's allocation policy to simultaneously satisfy its business goals and users' requirements. Federation members have to decide about several conflicting allocation alternatives, including when and where to outsource service requests, how to partition requests across providers, how many resources to allocate from each provider to achieve minimum costs, when and to what extent to contribute resources to the federation, and how to identify requests leading to less profit and those improving revenues. These are some of the questions the thesis is attempting to answer.
- How to evaluate the proposed allocation algorithms? The evaluation of allocation policies on a real Cloud federation is a major challenge for researchers. The assessment of this complex multi-Cloud scenario requires the implication of several providers with heterogeneous platforms and services, which is too expensive to be conducted. A common cost-effective solution is to resort to Cloud simulation frameworks [15] that enable reproducible experiments with various evaluation parameters and scenarios. However, existing tools provide limited support for federated Clouds and their use requires additional extensions and development work. Moreover, simulation experiments should be as realistic as possible to get convincing results and be certain of the model's applicability in real Cloud environments. Lastly, due to privacy and security reasons, there is no publicly available Cloud workload traces. Realistic workloads should be generated to feed the simulation experiments.

Driven by the above research problems, the thesis focuses on the design and development of resource allocation algorithms to help federated providers make profitable cooperation decisions. The objective is to investigate the challenges and opportunities related to resource sharing in cloud federations, and to define efficient allocation policies for workload distribution across federated infrastructures. A key step of our thesis work is

the review of related literature to gain a clear understanding of existing approaches and identify the key parameters to consider for the problem modeling. We aim to provide novel exact and heuristic algorithms advancing the state-of-the-art and considering new constraints and criteria often neglected in the past. Different approaches, ranging from combinatorial optimization to graph theory and simple heuristics, are explored and compared in terms of performance and scalability to identify the most favorable conditions for profit improvements. The proposed algorithms should be generic enough to deal with the new trends in Cloud and to support large-scale workloads with heterogeneous requirements and performance objectives.

1.3 Thesis Contributions

With respect to the defined objectives, this thesis brings the following key contributions:

1. A survey of the state-of-the-art solutions for profit-driven resource allocation in federated Clouds. The analysis allowed us to identify the relevant parameters and criteria to consider in our optimization model, including the providers' workloads, insourcing prices variation, resource and networking costs, providers' reputation, requests sizes and connectivity, etc. These parameters are studied in terms of impact on the federation profitability to shed additional light on this matter.

2. A novel exact algorithm for optimal request partitioning and allocation in distributed Cloud federations. The model is formulated as an integer linear program (ILP) that maximizes providers' profit and user satisfaction through insourcing and outsourcing resources. Based on a generic graph modeling of tenants' demands, the proposed approach can handle both simple and complex requests ranging from standalone VMs to composite services with connected elementary components. In addition to profit optimization, the algorithm minimizes both requests rejection and networking costs imposed by the desired VMs connectivity. All allocation decisions are treated jointly in a global objective function that takes into account the prices and quotas proposed by the federation and the costs of resources and their networking, to optimally split received requests across providers. A custom discrete-event simulator, using synthetic workloads generated according to stochastic models from the literature, was implemented to assess the algorithm performance. The results are reported with respect to profit improvements, requests acceptance rates, convergence times and scalability. The evaluation results show the algorithm efficiency in improving profits and user satisfaction and shed light into the most favorable conditions to join or build a federation.

3. A topology-aware heuristic allocation algorithm is proposed to handle large-scale federations and increasing request graph sizes and connectivity. The heuristic uses a Gomory-Hu tree based clustering algorithm for request decomposition into weakly connected subgraphs, which are distributed across the federation according to a Best-Fit strategy. Combining both techniques captures the essence of the optimization problem and meets the defined objectives in terms of profit and acceptance rate maximization, while respecting networking costs and requirements. A thorough evaluation and comparison of the heuristic and exact solutions have shown the efficiency of the proposed algorithm to scale with problem size and to achieve near-optimal solutions. The heuristic leads to small gaps in profit improvements compared to the ILP model (ranging in $[2\%; 10\%]$ in worst cases), while improving convergence times by several orders of magnitude.

1.4 Thesis Organization

This dissertation is organized into six core chapters. Besides the present chapter introducing the context, objectives and contributions of the thesis, the manuscript is organized as follows:

Chapter 2 provides the background information related to this thesis. It presents an overview of Cloud Computing including features, service delivery models, and prevalent resource pricing schemes. The chapter also introduces the inter-Cloud paradigm promising new business opportunities and better performance, with emphasis on Cloud federations and related economic and management challenges.

Chapter 3 investigates the problem of resource allocation in Cloud environments and provides a detailed review of the literature on profit-driven allocation strategies. It also discusses the related pricing and networking issues in cloud federations and presents some dynamic pricing models suitable for the studied scenario.

Chapter 4 introduces an integer linear program for profit optimization in Cloud federations. The federation system model is presented in terms of assumptions, users' requests modeling and pricing schemes used to derive the ILP formulation for distributed resource allocation. The chapter then describes the simulation experiments and provides the performance evaluation results about the algorithm effectiveness and favorable federation conditions.

Chapter 5 presents a novel topology-aware heuristic allocation algorithm to address the complexity of the exact model with large-scale instances. The heuristic is based on Gomory-Hu transformation and Best-Fit allocation strategy to speed up convergence

times to optimal and near optimal solutions. A comparative performance evaluation with the exact model is reported to assess the efficiency and scalability of the proposed algorithm.

Finally, Chapter 6 concludes the thesis with a summary of main contributions and findings and provides insights into future research directions.

For the sake of accessibility, we also provide in Appendix A a French summary of the thesis contributions.

Chapter 2

Background and Foundations

Contents

2.1 Introduction

The *Cloud Computing* is an emerging concept for on-demand resource provisioning, promising relevant and cost-effective IT solutions. While the economic benefits for Cloud customers have been extensively discussed, less attention has been paid to the challenges faced by service providers to ensure profitable business in such a competitive market. Cloud providers require novel methods for efficient resource allocation and management to reduce their costs and improve their profits. Among different potential solutions

for achieving such objectives, we investigate the cooperation and federation between providers. Before addressing this problem, it is crucial to study it in depth and identify its drivers and barriers.

This chapter introduces background information on the basic concepts related to our research topic. We first present in section 2.2 an overview of Cloud Computing describing its key features and enabling technologies, notably the virtualization technique. Afterwards, we investigate in section 2.3 the *Cloud Federation* concept which is the target platform addressed in this thesis. We broadly discuss the challenges and benefits of inter-Cloud model addressing the limitations of traditional Cloud architectures. Then, we narrow down our focus on Cloud Federation to study its main economic drivers and challenges, including the resource sharing and allocation problem. In section 2.4, we give a short overview on common resource pricing schemes used in the Cloud market today. Finally, we conclude with section 2.5 that gives a summary of the orientation and scope of this thesis.

2.2 Cloud Computing Overview

Cloud Computing [8] [9] [10] is a steadily maturing large-scale model for providing on-demand IT resources (compute, storage, networks, platforms and applications) as a service over the Internet. With the evolution of virtualization, high-speed Internet access and especially the support of leader IT companies, the long-envisioned dream of "computing as utility" has been achieved and Cloud Computing has become one of the fastest growing fields in the IT industry [1, 8, 11, 16]. This increasing attractiveness of Cloud results from its efficiency and flexibility, enabling customers to rapidly provision and access resources from anywhere and at any-time on a pay-per-use basis. Cloud Computing allows its users to avoid the installation and management efforts by externalizing their hardware and software resources to a large-scale environment promoting high availability and reduced costs.

Understanding the main characteristics of Cloud services, its advantages and limitations, is crucial for cloud actors to make appropriate decisions and get full benefits of this technology. This is the focus of this section that surveys the main aspects of Cloud paradigm.

2.2.1 Cloud definition and key features

"Cloud computing" has become the 21st century IT buzzword, that nearly everyone has heard about, but much less truly understand what it is and what are its economic

benefits. Although various Cloud definitions have been proposed in both academia and IT industry [17] [18] [19], there is still no consensus on a precise and complete definition for this evolving paradigm. The most accepted definition is that provided by the U.S National Institute of Standards and Technology (NIST) in [20]:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

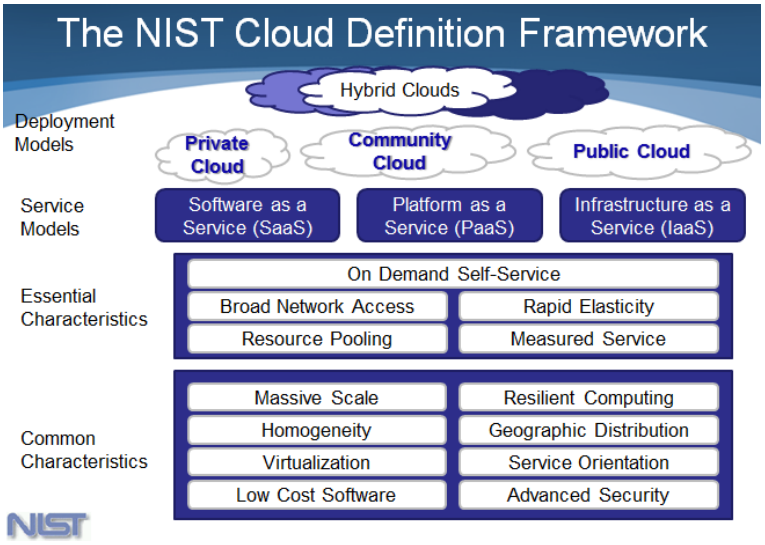


FIGURE 2.1: The NIST Definition of Cloud Computing [4]

This definition, shown in figure 2.1, covers the important concepts enabling the understanding of the Cloud Computing terminology, including cloud service types and deployment models. In particular, the NIST definition highlights the key features discerning Cloud offerings from other traditional IT services, as detailed below:

- **On-demand self-service:** Consumers are able to automatically provision IT resources at any time, in a simple and flexible way through a web-based management interface.
- **Broad network access:** Cloud resources are remotely accessible over the network through standard mechanisms supporting heterogeneous client platforms such as mobile devices and workstations.
- **Resource pooling:** The cloud provider pools its resources to serve multiple customers using a multi-tenancy architecture based on virtualization technologies.

Shared resources are dynamically assigned and reassigned to consumers, who have no control or knowledge about the exact physical location of delivered resources.

- **Rapid elasticity:** Cloud resources can rapidly scale up and down to cope with workload variations. The resources are allocated and released immediately on-request or automatically by customizable triggers to fit users demand and performance requirements without disturbing the running tasks. This gives the illusion of infinite resources available on-demand.

- **Measured service:** Cloud providers possess appropriate accounting mechanisms to measure the resource usage for each tenant. "Metered" resources are monitored, controlled, accounted and transparently reported, to enable "pay-per-use" billing and capacity optimization.

With the myriad definitions of Cloud Computing, understanding the above features is fundamental to avoid the widespread confusion between cloud solutions and other IT services. This helps consumers fairly evaluate the cloud services based on their price-performance values to take better advantage of the promised economic and functional benefits. Figure 2.1 lists also eight additional *"Common Characteristics"*, that can help customers prioritize important features for their needs. Cloud common characteristics include *"massive scale"*, *"homogeneity"*, use of *"virtualization"*, *"Low Cost software"* due to multi-tenancy model, *"resilient computing"* ensuring fault tolerance and disaster recovery, *"Geographic distribution"*, *"service orientation"* and *"advanced security"*.

According to NIST [21], the Cloud ecosystem involves five major actors that have distinct roles and interactions: *Cloud provider* is the entity operating and managing the cloud infrastructure and it is responsible for handling users' requests. *Cloud consumer* is a person or an organization that uses this service. The *Cloud broker* is an intermediary that may be solicited by the consumer to negotiate the service on his behalf. The *Cloud carrier* manages the connectivity and routing of services between providers and consumers. Finally the *Cloud auditor*, is an independent party that can assess the service performance to verify its conformance to standards.

Furthermore, the NIST definition [20] classifies Cloud services into three delivery models (i.e. Software as a Service "SaaS", Platform as a Service "PaaS", Infrastructure as a Service "IasS") based on the type of provided resources, and identifies four possible deployment models (i.e. private cloud, community cloud, public cloud, hybrid cloud) depending on the ownership and usage scope of the Cloud Infrastructure. Cloud services and deployment models will be described in details in section 2.2.3.

2.2.2 Virtualization and Cloud Computing

The Virtualization technology is the main foundation of Cloud Computing offerings [22]. This concept refers to the set of hardware and software tools enabling the abstraction of a physical resource into several logical units, that can be used separately by different Operating Systems (OS) and applications [23]. Resources Virtualization is a key feature for Cloud providers to build efficient, flexible and cost-effective computing models satisfying the Cloud market challenges. It enables simple resource management and dynamic resource resizing, reduced hardware costs due to resource sharing, isolation and fairness between tenants, increased availability and quick recovery through easy backups and rapid migrations [24, 25].

The virtualization can be implemented in various levels by using different methods. We distinguish three major virtualization forms namely the *"server virtualization"*, *"network virtualization"* and *"storage virtualization"*, which are all based on the concept of hardware abstraction and sharing. "Storage virtualization" enables the access to virtual disks independently of the data location and its mapping into the hard storage device. "Network virtualization" refers to the creation of isolated virtual networks overlaid on the same physical infrastructure and sharing the available bandwidth. A virtual network can combine multiple network resources and functionalities such as virtual NICs or logical switches and routers. Finally the "server virtualization" allows the consolidation of multiple isolated virtual servers into a single physical one. In this thesis, we focus on server virtualization, which is the most common technique used in Cloud systems.

2.2.2.1 Server Virtualization

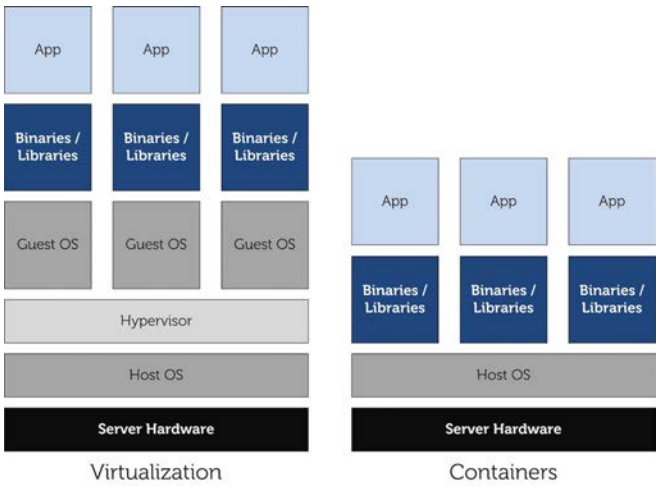


FIGURE 2.2: Hypervisor-based vs. Container-based virtualization.

Server virtualization enables to host on the same server multiple independent virtual units running their own OS and applications. This kind of virtualization can be implemented using different methods, classified into two main categories based on the abstraction layer: hypervisor-based and container-based virtualization, as illustrated in Figure 2.2.

Hypervisor-based virtualization: is the most popular virtualization approach and is based on a thin software layer running on top of the operating system, called Hypervisor or Virtual Machine Monitor (VMM)[26]. The hypervisor is responsible for the server resource management to provide necessary utilities to the virtual guests (Virtual Machines) running on top of it. Each guest VM can run a different OS and is fully isolated from both hosting OS and other VMs. This technology can be implemented in different manners including full-virtualization, para-virtualization and hardware assisted virtualization [27], which differ in the way the host and guest OSs are modified to support virtualization and interact with each others. Well-known hypervisors include KVM [28] and VMWare ESXi [29] as full-virtualization solutions, Xen [30] for paravirtualization implementation and Hyper-V [31] for hardware-assisted virtualization.

Container-based virtualization: also known as OS-level virtualization, is a lightweight alternative to run multiple virtual servers without requiring additional layer for hardware virtualization [32–34]. In this approach, an application running within the host OS manages the virtualization and isolation between the virtual servers. All guest instances, called containers, share the same underlying OS but have their own processes as depicted in Figure 2.2. Container-based virtualization is supposed to have weaker isolation compared to hypervisor-based approach. Examples of container-based solutions include Linux containers (LXC) [35], Docker [36] and OpenVZ [37].

Although both categories allow VMs isolation, each one has its assets and utilization cases depending on the hardware and workload characteristics [38]. Hypervisor-based approach is more suitable when different OSs or high security levels are needed. Even with the additional communications overhead introduced by the abstraction layer, this approach ensures high efficiency through heterogenous OS consolidation [39, 40], and high flexibility due to live VM migration [41]. The container-based virtualization is a good alternative offering near-native performance since there is no overhead for hardware device emulation. The drawback of this type is that guest containers depend on the hosting OS kernel, which makes portability more complex.

In Cloud Computing, container-based virtualization is commonly used for building lightweight PaaS environments while hypervisors are suitable for building IaaS services.

Cloud services can be encapsulated into virtual appliances (VAs) to be deployed using VM instantiation whenever needed [42], which allows economies of scale. The thesis work assumes hypervisor-based virtualization and provides VM-based algorithms for resource allocation in distributed Clouds. Through the rest of this thesis, and unless specified, the term virtualization refers to this type.

2.2.3 Cloud Services and Deployment Models

Cloud Computing brings new business opportunities and offers a wide variety of IT solutions. There are three basic services commonly associated with Cloud referred as *SPI* model: the **S**aaS, **P**aaS and **I**aaS services [20, 43], that differ in the resource types made available to users. Cloud Computing architecture is often represented as a stack of these three abstract layers (Infrastructure, platform and application), where each layer can offer its resources as a service to the upper layers, as depicted in Figure 2.3.

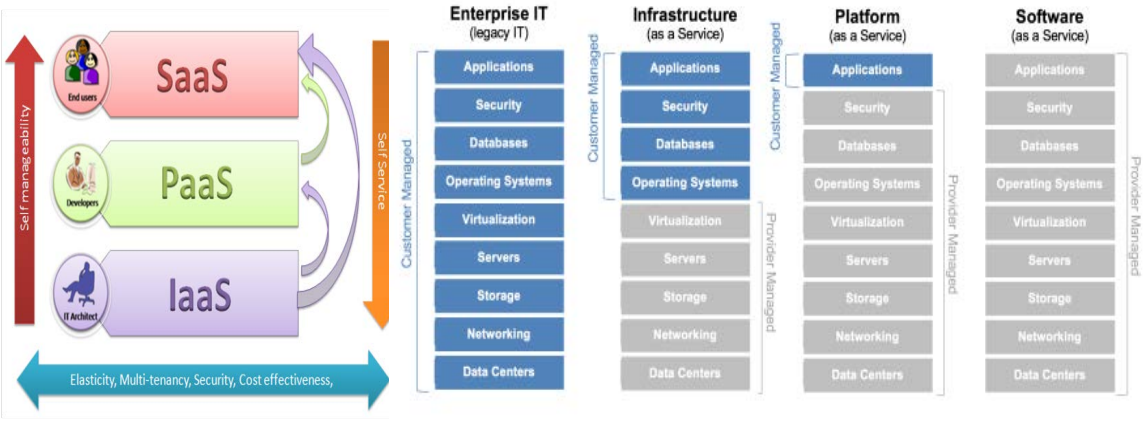


FIGURE 2.3: Cloud Computing Services models.

Software as a Service (SaaS): SaaS is the highest level of the Cloud stack that delivers complete applications to consumers through the internet. The SaaS provider is responsible for hosting, managing and controlling the application and its running environment (hardware infrastructure, software stack, access and security aspects,...). Details about the underlying infrastructure are transparent to SaaS users, who have simple access to the application's functionalities without the ability to control or customize its features. The SaaS delivery model has notably been popularized with SalesForce [44] and its Customer Relationship Management (CRM) application. Today the SaaS is more widespread with many new offerings such as social media platforms, e-mails, business accounting, collaboration and management applications and online-gaming. Among the most popular SaaS applications, we cite the storage solutions Box Inc [45] and Dropbox

[46] and the collaboration software suite Google Apps [47] that includes Gmail, GoogleDocs and GoogleDrive. These SaaS solutions are either available for free use or charged on a subscription basis.

Platform as a Service (PaaS): This Cloud delivery model is typically designed for software developers and provides them with platforms to design, develop and deploy applications. PaaS platforms are high-level integrated environments (OS, programming languages, libraries, databases, web servers ...) supporting the full software life-cycle. PaaS users have full control of the applications and the environment configuration settings, but no control of the underlying infrastructure that is maintained by the cloud provider. This aims at simplifying the software development process and allowing developers to focus on their applications' core features without worrying about complex low-level management operations. Examples of popular PaaS platforms include Google App Engine [48], Microsoft Azure Cloud Services [49] and Pivotal Cloud Foundry [50].

Infrastructure as a Service (IaaS): Moving down the stack, we get to the fundamental model for delivering Cloud services namely the IaaS service [51]. It refers to the on-demand provisioning of basic IT infrastructure resources (processing power, memory, storage and network). IaaS users can request either virtualized resources delivered in the form of virtual machines or containers (see section 2.2.2), or rent physical servers for sake of better performance (known as bare metal service MaaS). IaaS customers have higher control over their resources compared to SaaS and PaaS models. They are responsible for managing the deployed OS, applications and data, while IaaS providers still manage the underlying hardware and virtualization layers. IaaS users are able to dynamically scale their rented resources according to their workloads which allows them to pay only for what they use. The most prominent actor in the IaaS market is Amazon Elastic Compute Cloud (EC2) [52] offering different VM instances with various computational configurations and OS kernels rented at fixed and dynamic prices. Other popular IaaS providers include Microsoft Azure [53], Google Compute Engine (GCE) [54], IBM SmartCloud Enterprise [55] among others.

To easily distinguish these services, it helps to remember that IaaS is about hosting, PaaS is about building and SaaS is about consuming. With the advances of Cloud services, the market is moving to the scenario where each IT system component can be provided as a service over Internet (Network as a service or Naas, Monitoring as a service or MaaS,...). This increasing selection of services is often referred to as "XaaS" (Everything as a service).

This thesis work focuses on IaaS services. Nevertheless, the proposed allocation approaches can be easily extended to apply to the PaaS and SaaS models. Our graph-based requests modeling (discussed in section 4.2.2) is generic enough to address all services. Throughout the thesis manuscript, we refer to IaaS vendor by "the Cloud provider", "the Cloud Service Provider" or "the Cloud" unless otherwise specified.

IaaS services can be deployed in different scenarios, that can be classified into four main deployment models [20] depending on the Cloud infrastructure ownership and the access rights to deployed services, as illustrated in Figure 2.4:

Private Cloud: provides highly secure services used exclusively by the organization that owns the infrastructure and maintains full control over it.

Community Cloud: refers to an IT infrastructure owned and shared for collaboration between a group of organizations having common concerns.

Public Cloud: refers to a large and highly efficient IT infrastructure owned and managed by an external organization, that provides on-demand services to the general public. Services and data are hosted outside the users' premises.

Hybrid Cloud: refers to an infrastructure combining two or more Clouds (private, community, or public) that remain independent entities but are bonded together to enable in-house and external services deployment.

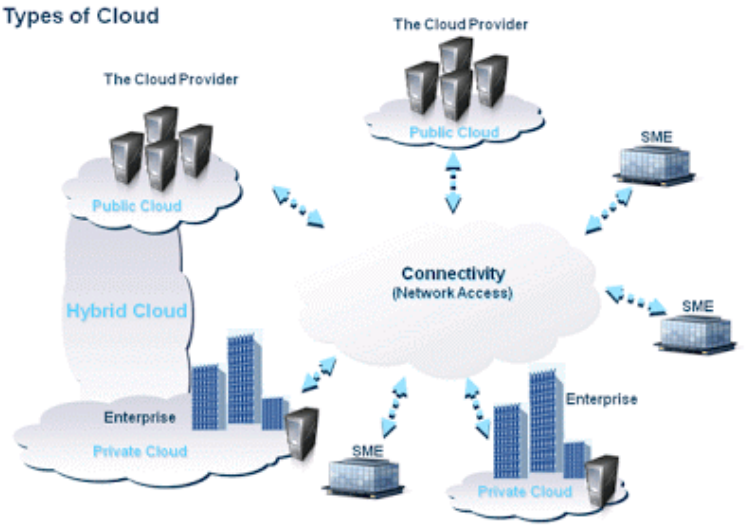


FIGURE 2.4: Cloud deployment Models.

Besides these traditional deployment models, new distributed and inter-Cloud approaches are recently emerging to satisfy customers and providers requirements and provide higher flexibility and scalability. The next section 2.3 is dedicated to an overview of these inter-Cloud models.

2.3 Federated Inter-Cloud Environments

Nowadays, the cloud adoption is increasing at a rapid pace and users become more demanding in terms of performance and QoS. To meet this evolution and higher demand, providers must be able to dynamically adjust their hosting capacities in response to workload variation and QoS requirements. Recent studies have introduced the concept of interconnecting and federating multiple Cloud platforms as an efficient solution to overcome resource limitation and satisfy users requirements. In this context, various recent works [56–60] have suggested different *Inter-Cloud* architectures for interoperability and cooperation between separate Clouds. In this section, we briefly discuss the limitations of current *"Monolithic"* Cloud architectures, before reviewing the main Inter-Cloud scenarios and major challenges in such distributed environments.

2.3.1 Limitations of Single-Cloud Deployment Model

Despite the advancements in cloud technology, traditional architectures are still having several challenges and limitations that hinder cloud adoption and performance. The major limitations of standard single-Cloud deployments are the following:

Limited scalability and availability: Although the Cloud gives the illusion of infinite resources, in practice there would always be a maximum bound on the provider's hosting capacity restricted by hardware and network capabilities. Resource shortage remains problematic for small and medium-sized providers due to increasing demands [8], and even for large providers during workload spikes and technical failures [61–64]. Service disruption and unavailability affect directly the Cloud providers profits, since it may result in losing reputation and customer initiated penalties for QoS violations.

Lack of interoperability and Vendor lock-in : Cloud systems were not designed to interoperate with each other and lack of standardization and compatibility between the underlying technologies. Cloud providers propose heterogenous proprietary solutions and access interfaces, which hinders business partnership achievement and profit improvement [65–67]. This lack of interoperability is also a crucial problem for Cloud customers since they become dependent on a particular vendor. This *vendor lock-in* may lead to economic and functional losses for users due to unfavorable deployment and pricing plans, and involve significant costs and technical efforts to migrate their workloads to other clouds.

Performances degradation: Single-cloud deployment may result in significant performance degradation due to the distant service location from the users distributed worldwide, which leads to increased latency and response times. This deployment model suffers also from a single point of stress and may result in total service interruption in case of failure. Furthermore, with a single Cloud it is hard to satisfy complex demands requiring advanced QoS requirements.

These issues accentuate the importance of interoperability and motivate Cloud actors to move into inter-Cloud architectures for better reliability and scalability.

2.3.2 Inter-Cloud: Definition, Benefits and Deployment Scenarios

IT experts predict that the cloud market will converge towards a federated interoperable environment, through a three-stage evolution [60, 68, 69]. The cloud market is currently moving from *"Monolithic"* providers delivering services based on their own infrastructures, to the *"Vertical supply chain stage"* where providers operating at different service levels can request resources from others to deploy their services [70]. The expected third stage *"Horizontal Federation"* consists in the cooperation and resource sharing between several providers to satisfy users' demands [71]. Achieving full transition to this federated model is far from trivial and requires overcoming many management and interoperability issues before it comes into wider usage. Our thesis is focusing on this federated Cloud model to contribute to the optimization of resource allocation and cooperation between providers. In the following sections, we briefly review main architectural aspects of *horizontal federations*, also called *inter-Cloud* [57] or *cross-Cloud* [58, 60].

2.3.2.1 Definition of the Inter-Cloud model

The term *"Inter-Cloud"* has been firstly introduced by *Cisco* [13] to define a novel vision of globally interconnected Clouds *"Cloud of Clouds"*, inspired by the *Internet* paradigm known as a *"network of networks"*. The Inter-Cloud concept focuses on the use of open standards and protocols to achieve interoperability across heterogeneous Clouds [65] and provide a unified mesh of shared resources between providers [13]. Hereby, providers can freely distribute their loads among distinct Clouds to meet requirements, while users can easily migrate their services to suitable providers whenever needed. The Inter-Cloud was formally defined by the Global Inter-Cloud Technology Forum (GICTF) as follows [14]:

"A cloud model that, for the purpose of guaranteeing service quality, such as the performance and availability of each service, allows on-demand reassignment of resources and transfer of workload through a interworking of cloud systems of different cloud providers

based on coordination of each consumer's requirements for service quality with each provider's SLA and use of standard interfaces".

The GICTF white paper [14] has also identified the main inter-Cloud use cases and the functional implementation requirements. Many other academic publications [56, 57, 59, 60, 72] adhered to the above definition and proposed various architectural initiatives for different inter-Cloud scenarios including Cloud brokering, bursting and aggregation. The next section describes the features of the main inter-cloud scenarios and identifies that addressed in this thesis.

2.3.2.2 Benefits of Inter-Cloud Deployment Models

Inter-Cloud models bring numerous advantages for both Cloud providers and customers. Among the key benefits of such interoperable environments we cite:

- **High scalability and flexibility:** Inter-Cloud models enable providers to meet workload fluctuations while saving costs. Instead of over-provisioning extra capacities for peak-load periods, federated Cloud enables providers to cost-efficiently adjust their hosting capacity through cooperation with others [56, 57].
- **Fault tolerance and high availability:** The distribution and replication of service components across multiple Clouds avoid the single point of failure and ensure better reliability and availability. During a site failure, the service downtime can be easily controlled through dynamic resource reallocation across other Clouds. Inter-Cloud model is identified as a substantial solution for fault tolerance and disaster recovery in case of failure [61, 73].
- **Cost and performance efficiency:** Since it is difficult for providers to own data-centers in each region, Inter-Cloud model is an efficient solution to expand their geographic footprints to satisfy users' location constraints and improve service performance and latency. The Inter-Cloud model allows also saving costs due to efficient resource aggregation. In [74], an evaluation based on a service brokering shows that multi-Cloud VMs deployment improves QoS and minimizes costs compared to the single-Cloud deployment case. Moreover, given the time-varying pricing among providers, dynamic resource reallocation can further reduce the overall deployment cost. Authors in [75] investigated the energy cost minimization problem through federating Clouds. They proposed dynamic allocation policies to place and migrate VMs based on time-varied electricity costs and cooling effects, and showed that dynamic strategies outperform static allocations.

2.3.2.3 Architectural Classification of Inter-Cloud Scenarios

Cloud actors can benefit from various Inter-Cloud models that differ in the initiator and degree of the collaboration, as depicted in Figure 2.5. Inter-Cloud scenarios can be classified ranging from *loosely coupled architectures* where cloud providers have limited control and basic operations over remote resources, to *tightly coupled architectures* enabling advanced control and cross-site networking and migration features [76–78].

On the other hand, we can distinguish according to the initiator of the inter-Cloud two usage scenarios: *Cloud Federation* and *Multi-Cloud* [79]. For *Cloud Federation*, designated as *provider-centric interoperability* [5], there is a volunteer cooperation between providers based on prior business agreements. The *Hybrid Cloud* combining private and public Cloud infrastructures [20] is also considered as a provider-centric federated approach [5, 76, 79]. In contrast for *Multi-Cloud* scenarios, known as *client-centric interoperability* [5], the resource aggregation is initiated and managed by end-users or more often by third-party *brokers* responsible for the full management cycle [59]. This section briefly describes the features of these inter-Cloud architectures.

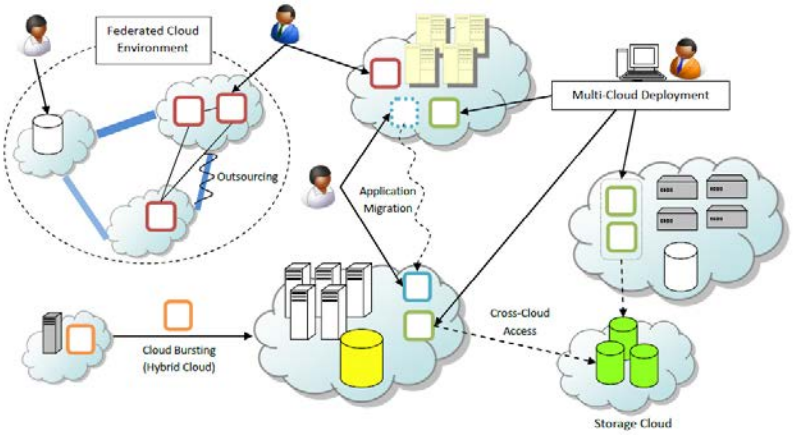


FIGURE 2.5: Interoperability and Inter-Cloud Scenarios.[5]

Hybrid Cloud: Is an infrastructure combining private resources owned by the user and restricted for its internal use, with remote resources provisioned dynamically from public Clouds, as seen in Figure 2.5. This loosely coupled architecture, usually referred to as *Cloud Bursting*, allows users to offload part of their workload to external Clouds when their data-centers are overloaded. Typically, non-critical tasks are outsourced to public Clouds while critical jobs are hosted in the private infrastructure. This allows taking advantage of both public (cost-efficiency, scalability,...) and private Clouds (privacy, control, security,...). For thus, most Cloud management platforms such as OpenNebula [80] or OpenStack [81], support hybrid deployment [82].

Multi-Cloud: In this scenario, the end-users are responsible for selecting the better aggregation of resources from multiple Cloud providers to meet their services requirements. Users are in charge of the full management cycle, including resource planning and deployment, SLA negotiations, performance monitoring and resource migration. To handle such task, users require different API adapters for the involved heterogeneous cloud providers.

Cloud Brokering Service: *Brokering Services* [74, 83] have emerged as fundamental mechanisms to facilitate the interoperability and reduce the management complexity in Multi-Cloud environments. A Cloud broker acts as a mediator that negotiates contracts between Cloud customers and providers and manages the service delivery and usage. The broker may have prior agreements with multiple public Clouds that update regularly the state of their service offerings (available resources, prices, QoS guarantees,...). Instead of directly soliciting providers, the customers submit their requests to the Cloud broker that selects from its repository the best providers and services matching users' requirements and criteria (cost, geographic location, performance,...).

According to the NIST [21] and Gartner Research Company [84], Cloud Brokers offer services, that can be categorized into three main roles:

- *Cloud Service Intermediation:* by enhancing the initial service's capabilities through value-added functionalities. This can include identity management, advanced billing services, performance monitoring and reporting, and failure recovery.
- *Cloud Service Aggregation:* through the integration of heterogeneous distributed services into a new cohesive one, accessed and managed through a unified interface.
- *Cloud Service Arbitrage:* allowing flexible and dynamic services deployment across suitable Cloud providers.

The inter-Cloud brokerage scenario has received considerable attention in scientific and industrial research. Several broker architectures, suggesting the system components and interactions between them, have been designed in the literature. Related work include the SLA-based broker [85], the STRATOS brokering service [86], the SLA-Based Cloud@Home broker [87], among many others. Other works have focused on particular broker component, notably the resource selection and scheduling module [74, 83, 88–91]. Relevant provisioning algorithms will be reviewed in next Chapter 3. To facilitate the adoption of brokering scenarios, several European and International research projects have designed and implemented various architectures dealing with different optimization

objectives. Among them we cite: the *mOSAIC Project* [92, 93] proposing Ontology-based brokering module for resource discovery and usage ; the *CompatibleOne* broker [94, 95] based on open standards (notably OCCI) to assist Cloud customers in their resource selection ; the *SLA@SOI* European project [96] designing a broker framework for automated SLA management in SaaS provisioning [97] ; the *OPTIMIS* project [98] implementing a Cloud brokerage module for SLA negotiation and management based on the WS-Agreement standard, with a special focus on identity and security issues [99]. A deep comparison of the most known Open-Source brokerage solutions can be found in [100]. In [5, 79, 101] authors have reviewed the brokering strategies in federated Clouds and highlighted the features, advantages and drawbacks of each solution.

In this context, we have contributed during the thesis work to the *CompatibleOne project* [94, 95], by proposing an OCCI-Compliant placement module called *COPS* (CompatibleOne Placement Service) to handle the resource provisioning across involved providers. The *COPS* component is invoked by the broker to select the optimal resources aggregation that best-match the users' requirements. The placement results are used by the broker to create the provisioning and SLA contracts with selected providers. To control users' constraints and providers' offerings heterogeneity (cost, performance, security, location,...), the placement module resorts to a multi-criteria algorithm to drive the placement decisions. The current *COPS* version uses two equal-weighted preference criteria to select the suitable clouds: the reputation of the providers and their pricing offers. The model is generic enough to easily integrate new criteria if needed. Each criterion is associated with a weight that reflects its importance compared with others. These weights can be specified by the Broker based on its financial and security objectives, or according to users' preferences. To communicate with the CompatibleOne ACCORDS platform [102], the *COPS* module is based on a Restful API implementing the OCCI HTTP rendering [103] and new-defined OCCI categories as specified in the CompatibleOne Resources Description System (CORDS) [104]. The *COPS* service can as well operate with the JSON rendering to address a broader audience. The *COPS* module was actually developed and designed in the context of this thesis work.

Volunteer Cloud Federation: Cloud federation consists in a trustful cooperation between two or more independent providers to share their respective resources. This coalition is governed through *"Federation Level Agreements (FLA)"* defining the rules and conditions that regulate the pooling and trading of resources [105]. This inter-Cloud model has been proposed as a new paradigm empowering IaaS providers to overcome resources limitation. Providers having complementary resources requirements over time can collaborate and dynamically adjust their hosting capacities to fulfill users' demands and gain additional revenues [105, 106]. During workload spikes, Cloud providers

can alleviate requests rejections and SLA violations through *"outsourcing"* part of the load by *"borrowing"* additional resources from federation members at negotiated prices. Providers can thus achieve higher resource availability and better reputation among customers. On the other hand, providers receiving low workload may avoid wasting resources via *"renting"* their idle capacities and *"insourcing"* partners' requests. Apart from this collaboration, participating providers remain independent and competitive in the Cloud market and may use different management and pricing strategies.

In this scenario, the cooperation and workload offloading between providers is totally transparent to the customer, who is not aware about the federation and the way its service is delivered (hosted locally or outsourced across the federation). Cloud federation architectures are in general partially coupled [76, 78] since providers should have some advanced control over the remote resources to seamlessly execute users' actions (migration, resizing,...).

Given its promising benefits, Cloud federation has received recently a growing interest to speed up its adoption among Cloud stockholders. The definition of architectural features and necessary standards enabling such collaboration has received the majority of attention from the scientific community [79, 101]. Relevant related works include:

- The *RESERVOIR* European project [56] that introduces a modular and extensible architecture for IaaS providers federation. Each Reservoir site contains three management layers: the *Service Manager* for high-level tasks including requests provisioning, SLA monitoring, accounting and billing; the *Virtual Execution Environment Manager (VEEM)* responsible for VEEs management and interaction with remote sites; and the *Virtual Execution Environment Host (VEEH)* handling virtualization features and VEEs migration among distributed platforms.

- In the *Contrail* European project [72], an SLA-centered federated approach is proposed to allow resources usage and deployment across different Clouds. The Contrail three-layered architecture provides a single unified access interface to the federation and supports advanced SLA management. The top *Interface* layer ensures interactions between users, providers and Contrail components through CLI and REST interfaces. The mid *Core* layer contains the necessary modules to support the federation features and requirements, including identity management, resources discovery and selection, applications life-cycle management, SLA negotiation and monitoring. The bottom *Adapters* layer contains internal and external adapters to enable the interaction with both Contrail and non-Contrail Clouds.

- Authors in [57] have proposed a market-oriented *InterCloud* architecture for flexible and scalable distributed resource provisioning. The *InterCloud* model is based

on three key elements: the *Cloud Exchange* that maintains the registry of available providers and their offerings, handles resource trading based on auctions, and enforces financial and payment transactions; the *Cloud Coordinators* managing the federation memberships by providing the basic features for the resource discovery, allocation and monitoring and the periodic updates of the registry; and finally the *Brokers* handling SLA and resources negotiation on behalf of users.

Other federation architecture proposals include the *OPTIMIS* toolkit [59], the *Open Cirrus* architecture [107], the *Cross-Cloud* approach [60], the *Dynamic Cloud Collaboration (DCC)* [108], among many others. Generally, Cloud federation architectures can be classified as *Centralized* and *Decentralized* (or *Peer-to-Peer*) [79] approaches. In centralized architectures, there is a central entity responsible for the resources trading and workload distribution among providers as proposed in [57, 72, 108]. On the contrary in peer-to-peer approach, the involved providers negotiate cooperation and resource sharing directly without any mediators as in [56, 60, 107].

Unfortunately, much less research works have been focused on the management aspects within a federation, notably the challenging problem of workload distribution and resources allocation. This thesis work assumes a decentralized volunteer Cloud federation, to propose a novel model assisting providers in their cooperation and placement decisions. Among the different incentives of federation, we focus on its economical benefits as solution for enhancing Cloud provides' profits through insourcing and outsourcing resources. A deep review of related literature works [105, 106, 109, 110] will be presented in next Chapter 3.

2.3.3 Drivers and Barriers for Cloud Federation

2.3.3.1 Drivers and Conditions for Federation Profitability

Ideally, a Cloud Federation should be profitable for all involved providers in terms of revenues and acceptance rates, but it is not always the case. A coalition can be less favorable for some providers owing to different factors that may impact the potential federation benefits, including the following:

- The sizes of involved Clouds, in terms of available and shared resources (balanced capacities or highly variable infrastructure sizes among providers).
- The federation size, meaning the number of providers joining the coalition.

- The instantaneous workload received by the federation. The achieved benefits can also depend on the providers' locations, since geographically and timely distributed federation may better manage load variations and peak hours.
- The types of offered resources within the federation. In fact, the existence of similar offerings among providers is fundamental since they are willing to serve each other's requests. The cooperation with providers offering different services may also be beneficial since it enables new business opportunities.
- The market rules and federation business agreements, in terms of pricing policies and resource sharing strategies.

Up to now, there is no study defining precise rules for building profitable federations according to these parameters. We discuss in chapter 4 some of these key drivers through the evaluation of our exact allocation algorithm and identified benefits, that have shed light on the favorable conditions leading to the best improvements.

2.3.3.2 Economic Challenges and Enabling Standards

The establishment of Cloud federation raises much more challenges than traditional Cloud models. These challenges cover broad requirements including services description and discovery, distributed resource provisioning, data portability and security, SLA negotiation and monitoring, inter-Cloud networking, accounting and billing, etc [5, 57, 111]. Substantial efforts are required to overcome these issues and develop necessary features enabling the wider adoption of Cloud federation. Reviewing all federation challenges is not the aim of this chapter that is limited to the description of its major economic issues relevant to our thesis work focusing on providers' profitability:

Interoperability and Portability between Clouds: To take full benefits from Cloud federation, providers should be able to seamlessly integrate and manage resources across various Clouds according to performance and business requirements. This requires the definition of standardized protocols and APIs for distributed resource management [13, 65, 82], which has been the focus of many industrial and research groups. Among the most adopted standards and APIs we cite: the *Open Cloud Computing Interface (OCCI)* providing a specification of a RESTful management API for provisioning and monitoring IaaS resources [112]; the *Open Virtualization Format (OVF)* for virtual appliances packaging and deployment across heterogeneous platforms [113]; the *Cloud Data Management Interface (CDMI)* providing a generic interface for common data storage operations [114]; the Libcloud API [115] that abstracts the heterogeneity between Clouds

and enables large-scale deployment. Despite these standardization efforts, many operational challenges remain and have to be addressed. It will take time for these standards to be commonly supported by public Cloud providers.

Resource Allocation and Management: The resource provisioning task is significantly challenging in Cloud federation. It consists in finding the optimal placement and mapping of requested services onto the distributed available physical resources. The decision making process is especially complex due to the increasing number of federation actors and parameters, and the highly heterogeneity and dynamicity in such environment. The optimization is dependent on a multitude of decision criteria including the providers' workloads and shared offerings, the applications' constraints, the outsourcing and local allocations costs, the potential insourcing revenues, etc [106, 109]. Efficient allocation strategies are needed to enable providers to automate the selection of the optimal resource aggregations that better fit their business goals and users' demands according to the current federation conditions.

Several research efforts have focused on this optimization problem and proposed policies for distributed resource placement in federated Cloud environments [105, 106, 109, 110]. A detailed overview of relevant works in this area will be provided in next Chapter. Similarly, this thesis work focuses on the design and development of optimal profit-driven resource allocation models. Our objective is to address complex services requiring the provisioning of distributed resources and their specific networking topologies. Both exact and heuristic algorithms for optimal requests partitioning and distribution across the federation are proposed and detailed respectively in Chapters 4 and 5.

Resource Pricing and Market Regulations: Defining adequate market agreements for cooperation and resource sharing is another crucial challenge for federated Cloud providers. The latter should have a clear understanding of the resource trading decisions to better improve their profits [105, 106]. In this regard, there are two key factors that strongly influence their revenues, namely the shared resource quotas and the insourcing prices proposed for other members. In fact, resources allocation and pricing strategies are correlated issues and should be considered jointly to achieve better performance. For that, there is a growing attention from the scientific community to the adoption of market-based approaches for federation resources management [116–118]. These methods have proven their efficiency for resources scheduling, and highlighted the relevance of using dynamic pricing schemes in improving providers' revenues compared to fixed pricing [7, 117]. A detailed description of these two pricing models is given in next section 2.4.

Ideally insourcing prices and shared quotas would be dynamically adjusted at each allocation cycle to better reflect the fluctuations in supply and demand. Elaborate pricing schemes, combining various market parameters (e.g. received workloads, resources utilization level, future demands,...), are needed to achieve better performance improvements. This complex case-study is out the scope of this thesis that primarily focuses on optimizing the allocation and partitioning of resource requests under cost and networking constraints. This later optimization task is hard enough in itself to merit separate treatment. Nevertheless, to capture the essence of this study, we integrate from the literature a realistic pricing mechanism [105] that dynamically updates the insourcing prices used by our profit optimization algorithms. A discussion about pricing models relevant to the research study, and our choice motivations are presented in next chapter.

2.4 Resource Pricing in Cloud Computing

Resource pricing is the process of determining the prices that providers receive in exchange of selling (renting) resources. Various pricing methods can be applied in response to the market criteria (peak or off-peak times, fixed or changing pricing rates, resources availability). Defining the appropriate pricing strategies is important for providers to achieve successful business, since it is a key factor in regulating supply and demand, controlling users' behaviors and improving resource utilization.

Therefore, cloud providers need to determine the right value of their services and capture it through pricing. Different factors should be considered when setting prices, including operating costs, targeted profits, market competition, consumers' satisfaction and the service's perceived value. The resource pricing problem has been extensively studied by both academia and IT industry. Various pricing strategies have been proposed in the literature [119], varying from complex to simple models. In next Chapter, we cover some of these studies through a detailed review of relevant related work. In practice, existing Cloud providers use their own confidential methods for service assessment and pricing, which leads to a myriad of pricing types and options among providers. This section presents an overview of the main pricing models used in the Cloud market today with a focus on IaaS services.

2.4.1 A General Taxonomy of IaaS Pricing Models

Despite the promises of simple usage-based Cloud services billing, the diversity of offerings and pricing among providers have led to a complex business market. A fundamental

COMMON CLOUD PRICING METHODS

451 Research

		PAYMENT TERMS	MEDIUM OF EXCHANGE	COMMITMENT	METERING	UNIT PRICE VARIABILITY
CASH PAY-AS-YOU-GO	On-Demand	In-Arrears	Cash	None	Metered	Fixed
	Spot Pricing	In-Arrears	Cash	None	Metered	Variable
	Reserved Instance	Hybrid	Cash	None	Metered	Fixed
COMMITTED VM	Prepaid VM	Up-front	Cash	None	Unmetered	Fixed
	Recurring VM	Up-front	Cash	Recurrent	Unmetered	Fixed
RESOURCE POOLING	Recurring Resource Pooling	Up-front	Resources	Recurrent	Metered	Fixed
CREDIT PAY-AS-YOU-GO	Subscription Credit	Up-front	Credits	Recurrent	Metered	Fixed
	Prepaid Credit	Up-front	Credits	None	Metered	Fixed

FIGURE 2.6: The Taxonomy of IaaS Pricing models.[6]

step for Cloud users is to understand these pricing options and their Pros and Cons, to select the best offer fitting their needs and budgets.

Generally speaking, resource pricing is usually based on some economic model such as commodity market, flat-rate or auctions. A detailed survey covering the pricing methods and metrics applied among 53 IaaS providers has been done by the 451 Research group [6]. The research points out the great diversity among studied models and defines a general "Taxonomy of IaaS pricing" including eight main pricing categories, as illustrated in Figure 2.6. The study also outlines the characteristics of each pricing method, its strengths and weakness and its best-practice usage scenarios. This provides a guideline for IaaS users to better understand the Cloud economic landscape and make efficient decisions. A comparative description of the most prevalent pricing models in IaaS market, namely the *"Cash Pay-as-you-go"* pricing (*On-demand*, *Reserved Instances*, and *Spot Pricing*) and the *"Committed VM"* pricing, is presented in next section.

2.4.2 Common Pricing Types and Models

As shown in Figure 2.6, there are two different pricing rates, namely the *fixed-rate basis* and *variable-rate basis* changing over time based on market parameters.

2.4.2.1 Fixed Pricing

With fixed pricing, cloud providers set to each service a predetermined selling price that will be maintained during an extended time period. Fixed pricing mechanisms are easy to implement (controllable using a simple cost-plus strategy) and are the most popular in cloud market (Figure 2.6). The most well-known fixed pricing implementations are the *On-demand usage-based pricing* and the *Subscription-based pricing* described below.

The on-demand usage-based pricing: known also as *pay-as-you-go*, is the most common pricing model offered by the majority of IaaS providers (more than 90% according to the 451 Research [6]), including Amazon [52], Google [54], Microsoft Azure [53], and many others. This model is based on metering the customers' resource usage to bill them accordingly. The resources are quantified as usage units charged at time-based fixed prices (VM instance per hour, gigabytes of storage per month,...). Customers acquire resources on the fly and pay only for their consumption independently of their request time. From users' perspective, the on-demand pricing model may be an attractive solution to rapidly scale up/down resources, to enable riskless service experimentation without long-term commitments and to ensure a guaranteed service during the whole time horizon at a known price. However, for long-term utilization, this pricing model may not be suitable for users to minimize their provisioning costs. To satisfy customers, many providers (over 50% according to [6]) offer new pricing alternatives enabling more cost-effective resource usage, as detailed in next sections.

Subscription-based pricing: is a fixed pricing based on the payment of some upfront fee to subscribe to a service during a predefined commitment period. This pricing model is implemented by many IaaS providers (Amazon [52], Google [54], Microsoft Azure [53], etc.) with different specificities. For example, Amazon provides the *"Reserved Instances"* pricing scheme [120] that allows users to reserve a VM instance for one or three years by paying an upfront fee and receive in turn significant discount on the hourly usage price. GoGrid [121] in contrast offers the *"Prepaid VM"* model enabling users to pre-pay only a subscription fee to have unlimited free usage during the contract term. Using subscriptions helps users get lower prices with guaranteed service availability. This pricing model is especially profitable if resource utilization can be planned in advance to extensively use reserved resources during the contract term. This pricing scheme is also beneficial for providers since it helps them optimize the utilization of their data-centers and gain an assured revenue through subscription fees. However, they must ensure the availability of reserved resources whenever requested to respect the SLA contracts.

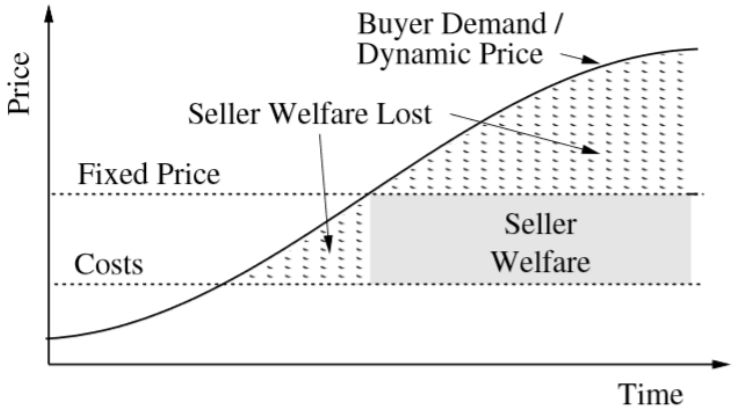


FIGURE 2.7: Fixed Pricing limits providers' profits. [7]

Fixed pricing remains the predominant strategy today. However, recent studies [7] have shown that this model can lead to financial loss for both providers and consumers, since it is not sensitive to supply and demand fluctuations. In case of under-demand, customers may pay a higher fee than market price or look for other providers and service offerings. Whereas in case of over-demand, fixed price may be lower than the market price which limits the provider's revenue, as shown in Figure 2.7. *Dynamic Pricing* has emerged as an efficient policy to cope with this issue and achieve better performance [7, 117].

2.4.2.2 Dynamic Pricing

Dynamic pricing is the practice of setting variable prices for the same service according to real-time market conditions, such as available resources or customers' expected QoS. The dynamic pricing strategy is the least common model in Cloud market. Amazon's *Spot Pricing* [120, 122] is the only implemented dynamic policy for selling IaaS services. However, this pricing strategy has received the highest attention in the literature [7, 116–118] due to its complex implementation and promising benefits.

Spot instance pricing: Is an auction-based scheme offering variably-priced resources via bid auctions. According to the Amazon price history, users can acquire spot instances at a reduced prices of 50% to 93% compared to on-demand instances. A spot request specifies the needed instance type, the availability zone, the reservation duration and especially the user's bid stating the maximum hourly price that he is willing to pay for using resources. Once sent, the request remains waiting until its bid meets the current spot price to be satisfied. The spot price is set by the provider and is supposed to be updated based on supply and demand. Once satisfied, the access to the VM instances remains active as long as the market price is fulfilled, otherwise these instances

terminate instantly. Although spot services are not guaranteed, this pricing remains an attractive cost-effective alternative for many interruption-tolerant applications such as web crawling and Map-Reduce tasks. Spot pricing is also beneficial for providers to sell unused resources and strategically manage customers' demands by adjusting prices.

2.4.2.3 Pricing Attributes and Resources Bundling

Current IaaS providers use different formats to provide their services to customers, including customizable computing resources, predefined bundles of packaged resources or in-between service offerings. The 451 Research survey [6] has identified four main levels of IaaS resources bundling, as detailed in the following:

Bundling Pricing Strategy: refers to the practice of combining several computing resources such as CPU and memory, into a single package to be sold as a unique resource for a single flat rate. This strategy is practiced by the majority of IaaS providers that offer a set of pre-configured bundles with varied resource capacities, known usually as *VM instance, VM class, or VM size*. Different bundling levels can be used, namely:

- *VM Bundled :* offering pre-configured VMs with specific CPU, Memory and disk capacities, but the bandwidth is charged separately. Many providers such as *Amazon*[52], *Microsoft Azure*[53], *Google*[54], and *IBM*[55] provide this bundling type.
- *Fully Bundled :* offering VM instances with predefined CPU, memory, disk capacities and unlimited data transfer bandwidth. *Dedicated Server-Arsys Cloud*[123] provides this pricing type.
- *Processor Bundled :* defines only the CPU capacity of VM instances while the rest of resources are charged separately. *VMWare vCloud*[124] uses this pricing type.

Unbundling Pricing Strategy: allows customers to purchase individual computing resources at a fine-grained level to configure their VMs by themselves. The requested resources are charged separately per unit-usage (e.g. 0.01875$ per CPU/hour, 0.04$ per GB of bandwidth, 6.48$ for 500 Mhz of CPU per month, etc...). CloudSigma [125] and ElasticHosts [126] are two well-known providers offering this pricing type.

Choosing between bundled or unbundled service offerings is not a trivial decision, and is mainly depending on the user's workload characteristics and requirements. Bundled services are favorable when most of the packaged resources are needed, otherwise it is more advantageous to purchase unbundled resources to avoid paying unused capacities

and enable fine-granular elasticity. Without loss of generality, our optimization study assumes a bundling pricing strategy offering VM instances with a preset amount of CPU and memory. For the pricing rates, both fixed and dynamic policies are used to feed the allocation algorithms. While fixed pricing are used to charge end-users requests, demand-oriented dynamic pricing are applied for resource trading within the federation.

2.5 Thesis Scope and Focus

This thesis addresses profit optimization for IaaS providers involved in a cloud federation. If this concept alleviates the problems of resource limitation and workload fluctuations, it introduces new management challenges and tradeoffs between users' satisfaction, revenue maximization, and federation agreements fulfillment. Therefore, we aim to define efficient resource allocation strategies, to assist federated IaaS providers in selecting the profitable cooperation decisions in response to their workloads, in-house available capacities and federation offerings. We focus exclusively on the federation management level in terms of outsourcing, insourcing, local hosting or request rejection decisions, but not on the VM placement and consolidation inside each data-center.

Both exact and heuristic solutions for the resource federation problem are proposed and compared in terms of complexity, performance and scalability. Our approaches are VM based and provide on-demand allocation strategies for complex and composite service requests. Resource requests are charged according to a bundle pricing strategy. We consider both fixed and dynamic prices for serving end-users and federation members respectively. The End-users prices are fixed according to standard on-demand schemes used in IaaS market, while insourcing prices are set using a pricing estimation model from the literature.

2.6 Conclusions

This chapter provided an overview of necessary background and foundations for the thesis work. We introduced the key concepts and enabling technologies of Cloud Computing paradigm and the commonly used pricing models for billing Cloud resources. We investigated as well the features and benefits of inter-Cloud scenarios, with a special focus on Cloud federation emerging as a potential solution for providers' profitability. We reviewed the major drivers and economic challenges for profitable federations including the optimization of resource allocation decisions, the focus of this thesis.

The next chapter investigates in more detail the problem of Cloud resource allocation, with a special focus on federated cloud environments. We review state-of-the-art solutions for profit-driven resource allocation, as well as some relevant market-based allocation studies and dynamic pricing policies, suitable for our optimization scenario. Then, we discuss the related gaps and issues and introduce our contributions in this field.

Chapter 3

Cloud Resource Allocation: State of the Art

Contents

3.1 Introduction

Cloud federation is a new concept enabling providers to cooperate and share their infrastructures to meet users' demands. A key challenge for federated providers is to define effective resource allocation strategies to take full advantages of this cooperation. The mapping of user requirements and provider goals to resource provisioning in the Cloud infrastructures raises several challenges due to the scale of modern data-centers, the heterogeneity of resource types and the variability of received loads. The problem

becomes more challenging in cloud federations involving multiple providers and various distributed resources.

To get a broader view of the Cloud resource allocation problem, a deep review of related studies is needed to identify the main issues and gaps. This chapter provides a detailed description of the problem and its associated challenges. Then it presents an overview of state-of-the-art solutions, with a special focus on profit-driven allocation models in federated Clouds. The literature analysis is based on several aspects such as the service request type, the optimization goal and the underlying Cloud architecture. The chapter presents also some relevant work on market-based resource allocation and dynamic pricing models suitable for our study.

3.2 Resource Provisioning and Allocation in the Cloud

Efficient resource management is one of the key issues in Cloud environments and is of prime interest to both Cloud providers and users. This challenging task has become an active area of research in recent years. In comparison to the studies devoted to user cost minimization, relatively much less attention has been paid to provider-centric allocation solutions to help them build profitable business. As the owner of the physical infrastructures, cloud providers are responsible for hosting, maintaining and allocating resources to the customers for their computational needs. Reducing the operational costs while maintaining high levels of user satisfaction are important factors for providers to increase their revenues and remain in business. Achieving this goal requires efficient allocation strategies to schedule user requests on the provider infrastructure. The resource allocation task consists in finding the best mapping or assignment of the received requests having different resources requirements and performance objectives, onto the available local and possibly remote physical resources having heterogeneous capacities and different performance characteristics and pricing models. This mapping has several challenges and is driven by both user requirements (e.g., SLA, localization, latency) and provider's business goals (e.g., cost optimization or energy consumption optimization).

The problem of resource allocation in large-scale shared cloud infrastructures is known to be NP-hard and has been studied in many contexts in the past. Related work has addressed different optimization problems that involve separate considerations and objectives. Studied topics covered different kinds of resource provisioning plans (e.g., on-demand, advanced reservation and Best-effort requests), various types of service requests (simple VMs, composite services, VDC, etc.) and different allocation policies. The proposed policies targeted initial VM placement, dynamic resource reallocation, energy-efficiency, load-balancing, cost-efficiency, reliability and fault-tolerance, etc. Since it

is a non-deterministic problem, several algorithms have been used to solve these allocation scenarios, including Mathematical Programming (Combinatorial optimization, Stochastic Programming, Constraint Programming, ...), Multi-criteria decision making, Genetic algorithms, Bin-Packing approximation heuristics.

This thesis addresses cost-effective allocation policy for on-demand resource requests to help providers select profitable distributed allocation plan for both simple and composite services. This chapter reviews the related literature and discusses the associated challenges. The prior art can be classified into two main scenarios, namely single-cloud environments and multi-cloud environments, as discussed in the following.

3.3 Resource Allocation in Single-Cloud Environments

In single-cloud environments, the resource allocation process consists in selecting an optimal set of physical machines to host the received services (VMs), while respecting resource and QoS constraints. The service and infrastructure characteristics (e.g real-time monitoring information, pricing policies,...) are usually exposed to the optimization algorithm and can be used as parameters for the placement decisions. Different approaches were used in the literature to solve this NP-Hard problem [127] with the aim to achieve good tradeoffs between solution quality and computation time. Related works address different allocation policies focusing on various criteria as described in the following:

- **Load balancing :** Authors in [127] design an end-to-end management layer for non-disruptive load balancing across the different resource layers. They propose the VectorDot algorithm, inspired from multidimensional knapsacks methods, to address the hierarchical and multi-dimensional resource constraints in datacenters. Using the dot products of capacity usage and resource requirement vectors, the algorithm identifies the best destination to migrate VMs from overloaded server, switches or storage nodes.
- **Service-Level Agreement (SLA) compliance :** Bobroff et al. [128] propose a dynamic server migration and consolidation algorithm to reduce resource consumption and SLA violations. The algorithm is based on measuring historical data, forecasting future demand, and then re-mapping VMs to the physical servers according to a Bin-Packing First-Fit heuristic.
- **Energy Efficiency :** Borgetto et al. [129] present an integrated management framework based on VM migration and server power management, to reduce energy consumption while keeping predefined SLA. The framework incorporates an

autonomic management loop that uses a variety of heuristics ranging from rules to random optimization methods, while taking into account the costs of VM migrations and server powering on-off.

- **Cost-based Consolidation :** Authors in [130] address the autonomic resource management problem through a two-level architecture, that decouples the application functions from the generic decision-making layer. A local decision module measures the applications satisfaction with regard to its performance goals using utility functions. Based on both these SLA fulfillment degrees and operating costs, the global decision layer optimizes the VM provisioning and placement into a minimum number of active PMs, using a Constraint Programming approach.

3.4 Resource Allocation in Multi-Cloud Environments

In multi-cloud scenarios, the resource allocation decision is usually focused on the selection of the best cloud infrastructures to distribute and deploy user workload. This is consistent with the cases of cloud bursting, cloud brokering and cloud federation described in Chapter 2. In such scenarios, the information about resource usage and load distribution inside each provider are commonly hidden from others. Only business-related information (e.g VM instance types and prices, datacenter locality, ...) are exposed to the placement optimization process. Therefore, most related work is centered on cost and profit aspects. In this section, we review relevant works on profit-driven allocation models, with a special focus on those using resource outsourcing as a technique for placement optimization.

3.4.1 Resource Allocation in Cloud Brokering Scenario

Cloud brokers [79] have recently emerged as mediators between users and providers to facilitate the selection and integration of services from different Cloud providers, which is too complex for users to manage by themselves. Users can solicit a Cloud broker that selects for them the best service offerings that match their demand requirements and achieve the cost-effective resource deployment plan, while hiding the complexity of contracts negotiation and services integration. Resource allocation in multi-cloud brokering environments has received the most attention from the research community. In this section we present some of the relevant related work in this area.

Tordsson et al. [74] propose a cloud brokering mechanism that optimizes the placement of VMs across multiple Cloud providers according to the demand constraints, resource

prices and user-selected optimization criteria (performance, cost and load-balance). The authors consider a static approach and propose a $0-1$ integer programming formulation to achieve optimal cost-performance tradeoffs.

Chaisiri et al. [89] address the optimization of resource provisioning costs in multi-cloud computing environment under demand and price uncertainty, by proposing a stochastic integer programming (SIP) formulation that minimizes both on-demand and oversubscribed costs. In [90], the authors extend the work and provide new methods for fast decision making, including deterministic equivalent formulation, Sample-Average Approximation and Benders decomposition approaches.

Li et al. [91] propose a linear integer program for dynamic VM placement across multiple Clouds. The model handles changes in both infrastructure conditions and services requirements through VM migration and uses different levels of migration overheads when restructuring the existing virtual infrastructures to fit optimization criteria.

Similarly, Lucas-Simarro et al. [83] [88] investigate the problem of dynamic VM placement and migration across available Cloud offers. In [83], authors propose a brokering scheduler module for the optimization of VM deployment costs in dynamic pricing multi-cloud environments. The scheduler uses a prediction model to estimate the next hour prices based on historical prices, their averages and their trends of variability, and accordingly suggests the best cost-effective deployment plan. In [88], a modular Cloud broker architecture offering different scheduling strategies is presented. The scheduler component, based on binary integer programming, supports both static and dynamic scheduling scenarios and can handle cost and performance optimization under different deployment restrictions (budget, performance, VM instance types, reallocation or load balancing constraints).

All the previously described approaches focus on the allocation of separate VMs across providers without considering the links and network topology between virtual resources. These models are not suitable for complex services where the satisfaction of networking requirements and relations between elementary components is essential to achieve optimum service performance. Moreover, the discussed works propose user-centric solutions that are not appropriate for the optimization of providers' performance objectives.

3.4.2 Resource Allocation in Hybrid Cloud

The outsourcing of cloud resources is not only considered in the context of federated Clouds. The outsourcing technique has also been investigated in hybrid clouds as a way to increase applications' scalability and improve performance and QoS. Managing the

outsourcing decisions, involving hybrid private and public resources, is a complex issue that has been addressed by several studies.

Van den Bossche et al. [131] propose a cost-optimal scheduling method for the problem of workload outsourcing in hybrid cloud environments, with a focus on preemptive deadline-constrained and non-migratable workloads. Their optimization objective is to maximize the utilization of the internal data center and to minimize the cost of external provisioning from public clouds, while respecting the applications' QoS constraints. They formulate the problem as a binary integer program considering both compute and data transmission requirements, and provide experimental insight into the scalability and performance of their formulation.

In [132], Fito et al. use the outsourcing technique to meet service level agreements (SLA) in hybrid clouds. The authors avoid SLA violations through an SLA-aware elastic model that outsources requests to a third-party cloud, whenever their SLA would be violated. Their work focuses on a reactive scheduler that scales up and down resources based on immediate state of local servers and outsources workload to a public Cloud provider whenever local resources are not sufficient to guarantee SLAs and/or exceeds an acceptable local hosting cost. A heuristic is used to make the outsourcing decisions.

Javadi et al. [133, 134] propose a flexible hybrid architecture with several failure-aware provisioning policies to address the issue of node failure in private Clouds. The proposed architecture is based on inter-Grid concepts and includes a gateway (IGG) to interconnect involved Cloud providers. The IGG's scheduler policies are responsible for sharing the loads between the private and public Clouds, and aim to improve the users' QoS by renting additional public resources during failures. The provisioning policies proposed in [133] are Knowledge-Free and consider the workload model and failure correlations to redirect users' requests to the appropriate providers. The objective being to reduce the dependency to public Clouds and the induced outsourcing costs, while satisfying users' requirements regarding request deadlines. In [134], the authors present a generic three-step provisioning model including resource brokering, dispatching and scheduling. The proposed brokering strategy is based on the stochastic analysis of routing in distributed parallel queues and is adaptive to the cost and response time of resource providers. For request dispatching, both probabilistic and deterministic sequences are investigated, while the resource scheduling is handled through well-known scheduling algorithms.

Similarly, Moreno-Vozmediano et al. [135, 136] investigate the cloud bursting scenario for deploying large clusters of loosely coupled applications on top of multi-cloud infrastructures. The authors analyze the viability, performance and scalability of hybrid infrastructures for different distributed web server architectures. Both web server clusters are deployed using real testbeds comprising computational resources from the in-house

infrastructure, and external resources rented on-demand from public Clouds to handle peak demand periods. The solution provides an elastic provisioning model that allows to dynamically adjust the cluster size in response to users demands and improve the service availability and cost-effectiveness.

Lee et al. [137] have addressed the problem of profit-driven service request scheduling in Cloud systems, while taking into account user satisfaction. They propose a client satisfaction-oriented scheduling heuristic (CSoS) that maximizes the providers' profit by accommodating as many requests as possible while maintaining the QoS at an acceptable level. The proposed algorithm exploits the outsourcing of services to third-party providers as a solution to handle overloading situations and avoid user request rejection or deadline violation.

In [138], Zuo et al. propose an integer-program based allocation model to solve the deadline constrained task scheduling (DCTS) problem in hybrid Clouds. The objective is to find the optimal allocation scheme of internal and external outsourced resources to schedule users' tasks, while maximizing provider's profit and guaranteeing promised QoS. A self-adaptive learning particle swarm optimization (SLPSO) based scheduling approach is proposed to overcome the tendency of standard PSO to trap into local optima, and speed up convergence times with large size problems compared to CPLEX. In SLPSO, four velocity updating strategies are used to adaptively update the particles properties to improve the quality and robustness of the scheduling solution.

Most of the works discussed above are dealing with the allocation of simple requests involving separate VMs and independent tasks. The proposed models do not incorporate inter-VM communication requirements and costs in the outsourcing decision making. This leads to performance degradation with complex services involving distributed and interconnected resources. Another difference between these works and our study is that we focus on the business opportunities of cloud federation, not only as a technique to avoid service request rejection, but also as a way to improve profits by sharing the otherwise-wasted resources at competitive prices between involved providers. Our study addresses the broader question of tradeoffs between SLA violations (rejection rate) and providers' profit.

3.4.3 Resource Allocation in Cloud Federation

Cloud federation has been recently proposed as a key solution to help providers handle workload fluctuations through cooperation and mutual resource sharing. Related work is mainly focused on the architectural aspects of Cloud federation [56–58, 72, 107, 108], but unfortunately much less on its functional and economic issues, including the profit

optimization problem. The provider's profitability depends on several parameters such as the incoming workload, the shared quotas and the costs of resources and their networking. Efficient management strategies are needed to help providers make strategic decisions including optimal resource placement and sharing, which is the focus of this thesis. This section presents an overview of the state-of-the-art solutions on profit-driven allocation models in federated clouds and the related pricing and networking issues.

3.4.3.1 Cooperation and Profit-driven Resource Sharing

Previous work closer to our study is found in [105], [106] and [109]. Toosi et al. [105] consider two types of VM requests: on-demand and spot VM requests. On-demand VM requests correspond to the type handled in our work. These VMs are provisioned immediately to users when requested with a fixed price per hour for each accepted VM [120]. In spot VMs [122], end-users make bids for specific VM instance types to the infrastructure provider. A spot request is accepted only if the value of the bid is greater or equal to the spot price that changes on an hourly basis depending on the provider's load. The profit optimization and allocation policies in [105] rely on a simple comparison of the profit of outsourcing resources to the federation with serving requests locally by terminating Spot VMs (making higher the spot price). This leads to suboptimal solution, since each action is evaluated separately without considering the potential profit improvement if splitting requests across multiple providers and combining allocation actions. Moreover, authors consider only simple requests with unconnected VMs and only one type of VM instance, when current cloud services actually involve more complex requests with distributed and connected heterogeneous VMs.

Goiri et al. [106, 109] present a profit-driven economic model that characterizes providers' decisions when operating in a cloud federation. Authors propose a series of decision equations to serve user requests locally, outsource them to the federation, insource other requests from other providers and include the possibility of shutting down unused nodes to optimize overall cost and revenues. Their approach consists of trying each action independently and compare them to select the best outcome, which may result in suboptimal request partitioning and allocation decisions. The authors limit the study to simple resource requests with unconnected VMs and one type of VM instance, which is far from the expectations of Cloud users who require more complex virtual infrastructure topologies with multiple VM instances. In addition, the insourcing prices are set using a simple discounting method that does not provide enough incentives for providers to cooperate and share their resources within the federation to regulate supply and demand.

In [139], Breitgand et al. address the problem of elastic service provisioning based on a federated cloud approach. They present a general framework for policy-driven VM placement optimization using both local capacity and remote (federated) resources, with profit maximization and SLA adherence as main objectives. The problem is formulated as an integer linear program applied with different placement policies, including the optimization of power saving and load balancing within a cloud, as well as the minimization of outsourcing costs to external partners. For scalability goals, the authors provide a 2-approximation greedy LP rounding heuristic and describe the integration of the proposed algorithms into the RESERVOIR federation architecture [56].

Casalicchio et al. [140] present an inter-cloud outsourcing model to scale the performance, availability and security guarantees offered to cloud customers, while maximizing the provider's revenue. The proposed model assumes a cloud federation involving several service providers located in different zones and characterized by different resource capacities, costs and QoS guarantees. Each provider receives several requests with different QoS levels requirements from customers dispersed in various zones and experiencing different latencies and network delays to access remote resources. A mathematical optimization formulation is proposed to determine the optimal distribution and allocation of the incoming load across providers, while satisfying SLA constraints and minimizing allocation and outsourcing costs.

Other works in [110, 141, 142] have addressed the problems of efficient resource allocation and revenue maximization in cloud federations using game theoretic approaches. The proposed game models are used to study the behavior and decision-making process of self-interested providers when engaging in a federation. In [110], Niyato et al. study the cooperative behavior in a federation to share revenues, ensure fairness and mutual benefit for providers using coalition game theory. The considered scenario corresponds to multiple providers cooperating to establish a logical resource pool (a coalition) to accommodate their internal users and serve public cloud users. A hierarchical cooperative game model, composed of two interrelated games, is proposed to analyze when providers' cooperation can lead to a higher profit. They develop a stochastic linear programming game to study the resource and revenue sharing for a coalition of providers while taking into account the uncertainty in user demand. Because this coalition may not necessarily result in higher individual profit for all members, authors applied a Markov-chain based coalitional game to study the rational formation of coalitions allowing to obtain stable cooperative group and help providers join or leave the federation based on the coalition's payoff.

Samaan [141] presents an economic model to regulate the resource sharing within a Cloud federation in presence of demand-oriented spot market, based on a repeated game

theory approach. The author introduces a set of self-enforceable allocation strategies that aim to maximize the provider's long-term profit, by sharing part of its unused capacity to the federation and selling the rest in the spot market. The uncertainty of workload fluctuations and expected revenues has been considered as an incentive for rational providers to insource federation requests for free, so as to build an informal insurance against the future workload peaks and the risk of grim punishment strategy for non-cooperative providers. An efficient update rule, depending only on the current workload and the history of previous interactions among providers, is derived to find the subgame perfect Nash equilibrium values for the spot market allocations and make the sharing decisions.

Another related work is done by Xu et al. [142] who propose a cooperative game resource allocation algorithm for profit and customer satisfaction maximization in a dynamic cloud federation, where the providers can join or leave the coalition at any time and the total amount of shared resources is adjusted according to customer's QoS requirements. The proposed model provides two different approaches for cost-sensitive and time-sensitive consumers. The allocation decision concerns the resource amount each provider need to supply to the federation and the optimal assignment of customer tasks to these resources, so that the global utility function is optimized. The main objective and optimization strategies of the above discussed investigations differ considerably from our work; even if they bear some similarities a direct comparison is not feasible with the works in [110, 141, 142].

3.4.3.2 Networking Requirements and Issues in Cloud Federation

Although resource allocation in Cloud federation has attracted significant attention recently, most research focuses primarily on the placement of individual VMs, and ignores the communication requirements between resources. This may degrade the service performance and lead to higher deployment costs, especially with network-sensitive applications. While loosely coupled architectures have minimal constraints to be distributed across multiple clouds, tightly coupled applications involve more stringent requirements on traffic flows and coordination between components. The request topology and communication requirements have to be considered in the request partitioning and deployment decisions across the federation. This issue is at the center of this thesis research, which aims to provide topology-aware allocation policies supporting both simple and composite Cloud services.

The Virtual Data Center embedding problem is a related research area that is receiving an increasing attention. A VDC is a resources request consisting of several virtual

machines connected through switches, routers and virtual links with different bandwidth requirements. The VDC allocation in Cloud environments is known to be NP-hard and is more complicated than scheduling independent VMs. Different techniques and algorithms are used in the literature to solve this mapping problem [143–148].

Most work has focused on the embedding of VDCs in a single data center. Rabbani et al. [147] present a three-step minimum-cost-flow-based heuristic algorithm that maps VMs, switches and links separately. The algorithm first tries to assign the VDC request to a single physical server. If any of the three phases fails, the heuristic adds a new adjacent server and iterates the mapping process, while considering server defragmentation, residual bandwidth, communication costs and load balancing. Authors in [145] extend the study to dynamic VDCs embedding, where VM migration can be used to dynamically adjust the resource allocation plan in response to demand fluctuations and system conditions. They propose a migration-aware dynamic VDC embedding framework that aims to achieve high revenue while minimizing energy and migration costs. A general mathematical formulation dealing with initial VDC embedding, scaling requests and dynamic VDC consolidation is presented and solved using greedy algorithms. Xu et al. [146] consider the problem of embedding Survivable Virtual Infrastructure (graph of correlated VMs and their backups) at minimum operational costs. The optimization problem is handled in two stages (VM placement and virtual link mapping) subject to resource and bandwidth demand constraints. Authors use a heuristic to solve the VM placement sub-problem, and propose a polynomial-time linear program for mapping virtual links to the data-center network, while guaranteeing sufficient bandwidth for regular and failover communications between primary VMs and their backups in case of failure.

Few works have studied the VDCs embedding problem across distributed cloud infrastructures. Amokrane et al. [143] propose a VDC management framework that aims to maximize the provider's revenue and reduce the energy costs and carbon footprint of selected infrastructures. The proposed solution uses a location-aware Louvain heuristic to split the VDC request into different partitions with highly-communicating VMs, while minimizing the inter-partition bandwidth. A greedy algorithm is then used to place each partition into a single datacenter to reduce the backbone network load. The work presented in [144] is based on a similar approach that integrates a minimum k-cut algorithm to partition the virtual topology into smaller subsets according to a weighted load-balancing cost function, which are then mapped to different cloud sites. Alicherry et al. [148] propose a network-aware resource allocation model minimizing the communication latencies between allocated VMs, since this can affect the application performance and delay the overall completion time. The resource allocation process is performed in four steps: datacenter selection, request partitioning, rack selection and VM placement. The datacenter selection, reduced from the Max-Clique problem, aims to minimize the

distance between selected clouds and is solved using a 2-approximation algorithm. The same algorithm is used to select hosting servers while reducing the inter-rack traffic inside datacenters. Finally, a greedy heuristic algorithm is proposed for request partitioning and VM assignment to the identified resources. However, none of the presented models takes into account the opportunities of resource sharing between clouds when joining a federation.

The end-to-end mapping of request links onto the federation network is out of the scope of this thesis, that mainly focuses on the optimization of workload distribution and cooperation decisions within a federation. We nevertheless aim to provide network-aware allocation policies to improve the performance and cost-efficiency of deployed services. The communication requirements among VMs and the costs of networking distributed resources are both included in our formulation. Making the system aware of the application structure may significantly improve the solution quality, and allow easy extension of the proposed model to support end-to-end service mapping in future work. The goal is to minimize the load on the links and the induced cost when distributing, across the federation, VMs that require connectivity for their interactions.

3.4.3.3 Resource Pricing Issues in Cloud Federation

Resource pricing is another challenging issue for Cloud providers, since it directly impacts their resource utilization and the efficiency of their allocation strategies. Currently, the usage-based pricing remains the predominant model for offering cloud services. However, recent studies have suggested that this fixed pricing can lead to inefficient outcomes, due to the mismatch between resource availability and user demand fluctuation. A common solution is to move towards dynamic pricing schemes that adjust prices in response to market and service conditions. This real-time dynamic pricing is more suitable for resource sharing in federated Clouds, since it helps regulate the supply and demand and gives incentives for providers to join the federation. The success of such cooperation cannot be achieved without the resolution of the federation economic aspects and the definition of efficient resource pricing strategies. In fact, the resource allocation, pricing and trading mechanisms are correlated issues that should be addressed jointly to achieve maximum benefits. This explains the increasing interest in the use of economic-based approaches to address the resource allocation challenge in federated Clouds.

The proposal of new models for dynamic resource pricing lies outside the scope of this thesis. However, to broadly address the problem of profit optimization in cloud federation, we use a pricing model from the literature [105] to dynamically adjust the insourcing prices. This section presents an overview of related literature on dynamic

pricing and market-based resource allocation models relevant to our work. Studies in this area can be classified into two main groups that use either economic-based models or computer science techniques to address the pricing and profit maximization issues.

Most related work is based on theoretical game approaches that determine the dynamic prices through a social welfare maximization problem in a competitive market of multiple providers. Mihailescu and Teo [7] present a strategy-proof dynamic pricing model for resource sharing and allocation in a cloud federation, where users are rational and the resource demand and supply fluctuate as consumers join and leave the federation. Using simulations, authors show that their dynamic pricing is more suitable for federated clouds and it achieves better economic efficiency than fixed pricing in terms of higher user welfare and successful requests. Similarly, Hassan et al. [149] address the problem of optimal distributed resource allocation in cloud federations based on a game-theoretic model ensuring mutual benefits among providers. Authors use a price-based resource allocation strategy and develop both cooperative and non-cooperative allocation games to examine the interaction and social welfare maximization among providers under each game. The cooperative game is shown to be more efficient, cost-effective and scalable.

Other works consider the application of market-oriented pricing mechanisms to address the resource trading and sharing within a federation. Mihailescu and Teo [117] present a reverse auction-based framework that uses dynamic pricing to allocate multiple types of shared resources in a cloud federation, where the sellers and buyers trading resources are rational users. The price auctions are carried out by a market-maker that collects the bids, selects the winner sellers for allocation based on the published price, and computes the actual payments based on the market supply of each resource type. Using simulations, they show that their dynamic pricing increases the buyer welfare and the percentage of successful buyer requests and allocated seller resources. Li et al. [118] address the profit maximization problem in a federation of selfish clouds under time-varying job arrivals and operational costs. They combine a truthful double auction mechanism with stochastic optimization techniques and design a dynamic algorithm for inter-cloud resource trading and scheduling. The proposed algorithm decides the best VM valuations and bids, schedules the received jobs onto VMs according to resources and SLA requirements and judiciously turns on and off servers based on electricity prices. Similarly, Toosi et al. [116] propose a financial option-based market model for resources pricing in cloud federation that helps providers increase their profits and mitigate the risk of QoS violations. The model uses option contracts between providers as a backup capacity for the reserved instances, which allows them to better exploit the underutilized reserved capacity for on-demand instances without concern to acquire resources when needs arise.

The above-discussed pricing mechanisms mostly focus on social welfare maximization in competitive federations and promote fairness and mutual benefits for involved actors. While such models are efficient to motivate providers to join a federation, they are not suitable for our research study. Our objective is the optimal allocation and partitioning of resource requests across the federation as a technique to maximize providers' revenues. The main objective of the two investigations differ considerably, even if they bear some similarities the presented pricing models are not adequate for our profit maximization problem. We believe that demand-oriented pricing approaches are more favorable for our optimization goal. The closest work to our study is found in [105] and proposes a utilization-based policy to dynamically update VM instance prices. The insourcing prices are set according to the providers' remaining capacity, which ensures load balancing between federated providers. The prices decrease with increasing idle resources to incite other providers to outsource their requests, and increase to discourage selection when the number of idle resources decreases. The pricing scheme of [105] will be combined with our allocation algorithms to adjust at each round the insourcing prices proposed by each provider to the federation members.

3.5 Conclusions

This chapter described the main research efforts in the area of profit-driven resource allocation in distributed and federated Clouds. We mainly focus on the service request type, the optimization goal and the underlying Cloud architecture as dimensions to classify the related work. The chapter also presents some relevant related work on market-based resource allocation and dynamic pricing models suitable for our study.

The main direction of this thesis is the design of efficient resource allocation models and algorithms for workload distribution and partitioning across a federation, while increasing providers' revenues and user satisfaction. The next chapters describe in detail our contributions to this research field.

Chapter 4

Exact ILP-Based Algorithm for Federating and Allocating Resources

Contents

4.1 Introduction

A trend in Cloud Computing is to extend cloud offerings to more complex services involving distributed resources across multiple infrastructures to meet users requirements.

To respond to these increasing and evolving workloads, Cloud Federation has been proposed as a key solution to random bursts in user demands (see section 2.3). Effective algorithms are needed to help providers define efficient resource management strategies to improve their profits and customer satisfaction. This includes the optimal workload distribution and resource sharing within the federation. Our thesis focuses on this optimization problem of federating and optimally allocating distributed resources amongst multiple infrastructure providers.

The chapter proposes an integer linear programming (ILP) formulation to increase providers' revenue according to the federation offerings through optimal "insourcing" and "outsourcing" decisions. Our goal is to provide an algorithm that automates the selection of a cost-effective distributed resource allocation plan that simultaneously satisfies user demand and networking requirements in such a distributed context. We address complex service requests requiring the provisioning of distributed resources and their networking to handle composite services. The proposed model aims at optimal partitioning of user requests across federation providers while respecting communication requirements between requests subsets. The algorithm performance evaluation and the identified benefits shed light on the conditions for a profitable federation and the efficiency of the proposed model in improving providers' profit.

Section 4.2 of this chapter describes the Cloud federation resource management model with our assumptions, including the request's graph modeling and assumed pricing schemes. Based on this system model, section 4.3 derives the exact ILP solution for distributed resource allocation in cloud federations. A number of valid constraints, equalities and inequalities are added to our mathematical formulation to speed up convergence and circumscribe much better the problem convex hull. Finally, section 4.4 reports the simulations and performance evaluations of the exact algorithm before concluding with a summary of main findings in section 4.5.

4.2 The System Overview

In this section, we introduce the assumptions and system model used in the design of the exact resources management algorithm. We first describe the federation scenario and the cloud resource requests modeling. Then, we point out the cooperation requirements and allocation costs that should be taken into account, while formulating the profit optimization problem, to achieve the above outlined objectives.

4.2.1 Cloud Federation Model and Assumptions

Figure 4.1 depicts the assumed cloud federation context involving m cloud infrastructure providers, $F = \{cp_1, cp_2, ..., cp_j, ..., cp_m\}$, cooperating in a peer-to-peer inter-cloud fashion as in [79]. The figure emphasizes cloud provider cp_j's view. Each provider has a finite amount of resources to split into a fraction to be used for internal use and another portion to share and make available as quotas to the federation. Each provider, at each round, will run our algorithm to find the optimal partitioning to serve users from local resources and accept requests for resources from federation members. The provider will also determine opportunistically the amount of requests to outsource to selected providers according to their respective proposed prices. As in public clouds (such as Amazon, Windows Azure and others), the providers will offer several resource instances types with emphasis on virtual machines with a preset amount of compute power (CPU) and memory (RAM) per instance type. The set of considered resource instances is denoted as $I = \{small, medium, large, xlarge, xxlarge\}$. To emphasize the limited amount of available resources per provider, the maximum capacity in compute power and memory available at home Cloud cp_j is capped at CPU_j and MEM_j. We use CPU_f^{Avail} and MEM_f^{Avail} to represent the quotas of compute power and memory each federated provider cp_f will cooperatively make available to the federation and d_f^{Avail} the period of their availability to other members.

When receiving its requests for resources allocations, each provider in the federation knows the quotas made available by other providers and the prices proposed per VM instance type. The providers will use this information and the proposed algorithm to split their requests into subsets to be served by selected members of the federation. The splitting and scheduling decisions have to lead to minimum costs and maximum revenues for the providers. Note that our work focuses exclusively on federation level optimization (outsourcing, insourcing, or local allocations). The proposed algorithm does not deal with optimal placement and consolidation at each provider [39, 40] to increase their hosting capacity. The work assumes that this is done locally and independently by each provider. The providers are also assumed to be interconnected by high performance and capacity links meeting applications' communications requirements. Our algorithm is not dealing with inter-cloud bandwidth provisioning and end-to-end networking for distributed requests components, but it will nevertheless aim at minimizing the load on the links and the induced cost when making partitioning and allocation decisions. While we believe that the inter-cloud networking issues are important, the management of providers' cooperation decisions under the hosting and communication costs criteria discussed in this work are complex enough in themselves to deserve separate treatment.

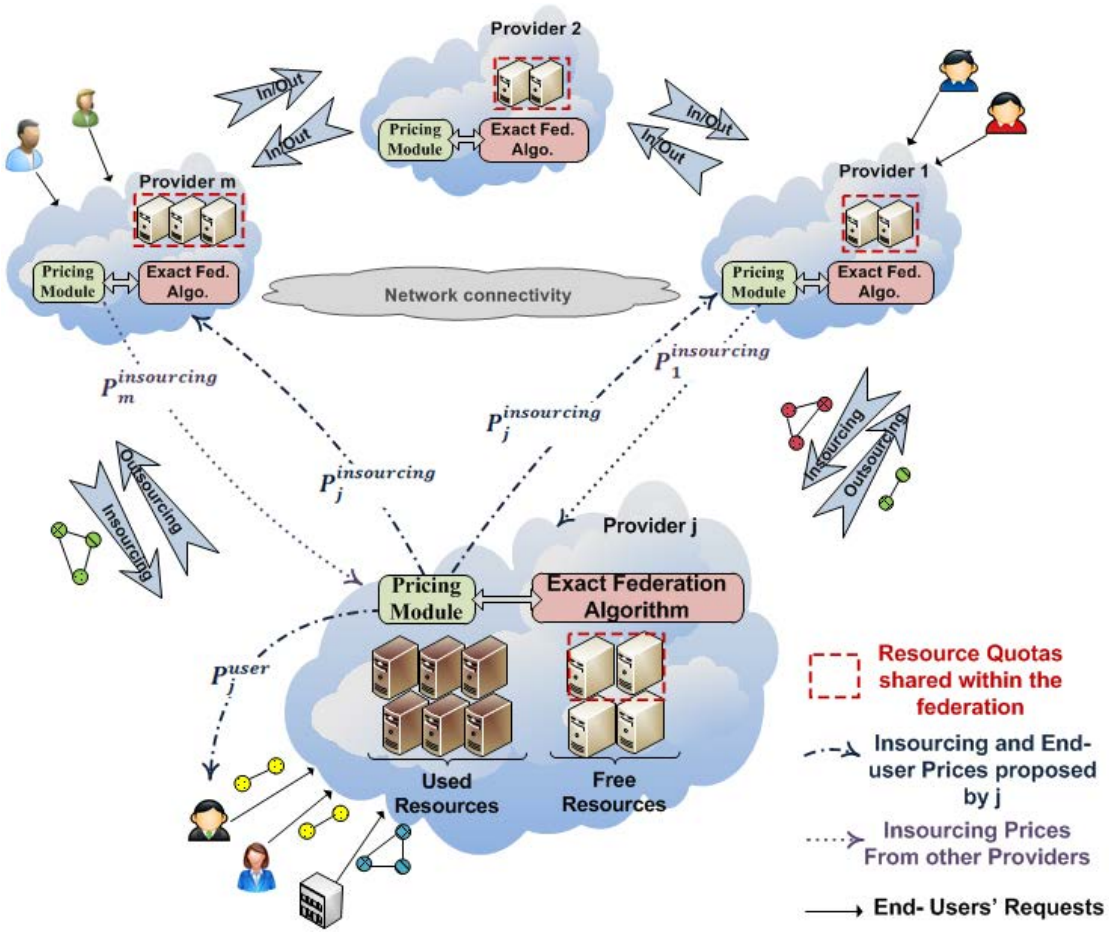


FIGURE 4.1: Cloud Federation Scenario

4.2.2 Resources Requests Model

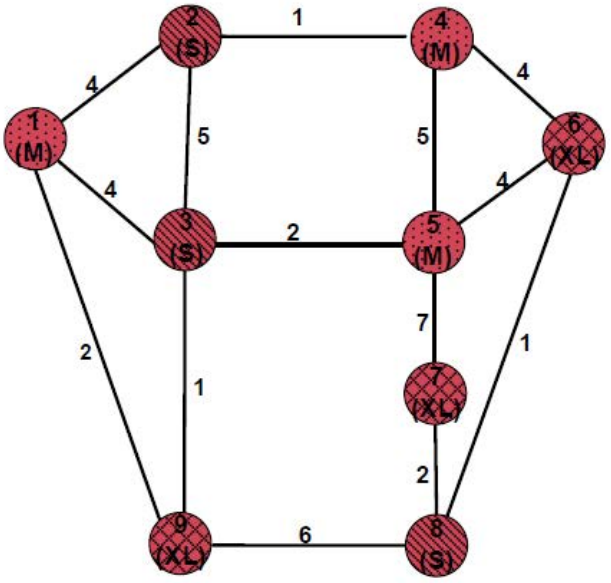
Each cloud provider cp_j can receive several batches of requests R, during a round, composed of requests from end users and from other providers. Cloud users can request resources for building complex applications, requiring distributed and connected elementary services, with different topologies and networking requirements. Each received request i can be modeled as an undirected graph $G_i = (V_i, Tr^i)$, where vertices V_i represent needed resources in terms of requested VM instances belonging to the set I of offered instances, and edges $Tr^i = (tr^i_{l,l'})_{1 \leq l,l' \leq |V_i|}$ reflect connections and traffic flow requirements between VMs l and l'. We assume that all virtual machines l ($l \in V_i$), associated to a user or a provider request i, are active during a specified duration $d_i = d_{i,l}, \forall l \in V_i$. Figure 4.2 provides an example of a received request G_i, with general topology defined by the inter-VMs traffic matrix Tr^i characterizing the amount of data units to be communicated between the 9 requested services (VMs) V_i, as illustrated respectively in Figure (4.2-(a)) and (4.2-(b)).

VMs	1	2	3	4	5	6	7	8	9
1	0	4	4	0	0	0	0	0	2
2	4	0	5	1	0	0	0	0	0
3	4	5	0	0	2	0	0	0	1
4	0	1	0	0	5	4	0	0	0
5	0	0	2	5	0	4	7	0	0
6	0	0	0	4	4	0	0	1	0
7	0	0	0	0	7	0	0	2	0
8	0	0	0	0	0	1	2	0	6
9	2	0	1	0	0	0	0	6	0

(a) Inter-VMs traffic matrix Tr^i

VMs	Instance type
1	medium
2	small
3	small
4	medium
5	medium
6	xlarge
7	xlarge
8	small
9	xlarge

(b) Requested VMs instances V_i



(c) Request graph modeling G_i

FIGURE 4.2: Resources Request Model

A typical class of distributed applications widely running on the Cloud is the "e-commerce applications". In this scenario, we can consider a user request for deploying a website for on-line shopping store. Figure 4.3 shows the modeling of such application composed of five types of servers: the load balancer redirecting incoming customers' requests to the suitable server, the web server providing the interface of the on-line store and handling HTTP requests and web pages browsing, the application server processing the users shopping requests (the preview of products catalog and prices, management of the virtual shopping cart, checking of the store stock, interaction with suppliers in case of product lack, interaction with shippers for customers' orders delivery, etc ...), the database server recording the site's information (available products and prices, customers' data, orders, etc ...), and finally the payment gateway/server ensuring secure payment process. Once deployed, these servers are hosted across different VMs with different capabilities satisfying each service's requirements, and communicate together to handle customers' shopping process. Some or all of these servers can be replicated to provide fault tolerance and disaster recovery solutions and/or to respect latency requirements. The user request can be presented to the Cloud provider as a template (or manifest) that specifies all needed resources and allocation constraints. Many other distributed applications like social networks, multi-players video gaming, simulations and data analysis applications can be easily represented using the underlying model.

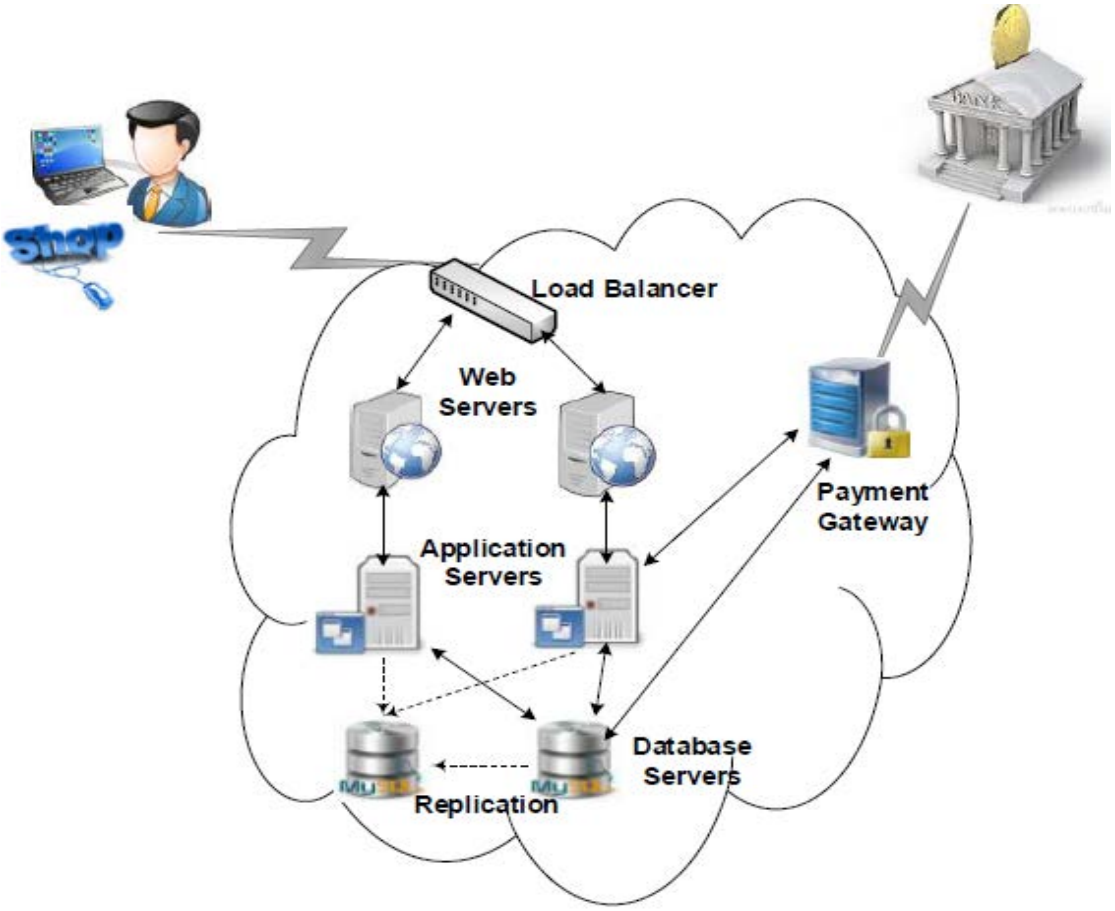


FIGURE 4.3: The example of an e-commerce website

4.2.3 Generic Pricing Model

In order to foster cooperation within the federation and make more profit at the same time, members will charge other providers less for insourcing actions than they would charge end users. Providers will earn a certain amount for each accepted request for resources. To remain competitive, the price charged to end users will be lower or close to the prices imposed by the cloud market. For each accepted VM l, a provider earns some revenue per resource unit represented by $P_{i,l}$ when satisfying a resource request i for a VM instance type l. The amount of resources providers share with the federation and their proposed prices for insourcing influence directly their potential profit improvements. Ideally, insourcing prices and quotas would be assessed dynamically based on workloads, available resources, future demands and cloud market fluctuations. This rather complex case is out of the scope of this work that limits the study by using a simple but realistic pricing mechanism [105] that captures the essence of the optimization problem. The pricing in [105] ensures load balancing between federated providers

depending on their remaining capacity. At each round, each provider will use the mathematical expression below to set their insourcing prices:

$$P_{type}^{insourcing} = \frac{Cap_{type}^{max} - Cap_{type}^{idle}}{Cap_{type}^{max}} * (P_{type}^{user} - Cost_{type}) + Cost_{type} \tag{4.1}$$

The expression takes into account the cost associated to the use of resources on a per VM instance type basis, the end user price and sets the proposed insourcing price according to the provider's idle resources. The proposed price decreases with increasing idle resources to incite other providers to outsource their requests and increases to discourage selection when the number of idle resources decreases. There are other costs than the cost of using local resources ($C_{j,l}^{local}$, the cost of a VM l served locally at provider cp_j) to take into account to derive the potential revenue improvements when joining a federation. The cost of outsourcing, when delegating some of the load to other providers, is one of these costs that is represented by $C_{f,l}^{out}$ for each VM l outsourced to a provider cp_f. The cost of networking VMs distributed across the federation infrastructures has to be included. The networking cost, to be paid to third party such as a network provider, of two VMs l and l' hosted respectively by providers cp_f and $cp_{f'}$ is represented by $C_{f,f'}^{net}$. We assume, however, that there is no cost for networking VMs hosted by the same provider or data center (i.e., $C_{f,f}^{net} = 0$) as this is embedded in the cost of the resources themselves (or VMs). We also assume an average loss in revenue when not accepting requests from users and federation members as this affects directly reputation and profit of the provider. To reflect this aspect we introduce in the model a penalty $\mathcal{L}_{rejection}^{penalty}$ representing the equivalent loss in revenue for each request rejected by the provider. This penalty can be used by the providers to set the weight and importance they give to their reputation. This has also the advantage of promoting cooperation in the federation.

With the generic pricing model, the outsourcing and networking costs expressions and the rejection penalty, we can mathematically formulate the revenue optimization problem and propose an exact algorithm to improve profit of the federation providers.

4.3 Exact Federation Allocation Algorithm

In this section, we present our ILP-based algorithm to assist IaaS providers, involved in a cloud federation, in adjusting their hosting and cooperation decisions in response to their workloads and available resources. We give the objective function to optimize under several linear constraints reflecting practical requirements of cloud resources allocation process.

Note that the proposed algorithm will run at each provider involved in the federation. Prior to running the algorithm that maximizes profit, each provider will use the generic pricing model of Equation (4.1) to set their insourcing prices at each round. These prices will feed the profit maximization algorithm when allocating resources. As depicted in Figure 4.4, the algorithm helps each provider cp_j $(j = 1, ..., m)$ partition received requests into subsets that will be hosted locally (at cp_j) or outsourced to other providers cp_f $(f \neq j)$ as well as select the insourcing requests from other providers to accept (served by cp_j on behalf of cp_f, $f \neq j$), while taking into account prices and quotas proposed by the federation members and the costs of resources and their networking. The goal of the algorithm is to optimally distribute the requests across the federation by maximizing revenues and minimizing costs at each provider to lead to improved profits for all federation members.

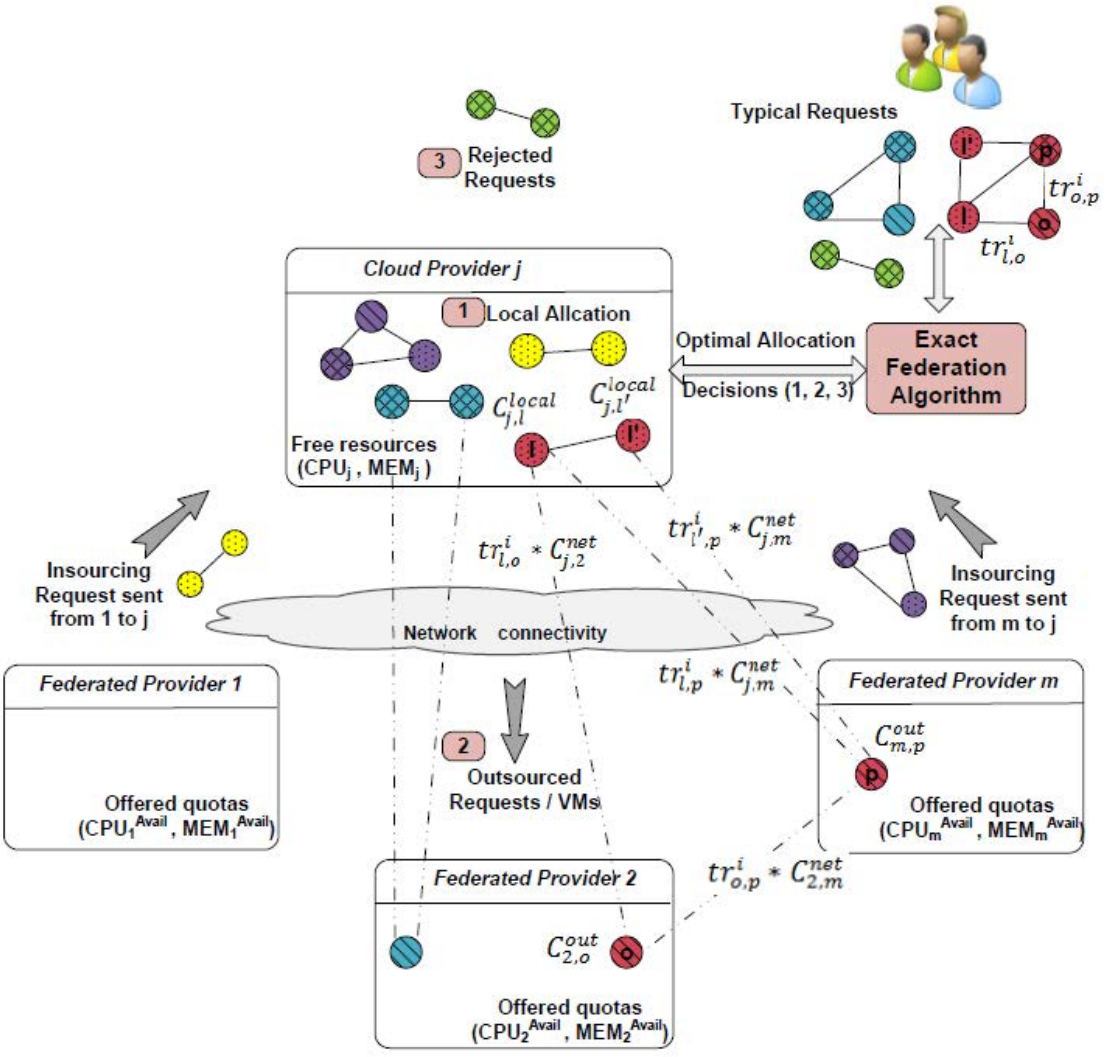


FIGURE 4.4: Decision Making Process

4.3.1 Linear Integer Program Formulation

We derive an integer linear program (ILP) for the problem for an arbitrary provider cp_j since the algorithm will run at each provider independently. The coupling is ensured by equation (4.1) that each provider uses to set its insourcing prices at each round. The objective function and the ILP should:

1. maximize profit achieved by provider cp_j when hosting (serving) typical and insourcing requests on local infrastructure (i.e. at provider cp_j);
2. enhance revenues by outsourcing requests to other members proposing advantageous prices when compared to the cost of local hosting and the price applied by provider cp_j to end users;
3. minimize networking costs when distributing, across the federation, VMs that require connectivity for their interactions;
4. maintain good reputation for the providers by minimizing the number of rejected resource requests.

To reach these goals, we define a number of boolean and integer variables as listed for convenience in Table 4.1. The bivalent decision variable $x_{j,f,l}$ indicates if a VM l received from provider cp_f is accepted and served locally by provider cp_j. Note that $x_{j,j,l}$ (when $f = j$) represents user requests received by cp_j and served locally by cp_j. In order to differentiate between end users requests and other providers ($f \neq j$) requests, we introduce the set S_i. This set is equal to $\{j\}$ for end user requests and to $\{f/f = 1, ..., m; f \neq j\}$ for hosting (insourcing) requests from the other providers. The set S_i is used also to control the price $P_{i,l}$ that is fixed for end users since governed by the market, while dynamically set for providers requests. The dynamic price is updated at each allocation round by the cooperating federation providers themselves in order to achieve the highest possible profit. For outsourcing decisions, we use variable $x_{f,j,l}$ to indicate if VM l is outsourced by provider cp_j to another provider cp_f in the federation or not. Variable a_i is used to indicate if a resource allocation request i is accepted or rejected in order to minimize rejection rate to maintain good reputation for the provider.

Using these notations, we can formulate our objective function achieving the optimization goals cited earlier and making optimal partitioning and allocation decisions. The maximization of the profit gained from local allocations during a round can be expressed using the following equation:

$$maximize \left(\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} \sum_{f \in S_i} (P_{i,l} - C_{j,l}^{local}) \cdot x_{j,f,l} \cdot d_i \cdot \Delta t \right) \tag{4.2}$$

Notation	**Meaning**
m	Number of providers in the federation.
j, f, f'	Are used to designate federation providers. cp_j refers to the Home Cloud and cp_f,$cp_{f'}$ to other federated members.
i	A resource request received from end-user or other providers.
R	The request batch composed of several received requests (from end-users and providers) to be handled together, $R = \cup_i \{i\}$.
V_i	The set of VMs desired by a request i. l and l' refer to VMs in this set ($l, l' \in V_i$).
cpu_l, mem_l	The resources requirements needed by VM l in terms of compute power and memory resources respectively.
d_i	The execution time of request i (request lifetime).
$tr^i_{l,l'}$	Traffic to be exchanged between VMs l and l' of request i.
CPU_j	Maximum CPU capacity on cp_j's local infrastructure.
MEM_j	Maximum Memory capacity on cp_j's local infrastructure.
CPU_f^{Avail}	CPU quotas shared by provider cp_f in the federation.
MEM_f^{Avail}	Memory quotas shared by provider cp_f in the federation.
d_f^{Avail}	The availability duration of provider cp_f's offered quotas.
$C^{net}_{f,f'}$	The networking unit cost between providers cp_f and $cp_{f'}$.
$C^{local}_{j,l}$	The local hosting cost of VM l.
$C^{out}_{f,l}$	The outsourcing cost of VM l among provider cp_f.
$P_{i,l}$	The unit price per each satisfied VM l of request i.
$\mathcal{L}^{penalty}_{rejection}$	The average revenue loss (penalty) for each request rejected by the provider.
S_i	Is the set of actors having sent the request to provider cp_j. According to our modeling, $S_i = \{j\}$ if i is an end-user request, and $S_i = \{f = 1, ...m; f \neq j\}$ if i is an insourcing request from provider cp_f.
Variables	**Definition**
$x_{j,f,l}$	A binary variable. $x_{j,f,l} = 1$ if the VM l has been received by the provider cp_j from cp_f and allocated locally, and 0 otherwise.
$x_{f,j,l}$	A binary variable. $x_{f,j,l} = 1$ if the VM l has been outsourced by the provider cp_j to the provider cp_f, and 0 otherwise.
$y^j_{f,f',l,l'}$	A binary variable. $y^j_{f,f',l,l'} = x_{f,j,l} \cdot x_{f',j,l'}$.
a_i	A binary variable. $a_i = 1$ if the request i has been accepted and 0 otherwise.

TABLE 4.1: Notations and Variables

Finding the best profit improvement through outsourcing virtual resources can be formulated as follows:

$$maximize \left(\sum_{i=1}^{|R|} \sum_{f=1, f \neq j}^{m} \sum_{l=1}^{|V_i|} (P_{i,l} - C_{f,l}^{out}) \cdot x_{f,j,l} \cdot d_i \cdot \Delta t \right) \tag{4.3}$$

The prices in equations (4.2) and (4.3) are updated using the pricing model of equation (4.1). The price $P_{i,l}$ in both terms (or expressions) is updated by provider cp_j in each round to set the new price when serving insourcing requests from other providers. This price is fixed for end users. In the second equation (4.3), representing revenue of outsourcing actions, the cost $C_{f,l}^{out}$ is the outsourcing cost applied by other providers to provider cp_j for outsourced requests. This cost, corresponding to the proposed insourcing price by other providers, is also updated by these providers ($f \neq j$), using equation (4.1) based on their respective resources usage and condition.

The networking costs, induced by partitioning requests across different providers, is minimized by adding this expression to the objective function:

$$minimize \left(\sum_{i=1}^{|R|} \sum_{f=1}^{m} \sum_{f'=1}^{m} \sum_{l=1}^{|V_i|} \sum_{l'=1, l>l'}^{|V_i|} tr_{l,l'}^{i} \cdot C_{f,f'}^{net} \cdot x_{f,j,l} \cdot x_{f',j,l'} \cdot \Delta t \right) \tag{4.4}$$

To avoid the resulting quadratic formulation, we define the binary decision variable $y_{f,f',l,l'}^{j}$ to linearize the expression in (4.4):

$$y_{f,f',l,l'}^{j} = x_{f,j,l} \cdot x_{f',j,l'} \tag{4.5}$$

This gives the following linear equation:

$$minimize \left(\sum_{i=1}^{|R|} \sum_{f=1}^{m} \sum_{f'=1}^{m} \sum_{l=1}^{|V_i|} \sum_{l'=1, l>l'}^{|V_i|} tr_{l,l'}^{i} \cdot C_{f,f'}^{net} \cdot y_{f,f',l,l'}^{j} \cdot \Delta t \right) \tag{4.6}$$

Finally, the minimization of the requests rejection rate during the allocation round can be expressed using:

$$minimize \left((|R| - \sum_{i=1}^{|R|} a_i) \cdot \mathcal{L}_{rejection}^{penalty} \cdot \Delta t \right) \tag{4.7}$$

The global objective function, combining all the stated optimization goals and criteria, is given by equation (4.8). This optimization is subject to several linear and integrity

constraints expressed respectively by equations (4.9) to (4.19) and equation (4.20). The introduced mathematical equalities and inequalities for the federating resources problem, formulate the conditions on the selected resources and providers that have to be respected when splitting requests across the federation. These linear constraints will speed up to some extent the convergence time of the exact approach, by reducing the search space for the optimal allocation decisions enhancing providers' revenues. The first two constraints (4.9) and (4.10) express resources limitation constraints so local allocations do not exceed the available maximum capacity in compute (CPU_j) and memory (MEM_j) at provider cp_j. Inequalities (4.11) and (4.12) make sure that outsourcing allocations remain below the quotas made available by providers to the federation. Inequality (4.13) guarantees that quotas are available during the entire lifetime of outsourced requests (or service time). Inequality (4.14) and equality (4.15) ensure that accepted requests are satisfied. Constraint (4.14) ensures that each VM is allocated to one and only one host. This also means that a satisfied VM request is exclusively served locally or outsourced to only one provider. Constraint (4.15) makes sure that all VMs associated to a request are allocated so the request is completely fulfilled otherwise the request is rejected. The family of constraints expressed by equality (4.16) prevent loops of insourcing and outsourcing actions. Insourcing requests from a provider cp_f ($f \neq j$) will not be outsourced back by provider cp_j to cp_f. Inequalities (4.17) and (4.18) define relations between the bivalent variables $x_{j,f,l}$, $x_{f,j,l}$ and $y^j_{f,f',l,l'}$. To guarantee provider cp_j gets at least some revenue when engaging in the federation, constraint (4.19) ensures that selected solutions always lead to a revenue higher than a minimum threshold R_0 ($R_0 \geq 0$, $R_0 = 0$ guarantees no revenue loss to the providers).

$$\begin{aligned}\max Z_j = \Bigg\{ \Bigg[&\left(\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} \sum_{f \in S_i} (P_{i,l} - C^{local}_{j,l}) \cdot x_{j,f,l} \cdot d_i \right) + \\ &\left(\sum_{i=1}^{|R|} \sum_{f=1, f \neq j}^{m} \sum_{l=1}^{|V_i|} (P_{i,l} - C^{out}_{f,l}) \cdot x_{f,j,l} \cdot d_i \right) - \\ &\left(\sum_{i=1}^{|R|} \sum_{f=1}^{m} \sum_{f'=1}^{m} \sum_{l=1}^{|V_i|} \sum_{l'=1, l'>l}^{|V_i|} tr^i_{l,l'} \cdot C^{net}_{f,f'} \cdot y^j_{f,f',l,l'} \right) \Bigg] - \\ &\left((|R| - \sum_{i=1}^{|R|} a_i) \cdot \mathcal{L}^{penalty}_{rejection} \right) \Bigg\} \cdot \Delta t \end{aligned} \tag{4.8}$$

Subject To:

$$\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|}\sum_{f\in S_i} cpu_l \cdot x_{j,f,l} \leq CPU_j \tag{4.9}$$

$$\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|}\sum_{f\in S_i} mem_l \cdot x_{j,f,l} \leq MEM_j \tag{4.10}$$

$$\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|} cpu_l \cdot x_{f,j,l} \leq CPU_f^{Avail} \quad \forall f = 1,...,m;\ f \neq j \tag{4.11}$$

$$\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|} mem_l \cdot x_{f,j,l} \leq MEM_f^{Avail} \quad \forall f = 1,...,m;\ f \neq j \tag{4.12}$$

$$\begin{gathered} d_i \cdot x_{f,j,l} \leq d_f^{Avail} \\ \forall i = 1 \in R;\ \forall l = \in V_i;\ \forall f = 1,...,m; f \neq j \end{gathered} \tag{4.13}$$

$$\sum_{f=1}^{m} x_{f,j,l} \leq 1 \quad \forall i \in R;\ \forall l \in V_i \tag{4.14}$$

$$\sum_{f=1}^{m}\sum_{l=1}^{|V_i|} x_{f,j,l} = |V_i| \cdot a_i \quad \forall i \in R \tag{4.15}$$

$$x_{f,j,l} = 0 \quad \forall i, \{i \in R \mid S_i \neq \{j\}\};\ \forall l \in V_i;\ \forall f \in S_i \tag{4.16}$$

$$\begin{gathered} x_{f,j,l} + x_{f',j,l'} - y^j_{f,f',l,l'} \leq 1 \\ \forall i \in R;\ \forall l = 1,...,|V_i|;\ \forall l' = 1,...,|V_i|; l' > l; \\ \forall f = 1,...,m;\ \forall f' = 1,...,m \end{gathered} \tag{4.17}$$

$$\begin{gathered} \sum_{f'=1}^{m} y^j_{f,f',l,l'} \leq x_{f,j,l} \\ \forall i \in R;\ \forall l = 1,...,|V_i|;\ \forall l' = 1,...,|V_i|; l' > l; \\ \forall f = 1,...,m; \end{gathered} \tag{4.18}$$

Instance type	CPU	RAM (Gbytes)
small	1	1.7
medium	1	3.75
large	2	7.5
xlarge	4	15
xxlarge	8	30

TABLE 4.2: VM's instances types

$$\left[\left(\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|}\sum_{f\in S_i}(P_{i,l}-C^{local}_{j,l})\cdot x_{j,f,l}\cdot d_i\right)+\left(\sum_{i=1}^{|R|}\sum_{f=1,f\neq j}^{m}\sum_{l=1}^{|V_i|}(P_{i,l}-C^{out}_{f,l})\cdot x_{f,j,l}\cdot d_i\right)-\left(\sum_{i=1}^{|R|}\sum_{f=1}^{m}\sum_{f'=1}^{m}\sum_{l=1}^{|V_i|}\sum_{l'=1,l'>l}^{|V_i|}tr^i_{l,l'}\cdot C^{net}_{f,f'}\cdot y^j_{f,f',l,l'}\right)\right]\geq R_0 \tag{4.19}$$

$$\begin{gathered}x_{j,f,l}\ ,\ x_{f,j,l}\ ,\ y^j_{f,f',l,l'}\in\{0,1\}\\ \forall i\in R;\ \forall l=1,...,|V_i|;\ \forall l'=1,...,|V_i|;l'>l;\\ \forall f=1,...,m;\ \forall f'=1,...,m\end{gathered} \tag{4.20}$$

4.4 Performance Evaluation

4.4.1 Evaluation Environment

The performance of the exact federation algorithm was evaluated using the ILOG CPLEX library [150] and a custom discrete event simulator on an Intel Xeon server with a 2.53 GHz Quad Core Processor and 24 Gbytes of RAM. A number of geographically distributed cloud providers (in $[2, 20]$) operating over different time zones with various infrastructure sizes were drawn randomly to perform the assessment in order to span the optimization space. Homogeneous and heterogeneous federation scenarios and conditions were used to collect the performance results. Request batches (ranging in $[1, 20]$ requests) with Poisson arrivals having different rates emulated the providers' day and night workloads. Each request is composed of a random number of connected VMs ($[1, 10]$) organized in a graph with random topologies and a random required service time in [1 hour, 5 hours]. The VM instance type was also drawn randomly in $I = \{small, medium, large, xlarge, xxlarge\}$ with configurations summarized in table 4.2. Connected VMs exchange traffic in the 1 to 5 Gbytes range.

Instance	End-user Prices	Hosting Costs	Networking Costs
small	[0.040$, 0.060$]	$(0.5 * P_{s}^{user})$	[0.001$, 0.005$]
medium	[0.062$, 0.120$]	$(0.5 * P_{m}^{user})$	[0.001$, 0.005$]
large	[0.140$, 0.240$]	$(0.5 * P_{l}^{user})$	[0.001$, 0.005$]
xlarge	[0.260$, 0.480$]	$(0.5 * P_{xl}^{user})$	[0.001$, 0.005$]
xxlarge	[0.520$, 0.980$]	$(0.5 * P_{xxl}^{user})$	[0.001$, 0.005$]

TABLE 4.3: Allocation's prices and costs

The end-users prices are fixed and set according to standard on-demand pricing schemes such as Amazon EC2 [120]. The insourcing prices are set dynamically by providers using equation (4.1). In the evaluation, the unit cost per instance type is fixed to ($Cost_{type} = 0.5 * P_{type}^{user}$) for each VM hosted locally. The cost of inter-providers networking units is drawn from a specific interval [0.001$, 0.005$]. All costs and prices used for the evaluation are summarized in Table 4.3. The revenue threshold R_0 is set to 0 to make sure providers do not lose revenue. The rejection penalty $\mathcal{L}_{rejection}^{penalty}$ that represents the average loss per rejected request is fixed arbitrarily to 1$ without loss of generality.

4.4.2 Comparative Baselines Approaches

Since previous proposals on distributed allocation and federation of virtual resources can not be directly contrasted with our exact algorithm (see Chapter 3 reviewing related works in the literature), we resort to a comparison with three baseline approaches to highlight the benefits of federations and of the exact algorithm:

- **Non-Federated Approach:** The providers operate independently and rely only on their own infrastructure to serve users. Requests are rejected if there are no more free resources. This approach corresponds to the exact algorithm with infinite outsourcing costs and zero insourcing prices (no out/in-sourcing). This approach is used as a normalizing reference for overall comparison.

- **Non-Splitting Approach:** This approach allocates each request (graph of VMs) without any splitting only to the provider with the most advantageous pricing. This corresponds to the exact algorithm with infinite inter-providers networking costs since this will force assignment of an entire request to one and only one provider.

- **Only-IF-Full Approach:** The providers outsource requests to the most appropriate federation members only when their own infrastructure is full.

4.4.3 Evaluation Results

The evaluation is performed for two working days (48 hours) with one hour rounds for the dynamic price updates by providers. The assessment concerns profit improvements, requests acceptance ratio, resources utilization and algorithm convergence time. All results correspond to a 95% confidence interval shown only for few curves (Figure 4.9) in order not to overload figures.

4.4.3.1 Effectiveness of the Exact Federation Algorithm

Improvement of Profit and Acceptance ratio: The first assessment of the exact federation algorithm performance concerns its effectiveness in improving profit for federation providers and gain insight on the most appropriate conditions for this improvement. A federation of 5 providers homogeneous in available resources (1500 CPU and 6000 Gbytes of RAM) is used and evaluated with homogeneous and heterogeneous loads within the federation. The parameters used for this simulation are summarized in Table 4.4. For homogeneous loads, providers receive batches of requests at a rate of 4 batches/hour during the day and 1 batch/hour at night. For heterogeneous case, providers experience different batches arrival rates in line with their respective end-users prices (see Table 4.4). Figures 4.5 and 4.6 show clearly profit improvements for all providers. When the load is homogeneous, providers with the highest end user prices gain the most see providers 3 and 4 that respectively improve their profits by 42% and 35%. This is due to highest gain margins between their prices and those offered dynamically by other federation members, especially those with lower end user prices. The latter will propose advantageous insourcing prices, and will be over solicited by incourcing requests. They end up proposing much higher prices to tamper other providers when overloaded, but they will not be able to frequently resort to federation because their end users prices need to be higher than outsourcing costs (insourcing price proposals from others) to improve their profit. This can be confirmed in Figure 4.5 by observing that provider 5 (with lowest end user prices) achieves only 9% profit improvement and suffers an increased rejection rate of 20% (compared to the case when operating without federation) while providers 3 and 4 improve their acceptance rates (46% and 48% respectively).

The case of heterogeneous loads provides additional insight and shows as expected that providers with the lowest proposed user prices and the highest workloads will set their insourcing prices to even higher values within the federation to avoid being overwhelmed by the other providers and will be able to achieve a better tradeoff between profit and rejection rate. The federation is more balanced overall in heterogeneous conditions even if providers 1 and 5 achieve better gains (respectively 47% and 23% profit improvement

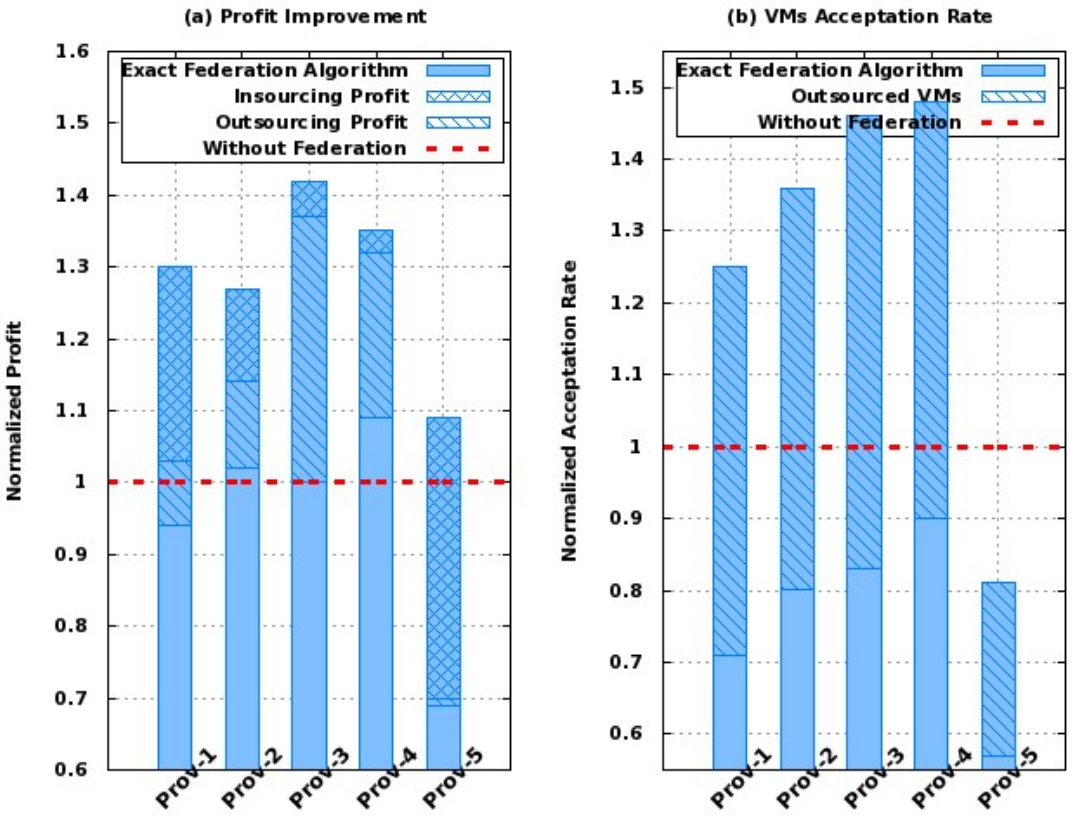


FIGURE 4.5: Impact of federation on providers' profit and acceptance rate (Same load)

and 69% and 3% request acceptance improvement) as depicted in Figure 4.6. In summary, the results highlight the importance of predicting future demands to derive more elaborate pricing schemes to improve profits even more for all providers. The pricing schemes should take into account both resource utilization and load prediction in Equation (4.1) to adjust insourcing prices. The pricing rounds when providers update their prices should also be set according to the workload arrival rate. This will be explored in future work that will combine the pricing with a load predictor.

Provider	CPU/RAM	(GMT)	Small prices	Medium prices	Large prices	XLarge prices	XXLarge prices	loads of Scenario 1	Loads of Scenario 2
Prov1	1500/6000	-8	0.044	0.070	0.140	0.280	0.560	4/h ; 1/h	6/h ; 1/h
Prov2	1500/6000	+1	0.047	0.077	0.154	0.308	0.616	4/h ; 1/h	3/h ; 1/h
Prov3	1500/6000	+9	0.061	0.101	0.203	0.405	0.810	4/h ; 1/h	4/h ; 1/h
Prov4	1500/6000	-4	0.058	0.095	0.190	0.381	0.761	4/h; 1/h	2/h ; 1/h
Prov5	1500/6000	+4	0.040	0.062	0.124	0.258	0.500	4/h ; 1/h	5/h ; 1/h

TABLE 4.4: Simulation parameters

Request splitting and outsourcing impact on profit improvements: Our exact federation algorithm using both insourcing and outsourcing and distributing requests across providers is compared to the baseline strategies in Figure 4.7. The results (normalized to the no federation strategy) are obtained for a federation of 5 providers receiving

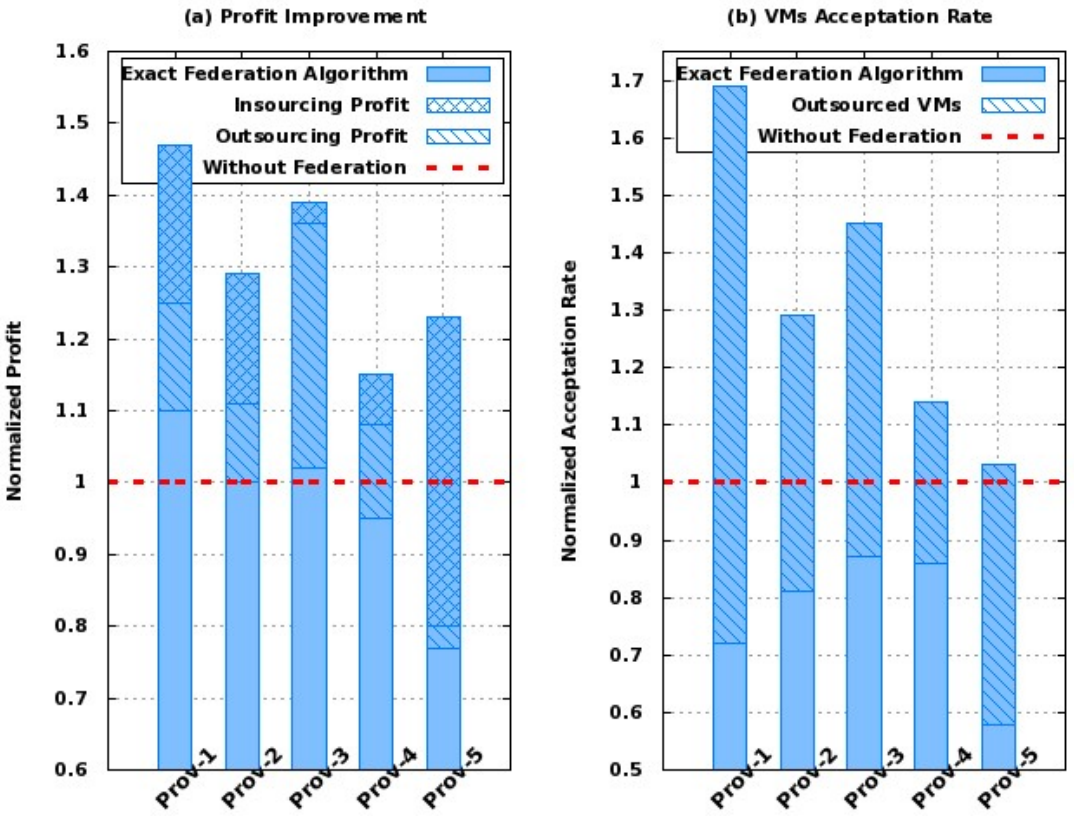


FIGURE 4.6: Impact of federation on providers' profit and acceptance rate (Heterogenous load)

batches of 20 requests per hour with 10 VMs per request and various connectivity ratios (all way to a full mesh scenario where all VMs are pairwise connected). Our exact algorithm that uses smart insourcing and outsourcing and thus optimizes allocations and partitioning of requests across the federation outperforms all other strategies. The exact algorithm improves by 8% and 11% the profit of provider cp_j compared to the "Non-Splitting" and "Only-IF-Full" approaches respectively. The improvement gap between our algorithm and the "Non-Splitting" approach decreases as the connectivity between VMs increases. The gap for only 2% for requests with full meshed VMs graphs is not surprising since the exact algorithm aims at minimizing networking costs by packing requests at each provider as much as possible thus behaves like the "Non-Splitting" strategy (allocates each complete request to the best provider only) in this case. The gap with the "Only-IF-Full" strategy is minimally affected (around 10%) by increasing connectivity. The "Only-IF-Full" approach has the drawback of filling the overall infrastructure of provider cp_j before resorting to any outsourcing.

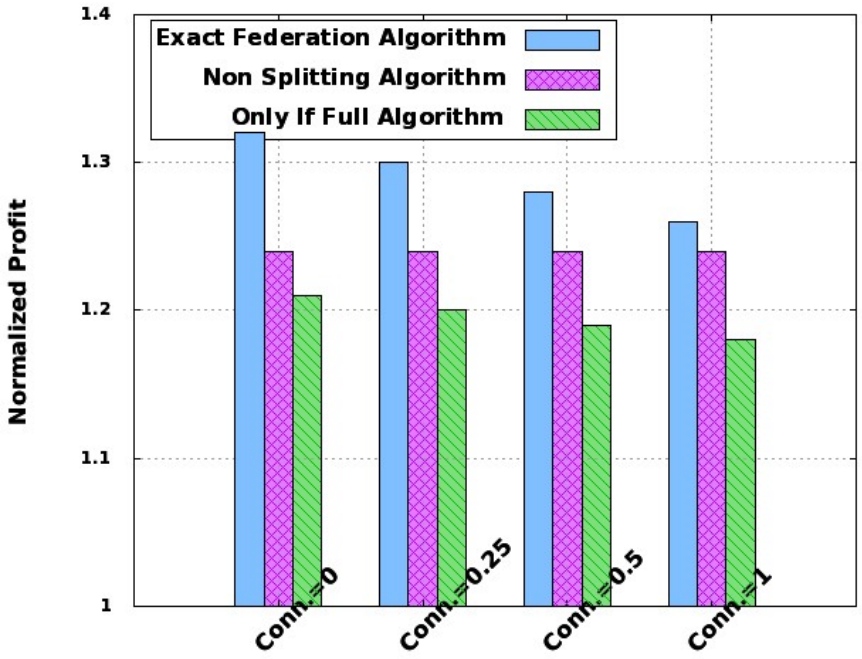


FIGURE 4.7: Splitting requests and smart outsourcing improve profit

4.4.3.2 Favorable Federation Conditions

Impact of load and federation size: This assessment aims at finding the conditions that are favorable for building and engaging in a federation. The optimal size of the federation and the profit improvements depend on the user base, the load (or total demands) users induce on the federation and the infrastructure size of each member. To shed some light into this question, three scenarios composed of providers with homogeneous characteristics are evaluated using the dynamic pricing scheme (equation 4.1) and our exact federation algorithm (equation 4.8), that each provider uses to respectively set their proposed prices and make outsourcing and insourcing decisions. The three scenarios correspond to a combination of provider sizes (in available CPU) and various arrival rates for Poisson distributed batches/hour. The evaluated combinations are tuples of (CPU units/provider, requests batches arrival rate/hour): ($CPU = 1000$, $\lambda = 20$), ($CPU = 2000$, $\lambda = 20$) and ($CPU = 2000$, $\lambda = 30$) as depicted in Figure 4.8. The achieved profits are recorded for selfish (serving their users locally) and cooperative providers (engaging in a federation) to evaluate the profit improvements when joining a federation for the three scenarios using formula in (4.21). The results report the achieved average gap in % in profit improvement as a function of federation size (number of providers), amount of resources made available to the federation and the received workloads.

$$Gap = \frac{Cooperative\ revenues - Selfish\ revenues}{Selfish\ revenues} * 100 \tag{4.21}$$

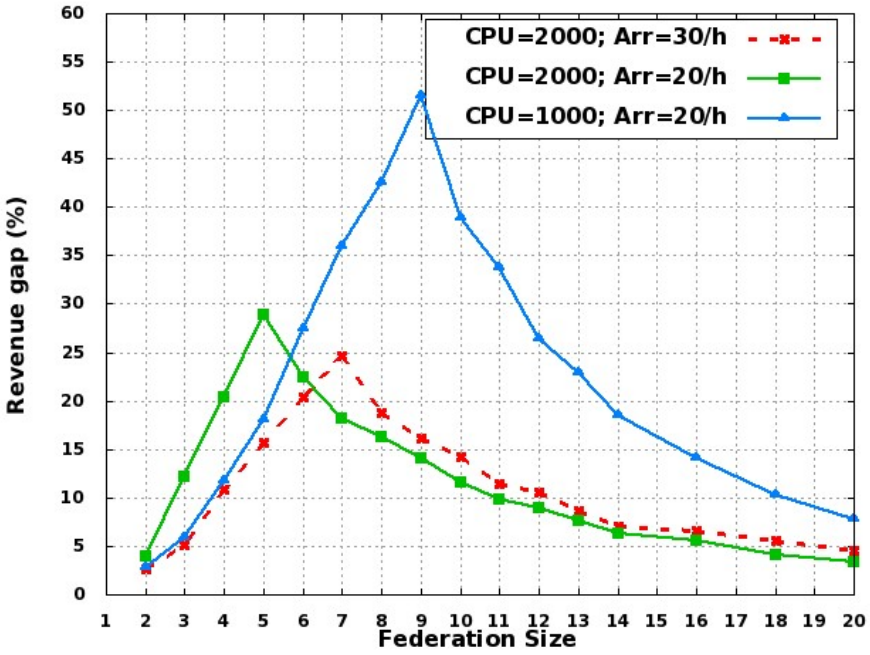


FIGURE 4.8: Average revenues evolution with the federation's size received load

As depicted in Figure 4.8, there is always an optimal size for a federation in terms of involved providers depending on the size of providers, requests workloads and shared resources. This optimal number is 9, 5 and 7 providers for the $(1000, 20)$, $(2000, 20)$, $(2000, 30)$ CPU/λ cases respectively. For higher amounts of available capacity per provider, the peak in profit improvement will occur at lower federation sizes (5 for $CPU = 2000$ versus 9 for $CPU = 1000$). When the workloads on the providers is higher, the optimal federation size will be higher since more providers are required to serve the overall higher load in the federation (7 for $\lambda = 30$ versus 5 for $\lambda = 20$). Beyond these optimal number of providers, the profits will decrease as there is a finite amount of money from users to share among providers.

Impact of Provider Size: The size of the provider in terms of available and shared resources has also an impact on the federation that is evaluated using 5 heterogenous providers receiving the same load but having different infrastructure sizes ranging from 1000 to 5000 of CPU capacity. The revenue gaps between selfish (serving users locally) and cooperative providers (using the exact federation algorithm to collaborate) are recorded, to identify the benefits for a provider to join a federation as a function of relative size to other federation members. The gaps are computed using the formula

in (4.21). Table 4.5 shows that all providers improve their profit by joining federation but with gains that depend on the amount of resources they own and share in the federation. Providers with fewer resources (providers 1 and 3) achieve the highest gains (72.64% and 49.20% profit improvements respectively). Other providers (2, 3 and 5) with moderate and large infrastructures earn much less. Provider 5 with the largest infrastructure (5000 CPU) achieves the lowest improvement (around 4.67%). Clearly providers with large infrastructure will have less interest in joining federation compared to small providers. The federations have to be balanced either in provider sizes or in the long term (availability of resources from providers become balanced with time, e.g. complementarity in terms of geographical situation and time zones) to be beneficial to all members.

Provider	CPU	RAM (Gb)	Gap (%)
Prov-1	1000	4000	72.64
Prov-2	2000	8000	34.99
Prov-3	1500	6000	49.20
Prov-4	3000	12000	18.76
Prov-5	5000	20000	4.67

TABLE 4.5: Revenues gap between selfish and cooperative behaviors

4.4.3.3 Scalability of the Exact Algorithm

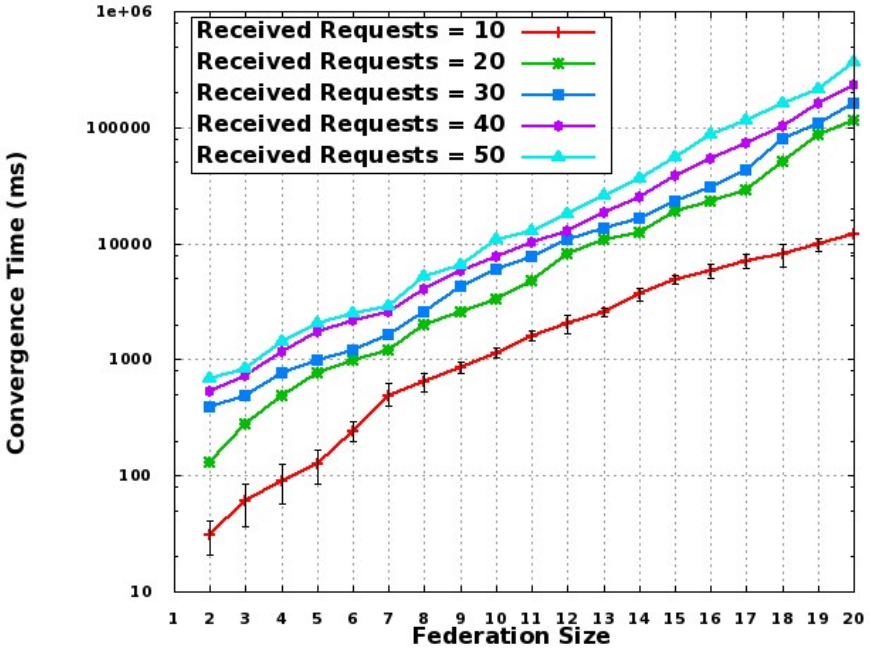


FIGURE 4.9: Impact of the number of received requests on the execution time of the exact allocation algorithm

Impact of requests batch size: In order to tune the size of batches (number of requests queued before the exact algorithm processes them jointly), a deeper understanding of the impact of the requests in arrival rate and graph structure is essential. Since the size of the batches should be selected as a function of the scalability of the exact algorithm, a simulation with different batch sizes were generated ($|R| = \{10; 20; 30; 40; 50\}$) for federations of 2 to 20 providers. The request size was set to 8 VMs with a connectivity of 50%. Figure 4.9 shows the exact algorithm convergence time performance for increasing federation size for the simulated scenario. For small federation sizes in line with the previous findings (see Figure 4.8) and limited number of batched requests, the algorithm is quite fast and achieves optimal partitioning and allocation in the order of second (30 msec to 1.1 sec for up to 10 providers and $|R| \leq 10$). For ($|R| > 10$) and larger federations with more than 10 providers, the convergence times increase to seconds and tens of seconds, and reach minutes for extreme cases in the simulation. The reported times can be used to set the target size for a request batch depending on the federation size and the user desired response time for resource requests (the limit accepted convergence time of the exact algorithm). Note also that the duration of rounds as stated earlier needs to be adapted dynamically according to the variation of loads at each provider through prediction of future demand. When both the optimal round duration and batch size (number of VM requests to lump into a batch for allocation) are known, all the needed information is available for optimal setting of the system parameters.

Impact of request topology: Figure 4.10 extends the analysis of the algorithm scalability by evaluating the impact of level of connectivity of VMs composing a request, by assessing performance for several request graphs: $|V_i| = 10$, $|V_i| = 20$ and $|V_i| = 30$ ranging from unconnected to fully meshed VMs. The results reported in the set of Figures 4.10 show an exponential increase in convergence time for increase request graph size and connectivity. For weakly connected request topologies with 10 VMs per request the partitioning and allocation can be achieved in the milliseconds range and the exact algorithm is quite efficient (see Figure 4.10(a)). In cloud services, typical requests sizes and topologies are rather small with partial connectivity. Hence, the algorithm performance is adequate for typical Cloud services with or without federation since the partitioning is achieved in times compatible with the quality encountered in current cloud services. For larger size of requests (20 and 30 VMs), Figures 4.10(b) and 4.10(c) show a significant increase in convergence times that reach tens of seconds and tens of minutes respectively depending on the degree of connectivity. The complexity and scalability of the algorithm are governed mostly by the request graph complexity and connectivity (third term in equation (4.8)). These high convergence times with large

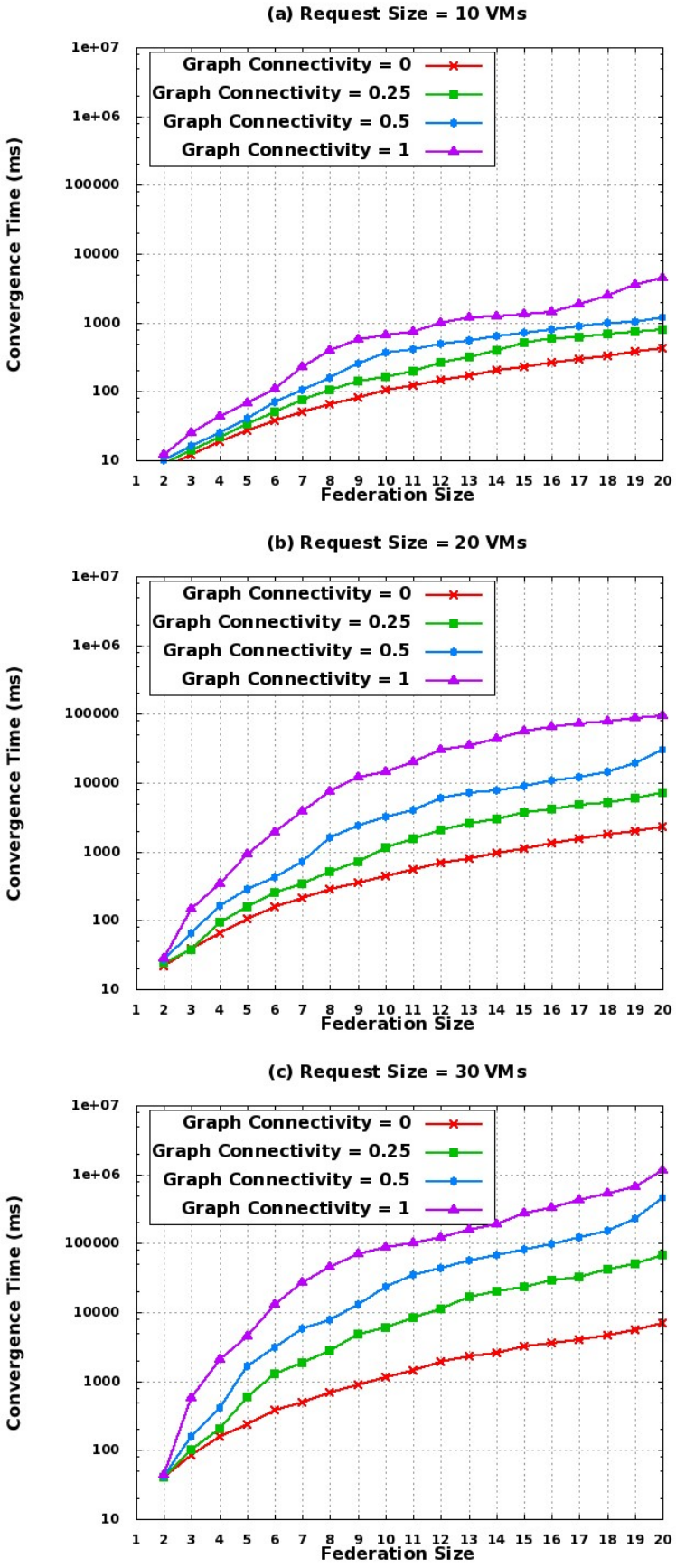


FIGURE 4.10: Impact of topologies of received requests on the execution time of the exact allocation algorithm

problem instances are impractical for on-demand services provisioning and can badly affect the reputations and profits of Cloud providers. Indeed, the response time spent by providers to deliver the requested services, is one of the key quality metrics that drive the decision of Cloud consumers when choosing the suitable providers to solicit for Cloud resources provisioning. This exponential performance degradation with large-scale instances, compels us to search for efficient heuristic algorithms that scale better with problem sizes and find optimal and near optimal solutions in polynomial times.

Based on all the performance evaluation results, the exact algorithm is a viable, exploitable and efficient solution for typical requests sizes and practical federations (usually less than 10 providers). With large connected graphs and federation sizes, the exact model encounters some difficulties in finding optimal solutions in prompt and practical computational times. Hence we need to resort to efficient heuristic algorithms to bring convergence times to convenient values for Cloud services today. The exact algorithm remains nevertheless important and useful for these problem sizes to compare and validate the performance of the proposed heuristics.

4.5 Conclusions

In this chapter, an exact algorithm for optimal resources request allocation across distributed providers is proposed and used to identify favorable conditions for joining a federation and assess the potential profit improvements for the involved providers. The presented ILP model is used by each provider to achieve optimal outsourcing and insourcing decisions to maximize its revenue. The algorithm is combined with a pricing scheme that updates at each round the proposed insourcing prices by each provider to other members, based on its resources usage and condition. In addition to maximizing providers' profits, the algorithm takes into account both networking costs imposed by the desired virtual machines connectivity and user satisfaction in terms of request rejection rate. Thanks to the use of a generic objective function englobing multiple optimization criteria and an efficient graph-based request modeling, the proposed model cloud be applied to other could service models (IaaS, PaaS, SaaS) supporting VM or container virtualization.

The exact algorithm complexity and scalability is governed by the size and connectivity of the virtual machines composing resource requests. With typical service requests (low complexity graphs and sizes) and few providers involved in the federation, the algorithm is shown to perform very well and find optimal solutions within seconds to tens of seconds. The proposed model is thus an efficient solution for small and medium-sized problem instances and achieves practical times in line with the performance experienced

by Cloud users today. The formulated ILP experiences exponential convergence times with large-sized instances. Therefore, we present in the next chapter a new efficient and scalable heuristic algorithm, based on Gomory-Hu requests transformation and Best-Fit matching, to improve performance when solving the resource federation problem.

Chapter 5

Graph Clustering based Algorithm for Resource Allocation in Cloud Federation

Contents

5.1 Introduction

The previous chapter introduced an exact ILP-Based algorithm for request partitioning and allocation across Cloud federation. The proposed model is shown to be efficient

in profit improvements and convergence times with small and medium-sized problem instances, but it exhibits exponential computational times for large-scale instances. With the growing adoption of Cloud services and increasing complexity of tenants' requests, cloud providers are often faced with challenging large-sized allocation problems involving multiple composite applications and heterogenous distributed resources. In addition, for the sake of resource usage optimization, cloud providers are inclined to combine several resource requests to be processed jointly in fixed allocation rounds. Batched requests can be modeled as a large composite graph with links between lumped subgraphs expressing placement and networking constraints, which leads to complex allocation tasks.

Moreover, even when handling moderate allocation problem instances, it would be beneficial for providers to further speed up their decision making process to satisfy quickly user demand. In fact, the provisioning response time ("request-to-deliver") has crucial effects on the provider's profit and reputation, since it is considered as one of the key quality metrics for cloud customers when selecting the appropriate providers for their workloads.

Efficient distributed resource allocation mechanisms are not only needed in Cloud federation, but also in many other multi-Cloud scenarios. For instance, private clouds involving multiple datacenters require such mechanisms to make better use of their infrastructures. In Hybrid Clouds, the "load bursting" problem needs effective strategies to distribute the applications' components and decide about subsets to be deployed in external Clouds and those to be hosted locally. The same need arises with Cloud Service Brokerages to select the best aggregation of resources satisfying user demand and selection criteria. These challenging resource allocation tasks are similar to our federation optimization problem. Minor adaptations of the model's parameters are needed to extend its usability to other multi-Cloud scenarios. This has motivated us to provide a generic efficient algorithm for federating and allocating resources across large-scale distributed Clouds, that is able to manage the plethora of requirements in practical computation times.

In this chapter, we present a new topology-aware resource allocation algorithm that utilizes a Gomory-Hu Tree based clustering algorithm and a best-fit matching strategy to make decisions. The combination of these two approaches is in our view suitable to meet the objectives of our optimization problem. The Best-Fit matching can minimize the hosting costs and request rejection rates by optimizing the utilization of available resources and quotas (seen as *bins*). On the other hand, the clustering approach allows to fulfill networking requirements and minimize inter-cloud communication costs by lumping together highly interacting VMs (seen as *items* to be packed into providers' infrastructures). The performance and solution quality of the proposed heuristic were

evaluated by using the exact ILP model as a benchmark for comparison. The simulation results proved the efficiency and scalability of the heuristic, that provides close to optimal solutions while improving convergence times by several orders of magnitude. The algorithm scales well with problem size and can handle large federations and complex requests in polynomial times.

The remainder of the chapter is organized as follows. The next section 5.2 describes our Networking-Cost-Aware Federating Resources Algorithm (***NCAFedRA***) as a scalable solution for the allocation problem. The complexity of the proposed heuristic is discussed in section 5.3. Section 5.4 provides a performance analysis of the simulation experiments, before concluding in section 5.5 with a summary of contributions and main results.

5.2 Networking-Cost Aware Federating Resources Algorithm (NCAFedRA)

Our ILP algorithm performs well with practical federations and typical requests sizes. However, like most exact solutions for NP-Hard problems (see computational complexity in 5.3), it does not scale with large-scale instances, especially with increase request graph size and connectivity (see section 4.4.3.3). To address this scalability issue, we resort to a heuristic allocation algorithm able to find efficient solutions in polynomial times. The *NCAFedRA* heuristic is based on request graphs clustering and consists of the following major steps visible in Algorithm 4:

1. Splitting the request graph into disjoint sets of VM-clusters with low inter-subgraph communication traffic, through a well-known *minimum k-cut* algorithm.
2. Assignment of these candidate partitions to the federation providers using a cost-based adaptation of the *Best-Fit matching* strategy.
3. Identification of the optimal k-cut leading to the minimum hosting and networking costs, by iterating the above steps across a range of possible k values.

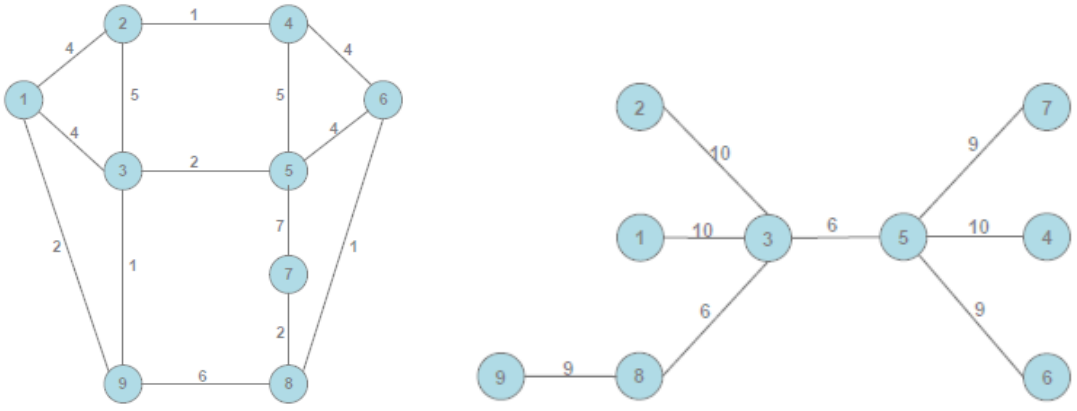
In the following subsections, we present the details of the designed algorithm stages. Since request partitioning is the key step of our approach, we start the heuristic description with the needed background on minimum k-cut and Gomory-Hu trees, before proceeding to the Best-Fit matching used for VMs and subgraphs assignment.

5.2.1 Request Graph Partitioning

This problem is a variant of k-cut graph partitioning problem. A k-cut is a set of edges $S \in E_G$ whose removal separates an undirected graph $G = (V_G, E_G)$ into k connected components. The minimum k-cut asks for the cut-set S with the minimum total weight (sum of capacities on the edges). This problem can be solved in polynomial and has a complexity of $\mathcal{O}(|V|^{k^2})$ for fixed k [151], but is NP-Complete if k is part of the input variables [152]. As the optimal k number of request graph partitions is not predetermined, we use a popular approximation algorithm with a ratio of $2 - 2/k$ [153], based on Gomory-Hu trees [154].

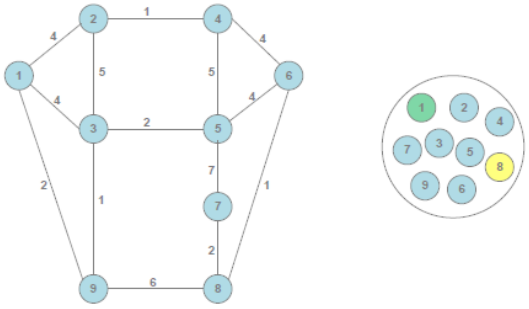
Definition 5.1. A Gomory-Hu tree (or cut tree) $T^{GH} = (V_T, E_T)$ of an undirected graph G, is a compact representation of the edge-connectivity between all pairs of its vertices. It is a weighted tree having the same nodes as G ($V_T = V_G$), but its $|E_T| = (|V_G| - 1)$ edges represent the minimum cuts between all vertex pairs in the original graph.

This tree can be built in polynomial time with only $(|V_G| - 1)$ max flow computations [154]. Figure 5.1-(b) shows an example of a cut tree T^{GH} for the undirected graph G shown in Figure 5.1-(a). For instance the weight 6 on the edge between nodes (3) and (5) in T^{GH} corresponds to the 3-5 minimal cut in G. The removal of this edge from T^{GH} will result into two disjoint connected components $(1, 2, 3, 8, 9)$ and $(4, 5, 6, 7)$ with a total flow of 6 across the cut-links in the original graph G. The minimum cut between any pair of vertices in G is equal to the minimum weight on the path connecting these two nodes in T^{GH}. For example, the 2-9 minimum cut is equal to $(min\ \{10, 6, 9\} = 6)$.

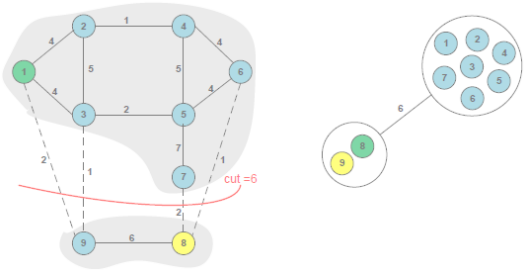


(a) Graph G (b) Gomory-Hu tree T^{GH}

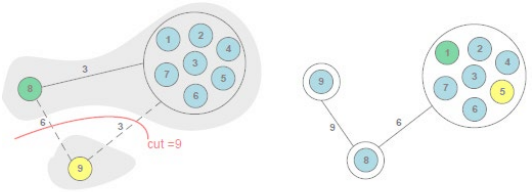
FIGURE 5.1: Example of Gomory-Hu Transformation



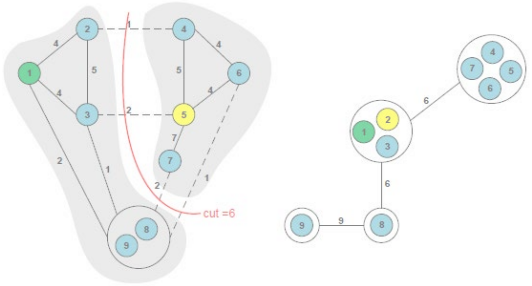
(a) The GH algorithm starts



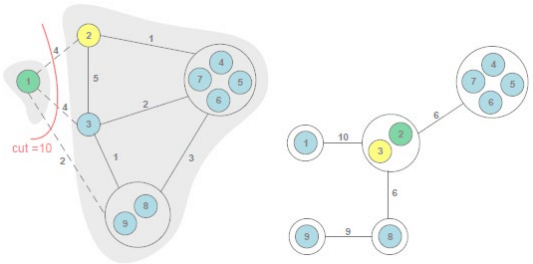
(b) Minimum 1-8 Cut



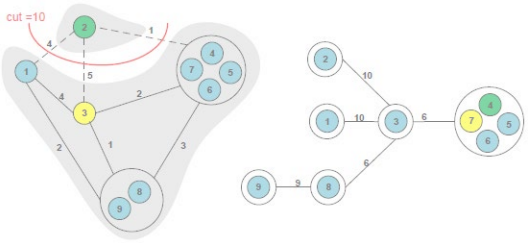
(c) Minimum 8-9 Cut



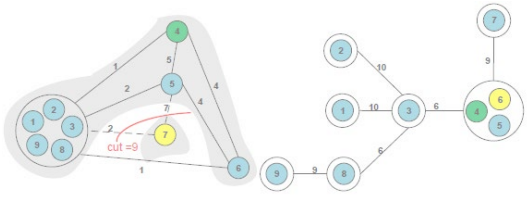
(d) Minimum 1-5 Cut



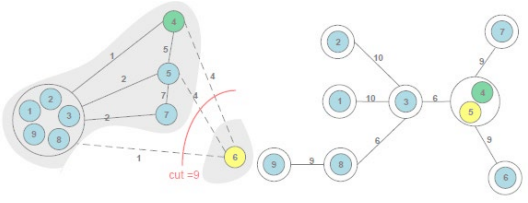
(e) Minimum 1-2 Cut



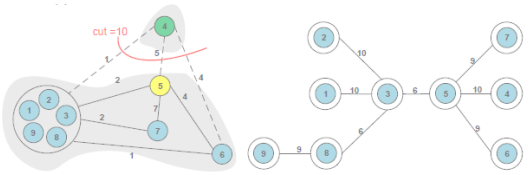
(f) Minimum 2-3 Cut



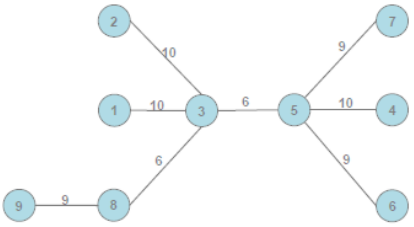
(g) Minimum 4-7 Cut



(h) Minimum 4-6 Cut



(i) Minimum 4-5 Cut



(j) The resulted GH Tree T^{GH}

FIGURE 5.2: Execution steps of the classical Gomory-Hu algorithm

5.2.1.1 Gomory-Hu Tree Construction

This subsection gives a summarized explanation of the cut-tree's construction for weighted undirected graphs. There are two well known algorithms, namely the Gomory-Hu's algorithm [154] and the Gusfield's algorithm [155]. Both algorithms consist in $(|V_G| - 1)$ maximum flow computations to determine all minimum cuts between the graph's vertices and lead to similar time complexities. They only differ in the used data structures, since the Gusfield algorithm uses the original input graph G to compute all cuts, while the Gomory-Hu algorithm contracts the graph G as iterations progress. We present in the following a formal description of the classical Gomory-Hu (GH) algorithm [154], used by our heuristic to get the tree representations of received requests. Figure 5.2 shows the execution steps of the GH algorithm applied to the graph G in figure 5.1-(a) to get its cut tree T^{GH} (figure 5.1-(b)).

The Gomory-Hu algorithm is detailed in 1. To distinguish the nodes of the input graph $G = (V_G, E_G)$ and those of the cut tree $T^{GH} = (V_T, E_T)$, we use the terms *"vertices"* and *"nodes"* to designate the elements of V_G and V_T respectively. In the rest of the thesis, these words are used interchangeably to denote the nodes (VMs) of the resource requests. At first, the algorithm initializes the cut tree T^{GH} to a single node V_T containing all vertices of the graph G. At each iteration, the algorithm picks from V_T a node X containing at least two vertices of V_G. For other connected nodes in $T^{GH} \backslash X$, it contracts the associated vertices in G into a same vertex and derives a new contracted graph G'. Two vertices s and t are chosen from node X to calculate the minimum s-t-cut in the generated graph G'. The nodes and edges of the current tree T^{GH} are updated according to the s-t-cut solution $\{A, B\}$. The node X is removed from V_T and split into two new nodes X_s and X_t containing respectively s and t. Other vertices of X are distributed between nodes X_s and X_t with respect to the s-t cut solution ($X_s = A \cap X$ and $X_t = B \cap X$). A new edge having a capacity equal to the minimum s-t cut weight is added to E_T to connect X_s to X_t. Already existing edges $e' = (X, Y)$ between X and other nodes Y in T^{GH} will be replaced with edges to connect Y to either X_s or X_t ($e' = (X_s, Y)$ if $Y \subset A$, or $e' = (X_t, Y)$ otherwise). The algorithm continues to handle the V_T nodes by iterating the above steps until all nodes contain a single vertex of V_G.

Note that the GH algorithm can result in different cut-tree representations for a same graph G, due to the variety of vertex permutations when calculating the $(|V_G| - 1)$ max flows and the potential multiplicity of minimum cuts between vertices. More details and explanation on the cut trees can be found in [154] and [155]. Other studies about experimental performances comparison of cut-tree algorithms and parallel implementations for faster convergence times can be found in [156–158].

Algorithm 1 Gomory-Hu Algorithm

Input: A weighted undirected graph $G = (V_G, E_G)$
Output: A Gomory–Hu Tree $T^{GH} = (V_T, E_T)$

$T^{GH} \leftarrow (V_T = \{V_G\}, E_T = \emptyset)$
while ($\exists X \in V_T$ such that $|X| \geq 2$) **do**
  Let $X \in V_T$ such that $|X| \geq 2$
  Let S_C is the set of nodes from V_T belonging to a connected component C of $T^{GH} \backslash X$
  Let $S = \{S_C \mid C$ is a connected component in $T^{GH} \backslash X\}$
  Construct the contracted graph $G' = (V_{G'}, E_{G'})$ such that $V_{G'} = X \cup S$ and $E_{G'} = E_G|_{X \times X} \cup \{(u, S_C) \in X \times S \mid (u, v) \in E_G;\ u \in X;\ v \in S_C\}$
  calculate the associated weights on $E_{G'}$
  choose two nodes $s,\ t \in X$ and find the minimum s-t cut in G'
  $\{A, B\} \leftarrow$ minimum s-t cut
  // update the Gomory-Hu tree vertices : Split X
  $X_s \leftarrow A \cap X$
  $X_t \leftarrow B \cap X$
  $V_T \leftarrow (V_T \backslash \{X\}) \cup \{X_s, X_t\}$
  // update the Gomory-Hu tree edges
  $e \leftarrow \{X_s, X_t\}$ with a capacity equal to the minimum s-t cut weight.
  for all edges $e' = (X, Y) \in E_T$ **do**
    if $Y \subset A$ **then**
      $e'' \leftarrow (X_s, Y)$
    else
      $e'' \leftarrow (X_t, Y)$
    end if
    $E_T \leftarrow (E_T \backslash \{e'\}) \cup \{e''\}$
  end for
  $E_T = E_T \cup \{e\}$
end while
replace each $\{v\} \in V_T$ by v and each $(\{u\}, \{v\}) \in E_T$ by (u, v)
return T^{GH}

5.2.1.2 Gomory-Hu Tree based Request Splitting

As discussed earlier in chapter 4, resource requests i from users and other providers are modeled by undirected weighted graphs $G_i = (V_i, Tr_i)$, where vertices V_i represent the requested VMs and edges Tr_i reflect traffic flows between VMs (see section 4.2.2). To find optimal request partitioning, our heuristic starts with applying the GH algorithm to the VMs graph to get a concise representation T_i^{GH} of the maximum flows between all service VMs, so that any $k \in [1; m]$ partitions can be obtained when needed. A k-cut of the request G_i is obtained by picking up the lightest $(k-1)$ edges from T_i^{GH}, which lead to k disjoint connected components. This weight-based sorting of the T_i^{GH}'s edges ensures that links with low traffic are removed earlier, so that highly connected VMs remain in the same partition assigned to a single provider, while disjoint VMs-clusters can be distributed across the federation with a minimum inter-cloud networking cost.

It is worth emphasizing that cut-trees were widely applied to solve many optimization problems in different research fields, such as scheduling problems [159], image segmentation [160] and social network analysis and mining [161], due to its efficient structure and properties. For our allocation problem, we have taken advantage of the connectivity property captured by the tree's edges to meet the cost minimization objective. Other properties can be used to solve the problem under different constraints. For instance, if dealing with QoS-oriented allocations, we can consider the VMs criticality through analyzing the vertices' degree (number of neighbors) in the cut-tree. VMs having a degree exceeding some threshold can be secured through restricted allocation decisions (local hosting only, replication for fault-tolerance, etc...).

Figure 5.3 illustrates the splitting of a complex graph G_i requiring several networked VMs into 3 subsets, through the removal of the two lightest edges from the corresponding cut-tree T_i^{GH}. After request partitioning, the heuristic should select for each candidate partition the suitable provider to allocate the needed resources according to hosting and networking costs. This resources assignment is conducted using a customized best-fit matching algorithm that aims to minimize the overall allocation cost and make better utilization of available resources. Before describing this assignment procedure, let us introduce some definitions and terminologies used in the remainder of this chapter.

Definition 5.2. A *VM cluster* V_c^{clus}, identified by a unique ID ($id = c$), is a group of VMs $l \in V_i$ and their networking. The list of all VM-clusters resulting from the k-cut splitting of request i is denoted by $LCs_i^k = \{V_1^{clus}, ..., V_c^{clus}, ..., V_k^{clus}\}$. The relation between LCs_i^k elements is given by:

$$\begin{cases} V_c^{clus} \cap V_{c'}^{clus} = \emptyset \quad \forall c, c' \leq k;\ c \neq c' \\ \underset{c}{\cup} V_c^{clus} = V_T = V_G = V_i \end{cases}$$

The terms *clusters*, *subsets*, *partitions* and *subgraphs* are used interchangeably to refer to these sets of connected VMs.

Definition 5.3. An *edge-cut* (or *cut*) $e_{(l,l')}^{cut}$, is a link connecting two nodes in the Gomory-Hu tree T_i^{GH}. This naming is used to avoid ambiguities with VMs links in the original graph G_i. The list of all $(|V_i| - 1)$ edges in T_i^{GH} is denoted by $cuts_i^{GH} = \{e_{(l,l')}^{cut}; l, l' \in V_i\}$. Each edge-cut $e_{(l,l')}^{cut}$ is defined by its endpoint nodes l and l' and its weight $bw_{l,l'}^{cut}$ corresponding to the maximum flow exchanged between VMs l and l'.

Definition 5.4. The set $cut_{remov}^k = \{e\_remov_{(c,c')}^{cut}; c, c' \leq k\}$ denotes the list of $(k-1)$ edge-cuts, whose removal from $cuts_i^{GH}$ split the request i into k disjoint clusters LCs_i^k. Each removed cut $e\_remov_{(c,c')}^{cut}$ is defined by its endpoint clusters V_c^{clus} and $V_{c'}^{clus}$ ($\in LCs_i^k$) and its weight $bw_{c,c'}^{cut}$. The relation between the sets cut_{remov}^k and $cuts_i^{GH}$ is

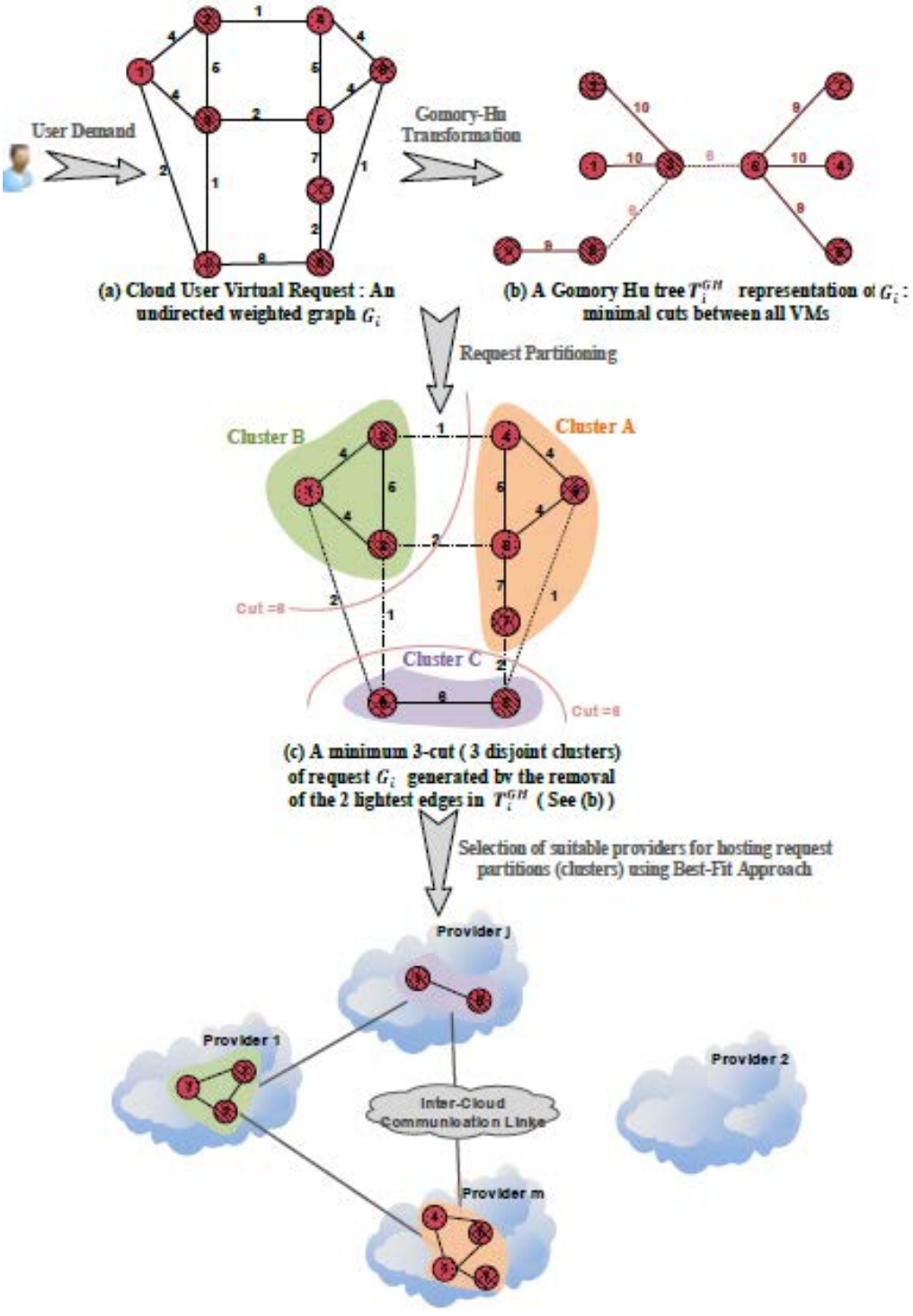


FIGURE 5.3: Request Splitting and Allocation across the Federation

defined as follows:

$$\begin{cases} cut^{k}_{remov} \subseteq cuts^{GH}_{i} \\ \forall e\_remov^{cut}_{(c,c')} \in cut^{k}_{remov}\ ;\ \exists e^{cut}_{(l,l')} \in cuts^{GH}_{i} \mid l \in V^{clus}_{c}, l' \in V^{clus}_{c'},\ and\ bw^{cut}_{c,c'} = bw^{cut}_{l,l'} \end{cases}$$

5.2.2 Cost Metric Computation

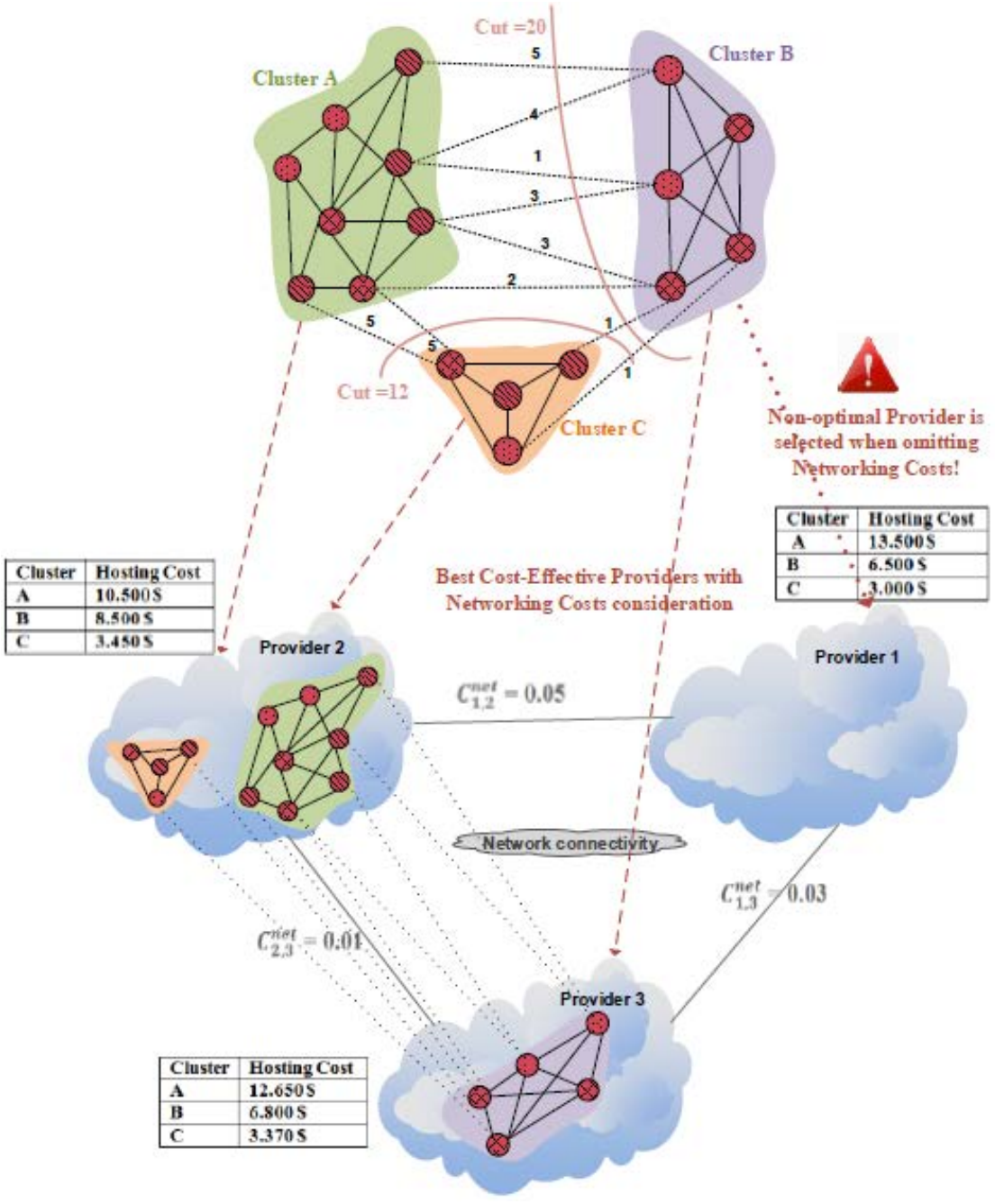


FIGURE 5.4: Providers' Selection based on the Aggregate Cost

Handling composite services requires special attention to the networking costs and requirements to achieve profitable allocation decisions. In addition to an efficient request partitioning to minimize the load on transit links, the inter-cloud communication costs must also be considered when making decisions. The example illustrated in Figure 5.4 highlights the impact of omitting networking costs on the solution quality. In such case, the clusters A, B and C are assigned to providers cp_2, cp_1 and cp_1 respectively, which leads to a hosting cost of 20\$ and a networking cost of $(28 * 0.05) = 1.4$\$ between distributed VMs, given a total of 21.4\$. This allocation plan is suboptimal and results in a revenue loss of 0.45\$ compared to the optimal solution considering both networking and hosting costs $(10.500 + 6.800 + 3.450 + 20 * 0.01 = 20.95$\$), as depicted in Figure 5.4. This revenue loss may have significant impact on the provider's profit, especially with higher traffic between distributed VMs and larger gap between inter-provider networking costs. In fact, we assume a generic cost reflecting links condition in terms of bandwidth availability, performance, latency and security levels. The consideration of networking costs in decision making becomes more difficult with increasing number of clusters and providers. To address this issue, we define an aggregate cost metric that approximates the overall charge of satisfying both computing and networking requirements of VM-clusters. Algorithm 2 summarizes the steps of calculating this cost metric.

The aggregate cost $C^{Aggregate}_{f,V_c^{clus}}$ of a virtual cluster V_c^{clus} when served by a provider cp_f is expressed by equation (5.1). It is equal to the sum of the hosting cost $C^{Hosting}_{f,V_c^{clus}}$ (5.2), required to accommodate all the cluster's VMs during the request lifetime d_i, and an approximate networking cost $C^{Networking}_{f,V_c^{clus}}$ for its interaction with other clusters. The term $cost(l)$ in (5.2) designates the cost of serving a VM instance l among the provider cp_f. This cost corresponds either to the local hosting cost $C^{local}_{j,l}$ if cp_f is the home Cloud $(f = j)$, or to the outsourcing cost $C^{out}_{f,l}$ applied by provider cp_f to provider cp_j $(f \neq j)$ for outsourced VMs.

$$C^{Aggregate}_{f,V_c^{clus}} = C^{Hosting}_{f,V_c^{clus}} + C^{Networking}_{f,V_c^{clus}} \tag{5.1}$$

$$C^{Hosting}_{f,V_c^{clus}} = \sum_{l \in V_c^{clus}} cost(l) \cdot d_i \tag{5.2}$$

$$C^{Networking}_{f,V_c^{clus}} = \sum_{V_n^{clus} \in N_c^{assign}} bw^{cut}_{c,n} \cdot C^{net}_{f,Ass(n)} + AC^{net}_f \cdot BW^{Ngh}_c \tag{5.3}$$

$$AC^{net}_f = \frac{\sum_{f'=1;f' \neq f}^{m} C^{net}_{f,f'}}{(m-1)} \tag{5.4}$$

Algorithm 2 Approximation of the aggregate cost metric

function: **Aggregate-Cost-Cluster**($V_c^{clus}, cp_f, LCs_i^k, cut_{remov}^k$):

Input: A cluster V_c^{clus}, a provider cp_f, the list of all k VM-clusters $LCs_i^k = \{V_1^{clus}, V_2^{clus}, \ldots, V_k^{clus}\}$, the set of removed $(k-1)$ cuts $cut_{remov}^k = \{e\_remov_{(c,c')}^{cut}; c, c' \leq k\}$ partitioning the original graph into k clusters

Output: The approximate overall cost $C_{f,V_c^{clus}}^{Aggregate}$ for serving the cluster V_c^{clus} on provider cp_f

Initialize: $C_{f,V_c^{clus}}^{Aggregate} \leftarrow 0$; $C_{f,V_c^{clus}}^{Hosting} \leftarrow 0$; $C_{f,V_c^{clus}}^{Networking} \leftarrow 0$
// Evaluate the hosting cost
calculate $C_{f,V_c^{clus}}^{Hosting}$ using equation (5.2)
// Evaluate the networking cost
if ($size(LCs_i^k) = 1$) **then**
  $C_{f,V_c^{clus}}^{Networking} \leftarrow 0$
else
  $N_c \leftarrow$ **List-of-neighbors**$(V_c^{clus}, LCs_i^k, cut_{remov}^k)$
  $N_c^{assign} \leftarrow$ **List-of-assigned**(N_c)
  $N_c^{unassign} \leftarrow$ **List-of-Unassigned**(N_c)
  if ($size(N_c^{assign}) = 0$) **then**
    calculate AC_f^{net} using equation (5.4)
    calculate BW_c^{Ngh} using equation (5.5)
    $C_{f,V_c^{clus}}^{Networking} \leftarrow AC_f^{net} * BW_c^{Ngh}$
  else
    // Evaluate the networking cost with assigned neighbors
    for ($V_n^{clus} \in N_c^{assign}$) **do**
      $Ass(n) \leftarrow$ **get-assigned-provider**(V_n^{clus})
      $C_{f,V_c^{clus}}^{Networking} \leftarrow C_{f,V_c^{clus}}^{Networking} + bw_{c,n}^{cut} * C_{f,Ass(n)}^{net}$
    end for
    // Evaluate the networking cost with non-assigned neighbors
    calculate AC_f^{net} using equation (5.4)
    calculate BW_c^{Ngh} using equation (5.5)
    $C_{f,V_c^{clus}}^{Networking} \leftarrow C_{f,V_c^{clus}}^{Networking} + AC_f^{net} * BW_c^{Ngh}$
  end if
end if
$C_{f,V_c^{clus}}^{Aggregate} \leftarrow C_{f,V_c^{clus}}^{Hosting} + C_{f,V_c^{clus}}^{Networking}$
return $C_{f,V_c^{clus}}^{Aggregate}$

$$BW_c^{Ngh} = \sum_{V_n^{clus} \in N_c^{unassign}} bw_{c,n}^{cut} \tag{5.5}$$

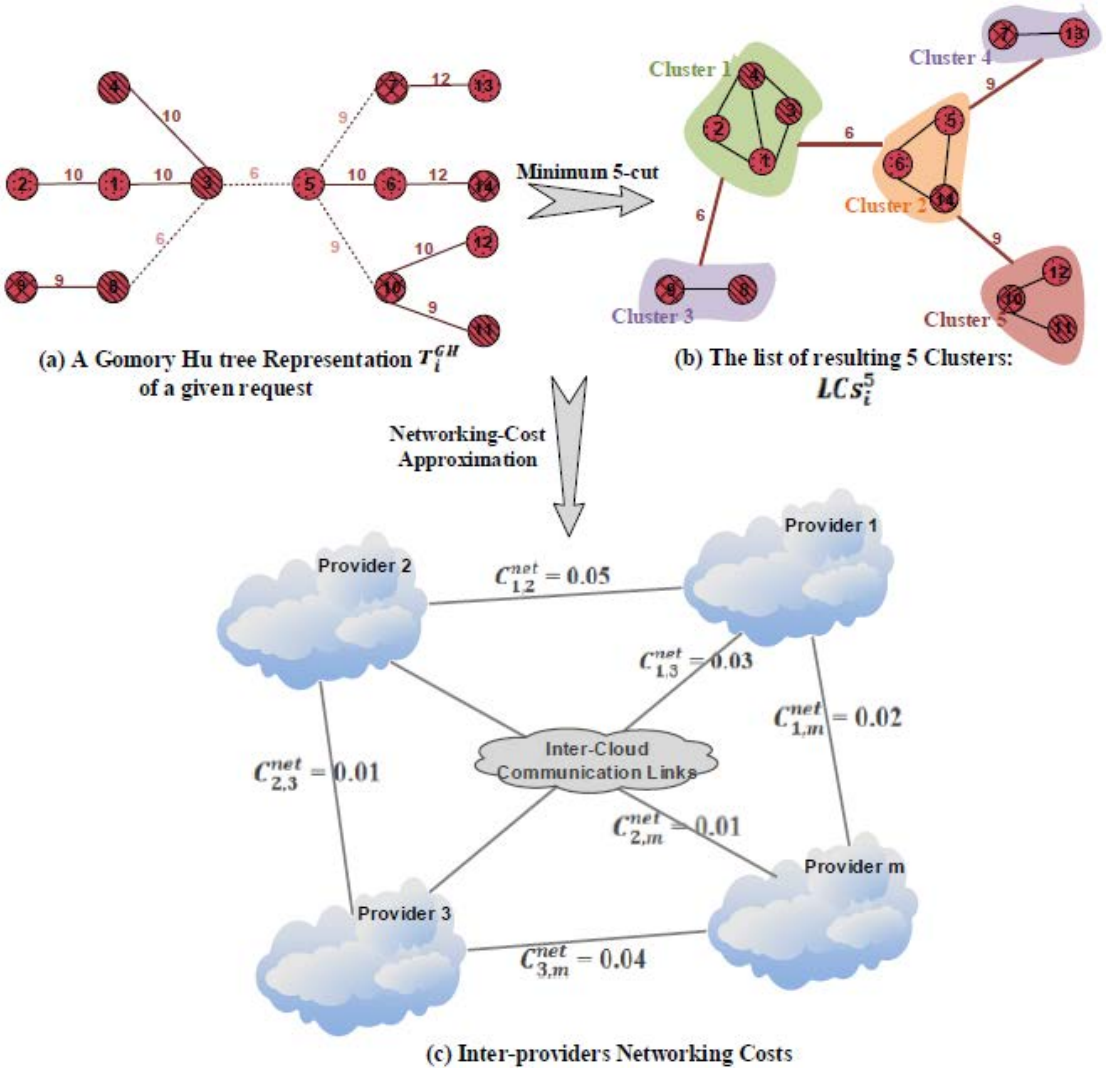


FIGURE 5.5: Networking cost Approximation

The networking cost $C_{f,V_c^{clus}}^{Networking}$ approximates the cost induced by interactions between cluster V_c^{clus} when served by cp_f and its neighbors distributed in the federation. This cost is set to 0 if there is a single cluster to allocate, otherwise it is estimated using Equation (5.3). As detailed in Algorithm 2, we start by finding the list N_c of neighboring clusters directly connected to V_c^{clus}, using the function **List-of-neighbors**$(V_c^{clus}, LCs_i^k, cut_{remov}^k)$. In figure 5.5, the list of neighbors of cluster V_2^{clus} is $N_2 = \{V_1^{clus}, V_4^{clus}, V_5^{clus}\}$. The N_c list is classified into two different subsets N_c^{assign} and $N_c^{unassign}$, containing respectively the neighboring clusters already assigned to some providers and those not assigned yet. If no cluster is assigned, the cost $C_{f,V_c^{clus}}^{Networking}$ is set to the product of the average networking unit cost AC_f^{net} between cp_f and other providers defined

by equation (5.4), by the total bandwidth BW_c^{Ngh} exchanged between V_c^{clus} and all its neighbors expressed in equation (5.5). Otherwise, $C_{f,V_c^{clus}}^{Networking}$ is calculated using equation (5.3), by first cumulating the costs of communication with already assigned neighbors, and adding an approximate cost of the interactions between V_c^{clus} and its unassigned neighbors as detailed above. In the example of figure 5.5, assuming that all neighbors of the first cluster $N_1 = \{V_2^{clus}, V_3^{clus}\}$ are unassigned, its approximate networking cost if served by provider cp_2 is equal to the product of $AC_2^{net} = 0.023\$ = ((0.05 + 0.01 + 0.01)/3)$, by the total bandwidth exchanged on the neighboring cuts $BW_1^{Ngh} = (6 + 6) = 12$, given a $C_{2,V_1^{clus}}^{Networking}$ cost of $0.276\$ = (12 * 0.023)$. Assuming now that the cluster V_1^{clus} is assigned to provider cp_2 for allocation, and we are handling the second cluster V_2^{clus}. In this case, $N_2 = \{V_1^{clus}, V_4^{clus}, V_5^{clus}\}$, $N_2^{assign} = \{V_1^{clus}\}$ and $N_2^{unassign} = \{V_4^{clus}, V_5^{clus}\}$. Using algorithm 2 and equation 5.3, the approximate networking cost of V_2^{clus} if served by provider cp_1 is equal to $C_{1,V_2^{clus}}^{Networking} = bw_{2,1}^{cut} * C_{1,2}^{net} + AC_1^{net} * BW_2^{Ngh} = (6 * 0.05) + (0.033 * (9 + 9)) = 0.894\$$.

The obtained aggregate cost $C_{f,V_c^{clus}}^{Aggregate}$ is used to drive the selection of suitable providers for request's partitions. The next steps consist in resource selection following a cost-aware best fit approach and identification of the optimal k-cut partitioning.

5.2.3 Cost-Aware Best-Fit Matching Algorithm

To distribute users' applications across the federation, we resort to an adaptation of the well-known Best-Fit algorithm [162]. The remaining hosting capacities within the federation, including local resources and shared quotas, are seen as ***bins*** to be filled. The set of k disjoint VMs-clusters resulting from request splitting represent the ***items*** to be packed. This matching policy was selected since it can achieve good performance in terms of resource utilization and item acceptance ratios compared to the classical Bin-Packing. This allows to improve providers' profits and maintain good reputation by reducing the number of rejected requests. To minimize the cost of federating resources, we use a cost-aware best-fit matching approach illustrated in Algorithm 3, that makes allocation decisions based on the aggregate cost metric calculated by Algorithm 2.

The assignment algorithm handles the k candidate clusters in a decreasing order of their total needed resources. Using equation 5.1, it estimates for each cluster the allocation costs among all providers to select the best one (having lowest cost and remaining capacity). The process continues until all k-cut clusters are assigned to target providers, and returns the assignment matrix to the main routine (Algorithm 4) to finalize the allocation task. If no provider could satisfy a cluster's requirements, the algorithm stops

Algorithm 3 Providers selection for hosting requests partitions

function: **Cost-Aware-Best-Fit-Assignment**$(LCs_i^k, F, cut_{remov}^k)$;

Input: The list of k clusters $LCs_i^k = \{V_1^{clus}, V_2^{clus}, \ldots, V_k^{clus}\}$ to allocate, the federation providers $F = \{cp_1, cp_2, ..., cp_f, ..., cp_m\}$ and their offerings, the set of removed $(k-1)$ cuts partitioning the graph $cut_{remov}^k = \{e\_remov_{(c,c')}^{cut}; c, c' \leq k\}$

Output: A cost-effective assignment plan $assign\_matrix[k]$ specifying the list of providers to host the k candidate clusters.

1: **Initialize:** $assign\_matrix[k] \leftarrow null$; $cost\_matrix[k,m] \leftarrow null$; $remain\_capacity\_matrix[m] \leftarrow null$; $nb_{Satisfied}^{clus} \leftarrow 0$;
2: **for** ($cp_f \in F$) **do**
3: $remain\_capacity\_matrix[f] \leftarrow$ **remaining-capacity**(f)
4: **end for**
5: Sort the list of VM-clusters LCs_i^k in decreasing order of their total needed resources.
6: **for** ($V_c^{clus} \in LCs_i^k$) **do**
7: boolean $assigned \leftarrow false$
8: **for** ($cp_f \in F$) **do**
9: $cost\_matrix[c,f] \leftarrow$ **Aggregate-Cost-Cluster**$(V_c^{clus}, cp_f, LCs_i^k, cut_{remov}^k)$
10: **end for**
11: Sort the list of providers F in increasing order of their aggregate allocation cost and remaining hosting capacities.
12: **for** ($cp_f \in F$) **do**
13: **if** cp_f has enough resources to host V_c^{clus} **then**
14: $assigned \leftarrow true$
15: $assign\_matrix[c] \leftarrow f$
16: update $remain\_capacity\_matrix[f]$
17: $nb_{Satisfied}^{clus} \leftarrow nb_{Satisfied}^{clus} + 1$
18: **break**
19: **end if**
20: **end for**
21: **if** ($assigned = false$) **then**
22: **break**
23: **end if**
24: **end for**
25: **if** ($nb_{Satisfied}^{clus} \neq k$) **then**
26: $assign\_matrix \leftarrow null$
27: **end if**
28: **return** $assign\_matrix$

iterations and returns a *null* value to the main process that will split the request into $(k+1)$ smaller subgraphs to fit available capacities (see section 5.2.4).

Applying the Best-Fit algorithm 3 to the example in figure 5.4 allows finding the optimal allocation plan for the 3-cut partitions. The first processed cluster A has an aggregate cost of $cost\_matrix[1] = [14.780\$; 11.460\$; 13.290\$]$ across the federation members, and hence is assigned to the provider cp_2. The aggregate costs of the second cluster B are equal to $cost\_matrix[2] = [7.500\$; 8.500\$; 7.000\$]$, and so it is assigned to cp_3. Finally the cluster C, having as aggregate costs $cost\_matrix[3] = [3.600\$; 3.450\$; 3.490\$]$, is attributed to cp_2. Thus, the assignment result for $LCs_i^3 = \{A, B, C\}$ is $assign\_matrix = [2, 3, 2]$, which is the optimal cost-effective allocation solution. Note that this step represents a selection phase without any effective allocation. These choices will be validated by the main algorithm 4 once costs criteria are checked and optimal k-cut is reached.

5.2.4 Description of the Heuristic Approach (NCAFedRA)

The proposed heuristic uses algorithms 2 and 3 as subroutines to find the optimal request partitioning into subsets to be hosted locally or outsourced to other providers. Algorithm 4 summarizes the key steps of the decision making process.

The heuristic starts with ranking the batch R of received requests in decreasing order of their potential revenues (selling prices) to prioritize profitable ones. If there are not enough resources across the federation to satisfy a request i, the algorithm rejects the demand and skips to the next item in R. Otherwise, it constructs the Gomory-hu tree of the graph $G_i = (V_i, Tr_i)$ to define the minimum-cuts $cuts_i^{GH}$ between all VMs. The algorithm iterates across a range of possible k values to find the best request partitioning leading to minimum costs. The federation size m defines the upper bound of k, since a request can be split at worst among all involved providers.

For each k value, the algorithm **Cost-Aware-Best-Fit-Assignment** (Algorithm 3) is used to find the best providers meeting the requirements of the candidate clusters LCs_i^k. The solution corresponds to a mapping function $\mathcal{M} : LCs_i^k \rightarrow F$, that associates for each VMs-cluster a hosting provider. If there is no mapping solution that satisfies all k clusters (a *null* result), the algorithm skips directly to the next k value to get smaller partitions fitting available quotas. Otherwise, the algorithm 4 evaluates the solution quality $assign\_sol$ in terms of costs criteria before validating it and effectively allocating resources for the request i. The current allocation cost $alloc\_cost_i^k$ (5.6) is compared with the minimum recorded cost $alloc\_cost_i^{best}$ corresponding to the $(k-1)$-cut partitioning. If it is higher, the algorithm exits iterations and selects the $K_{opt} = (k-1)$ partitions

Algorithm 4 NCAFedRA heuristic

Input: A batch of requests $R = \cup_i$, the list of federation providers $F = \{cp_1, cp_2, ..., cp_f, ..., cp_m\}$ and their offerings.

Output: A distributed allocation plan for requests R minimizing the overall costs.

Initialize: $accept\_matrix[\ |R|\ ] \leftarrow null$; $alloc\_plan[\ |R|, max(|V_i|)\ ] \leftarrow null$; $assign\_sol[m] \leftarrow null$; $LCs_i^{best} \leftarrow null$; $assign^{best}[m] \leftarrow null$;
Sort requests i in R in decreasing order of their profitability.
for ($i \in R$) **do**
  if (**total-capacity**$(i) \geq$ **total-remaining-quotas**(F)) **then**
    $accept\_matrix[i] \leftarrow rejected$
    $alloc\_plan[i, l] \leftarrow -1$, for all $l \in V_i$
  else
    boolean $solution\_found \leftarrow false$
    $alloc\_cost_i^{best} \leftarrow \infty$
    $profit_i^{best} \leftarrow -\infty$
    Construct the **Gomory-Hu tree** of i and obtain T_i^{GH} containing V_i nodes and $(|V_i| - 1)$ links $cuts_i^{GH}$
    Sort $cuts_i^{GH}$ by increasing weights
    for ($k \in [1, m]$) **do**
      $cut_{remov}^k \leftarrow$ the lightest $(k-1)$ links from $cuts_i^{GH}$ if exists, else **break**
      $LCs_i^k \leftarrow$ the k disjoint clusters resulted from the removal of cut_{remov}^k
      $assign\_sol \leftarrow$ **Cost-Aware-Best-Fit-Assignment**$(LCs_i^k, F, cut_{remov}^k)$
      if ($assign\_sol \neq null$) **then**
        $solution\_found \leftarrow true$
        // Calculate the allocation cost C(k) of this k-cut partitions
        $alloc\_cost_i^k \leftarrow$ **total-allocation-cost**$(LCs_i^k, assign\_sol, i)$ (5.6)
        if ($alloc\_cost_i^k > alloc\_cost_i^{best}$) **then**
          // Exit iterations : optimal solution is found for $(k-1)$ *partitions*
          break
        else
          $alloc\_cost_i^{best} \leftarrow alloc\_cost_i^k$
          // Memorize this allocation solution and evaluate the next k value
          $LCs_i^{best} \leftarrow LCs_i^k$
          $assign^{best} \leftarrow assign\_sol$
        end if
      end if
    end for
    $profit_i^{best} \leftarrow$ (**total-selling-revenue**$(i) - alloc\_cost_i^{best}$)
    if ($solution\_found$ & ($profit_i^{best} \geq R_0$)) **then**
      $accept\_matrix[i] \leftarrow accepted$
      // Effective Allocation: update resources and quotas
      $alloc\_plan[i, |V_i|] \leftarrow$ **allocate-request**$(LCs_i^{best}, assign^{best})$
      update-profit(cp_j)
    else
      $accept\_matrix[i] \leftarrow rejected$
      $alloc\_plan[i, l] \leftarrow -1$, for all $l \in V_i$
    end if
  end if
end for
return $accept\_matrix$, $alloc\_plan$

as optimal solution; otherwise it pursues with the next k value. Once the optimal k-cut is reached, the algorithm verifies if the resulted revenue is higher than a minimum threshold R_0 ($R_0 \geq 0$) to accept the request i and allocate resources according to the selected mapping solution; otherwise the request is rejected. The process updates the providers' profits and remaining capacities and continues with next requests until handling the entire batch.

$$alloc\_cost_i^k = \left(\sum_{\substack{V_c^{clus} \in LCs_i^k \\ assign\_sol[c]=j}} \sum_{l \in V_c^{clus}} (C_{j,l}^{local} \cdot d_i) \right) + \left(\sum_{\substack{V_c^{clus} \in LCs_i^k \\ assign\_sol[c]=f \neq j}} \sum_{l \in V_c^{clus}} (C_{f,l}^{out} \cdot d_i) \right) + \left(\sum_{\substack{V_c^{clus} \in LCs_i^k \\ assign\_sol[c]=f}} \sum_{\substack{V_{c'}^{clus} \in LCs_i^k \\ assign\_sol[c']=f' \neq f}} \sum_{l \in V_c^{clus}} \sum_{l' \in V_{c'}^{clus}} (tr_{l,l'}^i \cdot C_{f,f'}^{net}) \right) \tag{5.6}$$

5.3 Computational Complexity

This section analyzes the complexity of our profit maximization problem and assesses the ability of the proposed heuristic to handle large-scale instances in polynomial times. The algorithm performance evaluation in section 5.4 will bring additional experimental proof of its efficiency to find near-optimal solutions in reasonable convergence times.

Theorem 5.5. *The problem of profit maximization in Cloud federation is NP-Hard.*

Proof. Our optimization problem focuses on the allocation of complex requests requiring distributed and networked VMs across multiple federated infrastructure providers. The goal is to provide the optimal combination of insourcing, outsourcing and local allocations that maximize the providers' revenues while respecting resources and networking requirements. If considering only separate VMs allocation, the studied problem can be viewed as an instance of the Bin-Packing problem, where the items are the requested VMs and the bins are the hosting providers. Compared to the classical Bin-Packing problem, the specificities of our allocation problem are mainly the communication requirements between VMs (items) and the inter-cloud networking costs between federation providers (inter-bins costs). The problem bears other modifications including restricted availability periods of offered quotas, varied hosting costs between providers, different selling prices of VM instances, and rejection penalty for not-served requests.

Adding these networking and pricing constraints increases the complexity compared to the classical Bin-Packing that is known to be NP-Hard in its basic form [152]. This proves the NP-Hardness of our profit maximization problem seen as a generalization of the Bin-Packing problem. □

The *NCAFedRA* heuristic has been proposed to handle this NP-Hard problem in practical times. In the following, we assess its complexity through the analysis of the main steps of the algorithm:

1. **Gomory-Hu Tree Construction:** For each received request $G_i = (V_i, Tr^i)$, the heuristic constructs the corresponding Gomory-Hu tree, that requires $(|V_i| - 1)$ maximum flow computations. The fastest known max-flow algorithm is running in $\mathcal{O}(min\{|V_i|^{\frac{2}{3}}, |Tr^i|^{\frac{1}{2}}\} \times |Tr^i|)$ time [163], which is better than $\mathcal{O}(|V_i| \times |Tr^i|)$. An extra $\mathcal{O}(|V_i|)$ factor for handling all max-flow iterations gives a $\mathcal{O}(|V_i|^2 \times |Tr^i|)$ time complexity.

2. **Aggregate Cost Computation:** This step consists in computing the overall allocation cost for a given VMs-cluster. The hosting cost evaluation is done at worst in $\mathcal{O}(|V_i|)$ when the cluster lumps all VMs. For networking cost evaluation, the algorithm determines the cluster's neighbors achievable in $\mathcal{O}(m-1)$ in worst case when dealing with m partitions, and then calculates the necessary communication costs between neighbors executed also in $\mathcal{O}(m-1)$. This leads to a total time complexity not exceeding $\mathcal{O}(|V_i|+m+m)$, which is equivalent to $\mathcal{O}(max\{|V_i|, m\})$.

3. **Best-Fit Assignment:** This step assigns the list of current k-cut clusters to the federation providers, that is m clusters in worst case. For each cluster, the algorithm estimates its aggregate allocation costs among all providers to select the best one. The best-fit algorithm has an average complexity time of $\mathcal{O}(m \log m)$ and in worst case $\mathcal{O}(m^2)$. This leads to an overall time complexity not exceeding $\mathcal{O}(m \times [(m \times max\{|V_i|, m\}) + m^2])$, equivalent to $\mathcal{O}(max\{m^2 \times |V_i|, m^3\})$.

4. **optimal k-cut identification:** For each request i, we iterate the Best-Fit cluster assignment across a range of k values until converging to the optimal k-cut. In worst case, we have to deal with m partitioning steps. This leads to an overall time complexity of $\mathcal{O}(m \times [(max\{m^2 \times |V_i|, m^3\}) + (|V_i| + |V_i|^2)])$. The term $\mathcal{O}(|V_i| + |V_i|^2)$ corresponds to the solution cost evaluation in each iteration using equation (5.6). This is equivalent to $\mathcal{O}(max\{m^3 \times |V_i|, m^4\} + (m \times |V_i|^2))$.

5. **Requests batch processing:** Finally, an extra factor $\mathcal{O}(|R|)$ is added to handle all requests in the received batch.

In summary, the average computational complexity of the proposed heuristic algorithm is: $\mathcal{O}\left(|R| \times \left[(|V_i|^2 \times |Tr^i|) + (m^3 \times max\{|V_i|, m\}) + (m \times |V_i|^2)\right]\right)$. If we assume that the average requests size is higher than the federation size ($|V_i| \geq m$), the time complexity can be simplified to $\mathcal{O}\left(|R| \times \left[|V_i|^4 + |V_i|^4 + |V_i|^3\right]\right)$, that is $\mathcal{O}\left(|R| \times |V_i|^4\right)$.

5.4 Performance Evaluation

This section evaluates and compares the performance of the NCAFedRA algorithm with the exact model detailed in chapter 4, in terms of solution optimality and scalability. The assessment and comparison of the proposed algorithms was performed using a custom Java-based discrete event simulator and the CPLEX library [150] to solve the exact ILP model. Simulation results will show the efficiency of the heuristic algorithm that rapidly converges to near-optimal solutions, contrary to the ILP algorithm that does not scale well due to the limitations of the branch and bound method.

5.4.1 Simulation & Evaluation Settings

The performance evaluation was carried out using similar settings as in section 4.4.1 to compare the heuristic and exact approaches using the same conditions. The simulation parameters were drawn randomly in order to span the optimization space. Different federation scenarios were generated with various sizes ($[2; 30]$) and heterogenous providers' capacities and offerings. The request batches are generated according to a Poisson process with different rates to emulate the providers' day and night workloads. Each request is composed of a random number of connected VMs ([1; 50]), belonging to different VM instances (table 5.1) and organized in a graph with random topologies and traffic requirements. To emulate price variations between cloud providers, the end-user prices and the hosting and inter-cloud networking costs were randomly drawn from specific intervals as summarized in Table 5.2. The insourcing prices are dynamically adjusted by providers at each round using equation 4.1 based on their resources usage. For all simulated scenarios, 100 independent runs were conducted and averaged to produce each performance point in the reported curves.

5.4.2 Evaluation Results

Through extensive experiments, we first study the scalability of the heuristic approach by evaluating the algorithm convergence time with different problem sizes. We evaluate the heuristic solution quality in terms of profit improvements and requests acceptance

Instance type	CPU	RAM (Gbytes)
small	1	1.7
medium	1	3.75
large	2	7.5
xlarge	4	15
xxlarge	8	30

TABLE 5.1: VM's instances types

Instance	End-user Prices	Hosting Costs	Networking Costs
small	$[0.040\$; 0.060\$]$	$(0.5 * P_s^{user})$	$[0.001\$; 0.005\$]$
medium	$[0.062\$; 0.120\$]$	$(0.5 * P_m^{user})$	$[0.001\$; 0.005\$]$
large	$[0.140\$; 0.240\$]$	$(0.5 * P_l^{user})$	$[0.001\$; 0.005\$]$
xlarge	$[0.260\$; 0.480\$]$	$(0.5 * P_{xl}^{user})$	$[0.001\$; 0.005\$]$
xxlarge	$[0.520\$; 0.980\$]$	$(0.5 * P_{xxl}^{user})$	$[0.001\$; 0.005\$]$

TABLE 5.2: prices and costs

Curve	Performance	Algorithms	m	$\|R\|$	$\|V_i\|$	D^{conn}
Figure 5.6	Convergence Time	Exact, NCAFedRA	$2-30$	1	10	$0-1$
Figure 5.7	Convergence Time	Exact, NCAFedRA	$2-30$	1	20	$0-1$
Figure 5.8	Convergence Time	Exact, NCAFedRA	$2-30$	1	30	$0-1$
Figure 5.9	Convergence Time	Exact, NCAFedRA	$2-30$	$10, 30, 50$	6	0.5
Figure 5.10	Convergence Time	NCAFedRA	$30-60$	$5-50$	$10, 20$	0.5
Figure 5.11	Convergence Time	NCAFedRA	$30-60$	$1, 10$	$40, 50$	$0.5, 1$
Figure 5.12	Profit Improvements	Exact, NCAFedRA	$5, 8$	20	$2-30$	0.5
Figure 5.13	Profit Improvements	Exact, NCAFedRA	$2-15$	20	15	0.5
Figure 5.14	Acceptance Ratios	Exact, NCAFedRA	5	20	$2-30$	0.5

TABLE 5.3: Performances evaluation and Simulation settings

ratio. For convenience, Table 5.3 summarizes all the conducted simulations with the reported performance and evaluation settings for each experiment.

5.4.2.1 Scalability of the NCAFedRA Heuristic Algorithm

Impact of request topology: The evaluation of the ILP model has shown that the algorithm complexity is essentially governed by the size and connectivity of the request graphs. The first assessments aim at evaluating the scalability of the heuristic algorithm when dealing with these complex instances and its ability to reduce convergence times to acceptable levels. The experiments consist in comparing the exact and heuristic algorithms for several request graphs ($|V_i| = \{10, 20, 30\}$) ranging from unconnected to fully meshed VMs. The algorithm behaviour as a function of increasing federation size is reported in figures 5.6, 5.7 and 5.8 for request sizes of 10, 20 and 30 VMs respectively.

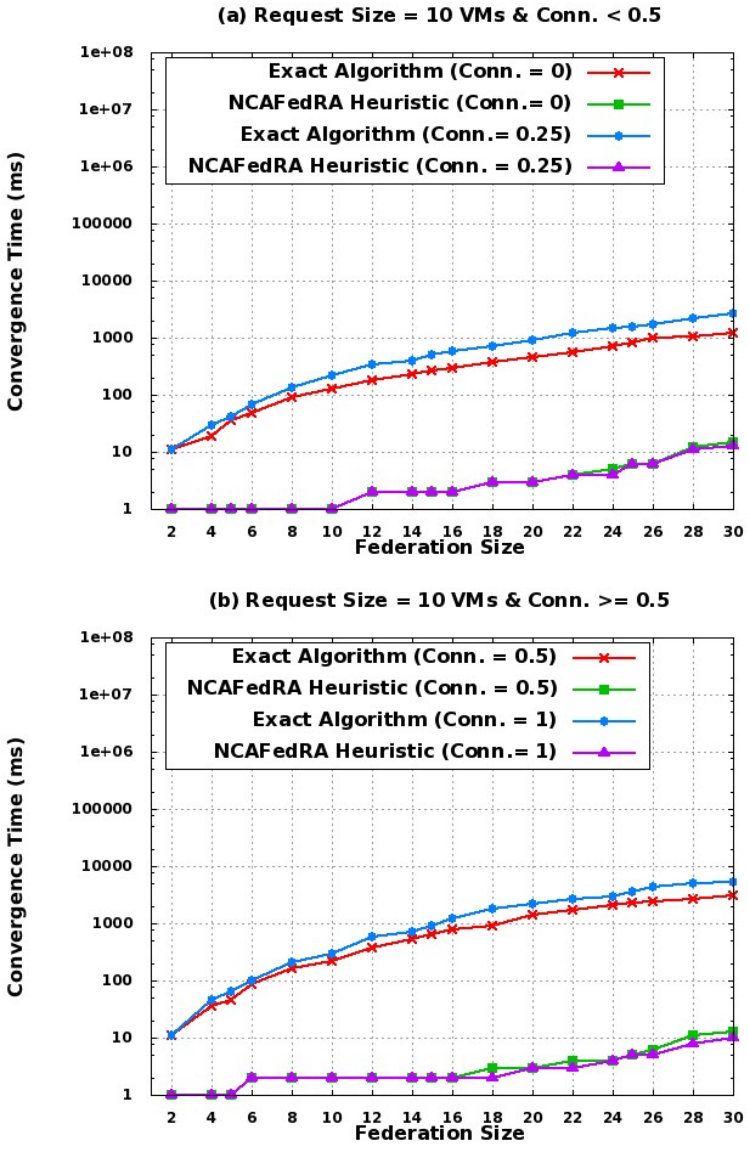


FIGURE 5.6: Convergence Time comparison between Exact and Heuristic Approaches for $|V_i| = 10$

For small and weakly connected request topologies ($|V_i| = 10$ and $D^{conn} < 0.5$), the exact algorithm is quite efficient and achieves optimal request allocation in milliseconds range for small and medium federation sizes ($m < 20$). For higher number of providers ($m \geq 20$) and highly connected request graph ($D^{conn} \geq 0.5$), the convergence times increase slightly to second and few seconds as depicted in Figure 5.6. The heuristic algorithm achieves better performance and reduces even more the convergence times to be in the order of 10 msec for all evaluated scenarios. The gap between both algorithms is in the $[10; 5 * 10^2]$ improvement factor in favor of the heuristic algorithm and it increases with increasing federation size and request graph connectivity D^{conn}.

With larger request sizes (20 and 30 VMs), the heuristic is shown to be much more robust

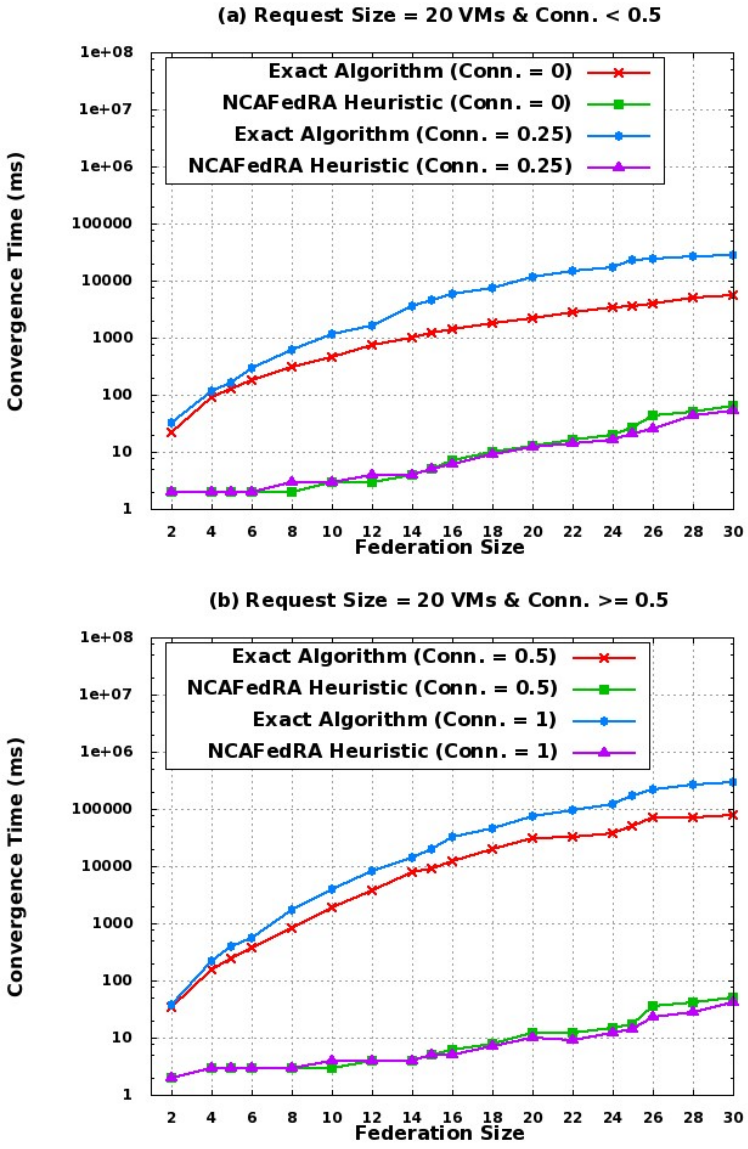


FIGURE 5.7: Convergence Time comparison between Exact and Heuristic Approaches for $|V_i| = 20$

and to achieve higher improvements in terms of convergence times. The results reported in Figure 5.7 show that the exact algorithm experiences an exponential increase in computational times to several seconds and minutes depending on the degree of request connectivity. In contrast, the heuristic algorithm scales much better and finds optimal solutions in less than 100 msec for all evaluated scenarios. A significant convergence time improvement ratio in the range of $[10^2; 10^4]$ can be observed in figure 5.7 depending on the federation size and request graph connectivity. For example, with a request graph of 20 fully meshed VMs and 30 participating providers, the heuristic converges to the allocation solutions within 41 msec, that is much faster with a factor of $(7 * 10^3)$ compared to the exact algorithm convergence time (over than 300 seconds). Note that this convergence time gain is significantly higher with complex topologies. Across a

30-sized federation, an improvement gain around $7 * 10^3$ is experienced with complete VM graphs versus only 10^2 ratio with unconnected VMs.

Furthermore, the simulation results show that the heuristic's performance is minimally affected by the request connectivity degree D^{conn} and yields similar convergence times. This behavior is reflected by the algorithm computational complexity, expressed by $\left(\mathcal{O}\left(|R| \times \left[(|V_i|^2 \times |Tr^i|) + (m^3 \times max\{|V_i|, m\}) + (m \times |V_i|^2)\right]\right)\right)$, that is mainly governed by the request and federation sizes ($|V_i|$ and m), but less by the batch size $|R|$ and the number of links $|Tr^i|$. The heuristic converges even a bit faster when dealing with highly connected topologies ($D^{conn} \geq 0.5$), as in the example scenario ($m = 30, |V_i| = 20$), where it converges to the solution in 40 msec with fully meshed VMs versus 65 msec with unconnected VMs. This minor increase of computational time is due to the use of the min k-cut approach. In fact, the k-cut algorithm has the weakness of possible unbalanced partitions, since it is based only on the link weights without any consideration of partitions sizes, which may generate large clusters that cannot be served by any provider. Thus, additional partitioning will be needed to get smaller clusters fitting available capacities, which will increase the convergence time. This is the case with unconnected VMs, which are related by fictitious zero-weighted cuts removed arbitrarily from the Gomory-Hu tree. Since the increase of partitions does not generate additional inter-cloud traffic and networking costs, the algorithm is forced to iterate until the maximum k value to evaluate all partitioning possibilities. In contrary with highly connected graphs, the algorithm may stop iterations before reaching the maximum k value and converge much faster to the best solution, since new partitions induce additional inter-Cloud networking costs (see Lines 21-23 in algorithm 4). Additional evaluations with higher request sizes ($|V_i| = \{30; 40; 50\}$), reported in Figure 5.11, confirmed this performance behavior as will be detailed later. Nevertheless, this slight increase in convergence times remains marginal and does not impact the heuristic efficiency in achieving good and adequate performance for cloud services provisioning.

Figure 5.8 extends the analysis of the algorithms scalability with larger requests composed of 30 VMs. Reported results confirm the exponential performance degradation of the exact algorithm with increasing federation size and request connectivity. For weakly connected graphs ($D^{conn} \leq 0.25$), the exact algorithm finds solutions in the order of tens of seconds to minutes depending on the federation size. With higher connectivity ($D^{conn} \geq 0.5$), the convergence times raise dramatically to tens of minutes (around 45 min for extreme cases). Beyond these input settings, the ILP-based algorithm reaches its limits and experiences unfeasible convergence times for operational cloud systems as several hours are required to find an allocation plan. In contrast, the heuristic algorithm is quite robust and exhibits far better performances in the order of tens and hundreds of milliseconds (125 msec with 30-sized federation). The relative performance gap between

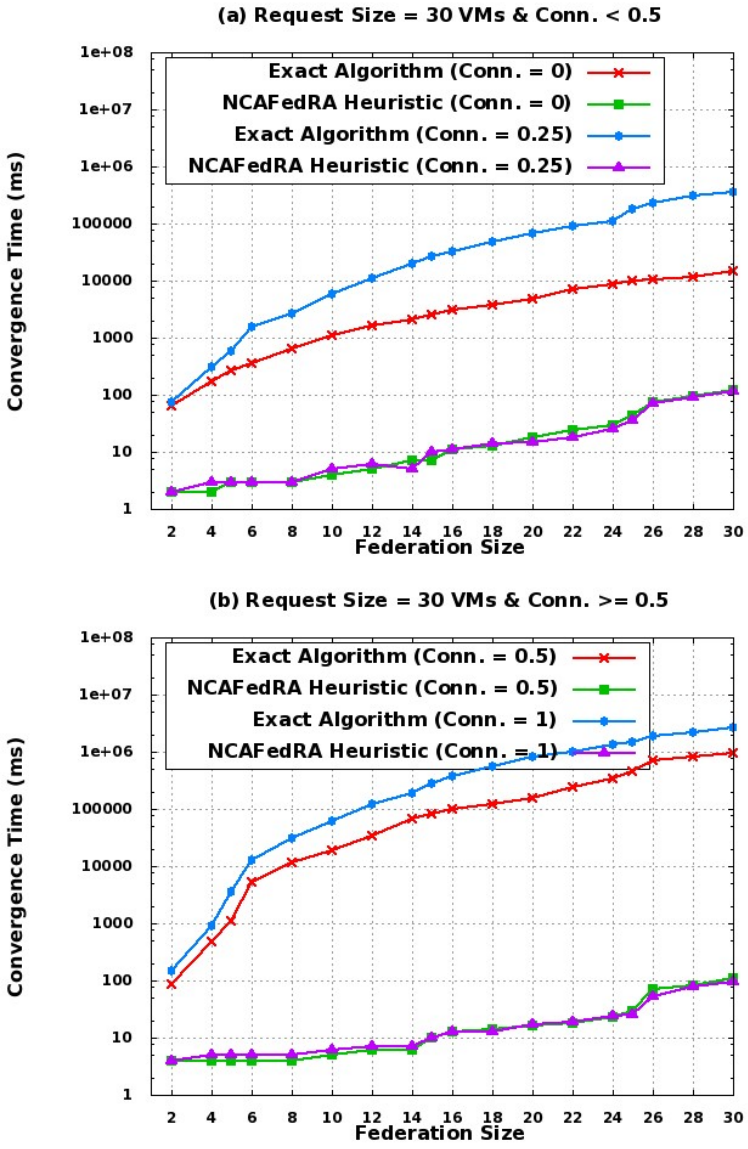


FIGURE 5.8: Convergence Time comparison between Exact and Heuristic Approaches for $|V_i| = 30$

both algorithms is more significant and ranges in the $[10^2; 10^5]$ interval in favor of the heuristic algorithm.

Impact of requests batch size: Figure 5.9 extends the scalability study of the heuristic algorithm by evaluating its convergence times when handling a batch of several requests. A simulation of different batch sizes ($|R| = \{10; 30; 50\}$) was conducted for federations ranging from 2 to 30 providers. The size of received requests was set to 6 VMs with a connectivity of 50%. Experimental results shown in Figure 5.9 confirm the efficiency of the heuristic algorithm that remains quite fast and achieves allocation decisions in the order of tens and hundreds of milliseconds for all simulated scenarios, unlike

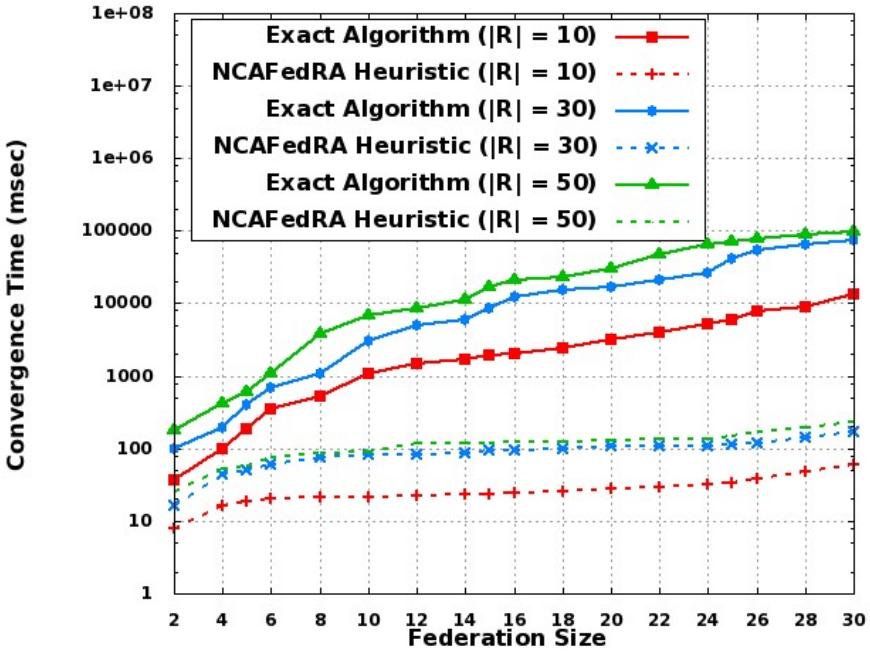


FIGURE 5.9: Impact of the batch size on the Convergence times of the Exact and Heuristic algorithms

the exact algorithm that rapidly reaches tens of seconds and minutes. The proposed heuristic is relevant for handling batched requests and reduces the convergence times by 2 to 3 orders of magnitude compared to the exact ILP model.

NCAFedRA Heuristic and large-scale problem instances: The above experiments have shed light on the limits of the ILP algorithm, that experiences exponential response times beyond ten providers and thirty partially meshed VMs. These problem sizes cover only a portion of likely encountered Cloud resource provisioning scenarios in distributed and federated Clouds. Does the heuristic algorithm scale with higher federation sizes and request graph complexity, and what are its limits? The answers to these questions are pointed out in Figures 5.10 and 5.11, that depict the heuristic performance for large federations (30 to 60 providers), large batch sizes and complex request graphs with 40 and 50 highly connected VMs.

For the first assessment of the heuristic scalability, we generated different batch sizes ($|R| = \{5; 10; 20; 30; 40; 50\}$) composed of request graphs with 10 or 20 VMs and an average connectivity of 50%, to evaluate the impact of increasing federation and batch sizes on the heuristic performance. Reported results in 5.10 confirm the robustness of the proposed approach that finds solutions in milliseconds and seconds and in less than 16 seconds for the extreme simulation case $(m, |R|, |V_i|) = (60, 50, 20)$.

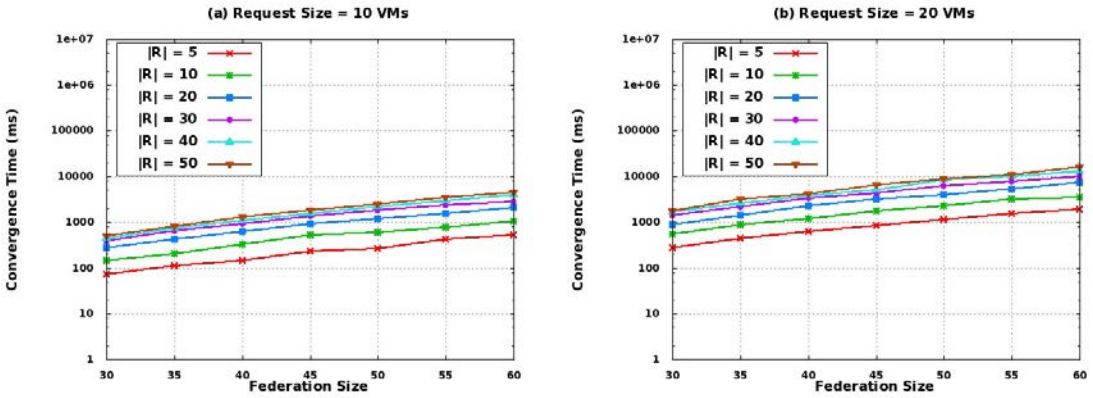


FIGURE 5.10: Heuristic algorithm's convergence times for large federations.

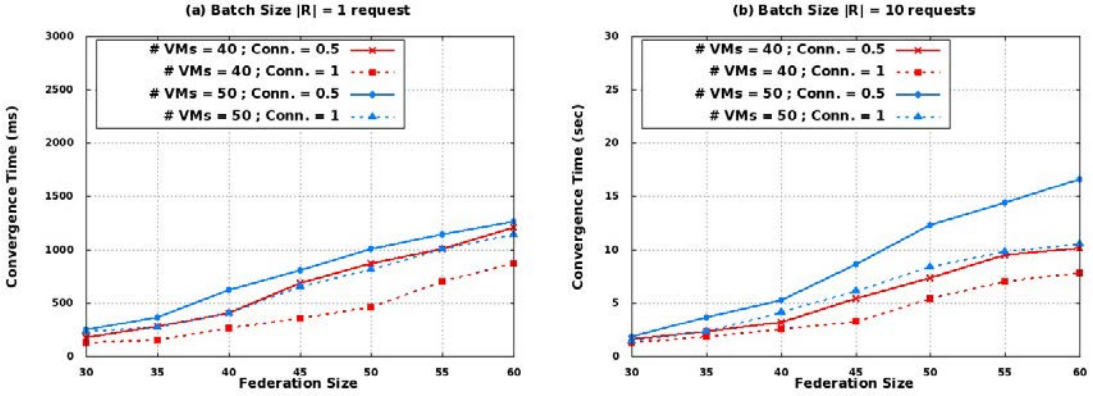


FIGURE 5.11: Heuristic algorithm's convergence times for large and complex requests

Figure 5.11 confirms the expected stable behavior of the heuristic with larger request graphs composed of 40 and 50 VMs, when handled separately or lumped into a batch. Both full and partial mesh graph topologies were used for the evaluation, and the experienced convergence times were reported as a function of federation size. For both Batch sizes ($|R| = \{1, 10\}$), the heuristic is quite fast and finds allocation solutions in the order of second and a few of seconds respectively (a maximum of 1.3 sec and 16 sec are recorded in worst cases for $(|R|, |V_i|, m, D^{conn}) = (1, 50, 60, 0.5)$ and $(|R|, |V_i|, m, D^{conn}) = (10, 50, 60, 0.5)$). Moreover, the reported results illustrate that the heuristic achieves better performance with complete VMs graphs compared to partially connected graphs. A gap of tens to hundreds of milliseconds between the 50% and 100% connectivity levels is experienced with single request allocation. This performance gap increases when dealing with request batches to a few seconds in favor of complete graphs. The heuristic finds the solution in 7 and 10 seconds for the $(m, |V_i|, D^{conn}) = (60, 40, 1)$ and $(m, |V_i|, D^{conn}) = (60, 50, 1)$ scenarios respectively, versus 10 and 16 seconds in case of 50% connected topologies. This gain in convergence time is due to the stringent networking requirements in complete VMs graphs, that impose to the heuristic a limited

number of cuts to satisfy the cost minimization constraint. In the contrary, with partially connected graphs there is less strict networking requirements which leads to more splitting iterations before converging to the best allocation plan.

Based on all these evaluation results, the NCAFedRA heuristic keeps its promises in terms of scalability and stands out as a viable and efficient solution for resource allocation problems in large-scale federated and distributed Clouds. What remains to be verified is the optimality of the heuristic solutions and its quality in meeting the optimization objectives and constraints.

5.4.2.2 Effectiveness of the NCAFedRA Heuristic

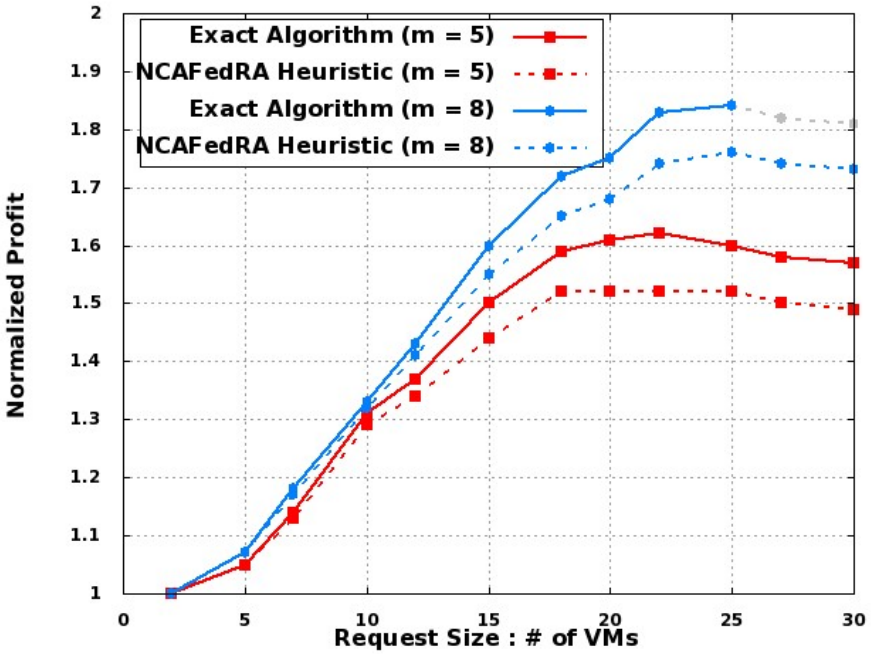


FIGURE 5.12: Exact and Heuristic achieved profit improvements

Profit Improvement: To assess the effectiveness of the proposed algorithm, we compare in Figure 5.12 the profit improvements achieved based on the heuristic decisions to the optimal profit generated by the exact ILP. The assessment scenario corresponds to a federation size of 5 and 8 providers, homogeneous in available resources (1500 CPU and 6000 Gbytes of RAM) and received loads. Providers receive batches of 20 requests, composed of $[2; 30]$ partially meshed VMs with an average connectivity of 50%, arrived at a rate of 2 batches/hour during the day and 1 batch/hour at night. The reported results correspond to the realized profits during 48 hours as a function of request size for an arbitrary reference provider cp_j. The gaps in % in profit improvement are summarized

$\|V_i\|$	2	5	7	10	12	15	18	20	22	25	27	30
$m = 5$	0.00	0.25	0.71	2.42	3.75	6.18	7.94	8.31	10.03	8.48	7.62	8.26
$m = 8$	0.00	0.26	0.37	1.28	2.18	4.73	7.22	7.39	9.24	7.95	–	–

TABLE 5.4: Gaps (%) between Exact and NCAFedRA achieved profit improvements

for convenience in Table 5.4. This gap is defined as the difference between cooperative profits achieved by both federation algorithms when normalized to the selfish profit (non-federation strategy) according to formula in (5.7). Using normalized profits allows to show the comfortable benefits achieved by the heuristic algorithm despite the decrease in revenues compared to the optimal. Reported gaps represent an average over 100 independent runs for small and medium request sizes, for which the exact algorithm is able to find optimal solutions.

$$\begin{aligned} Gap(\%) &= (Cooperative\ Exact\ revenues - Cooperative\ Heuristic\ revenues) * 100 \\ &= \left(\frac{Exact\ revenues - Selfish\ revenues}{Selfish\ revenues} * 100\right) - \\ &\left(\frac{Heuristic\ revenues - Selfish\ revenues}{Selfish\ revenues} * 100\right) \end{aligned} \tag{5.7}$$

For both algorithms there is an optimal workload leading to a profit improvement peak depending on the federation size and available capacity per provider (see section 4.4.3.2 on favorable federation conditions). With 5 federated providers, the higher profit improvement occurs with 22 VMs per request, while with 8 providers the peak takes place with higher request sizes (25 VMs) since more workload can be served across the federation. Beyond these optimal values, the profits decrease with increasing workload as the federation providers will be overloaded and forced to reject requests, and hence lose revenues due to the rejection penalty $\mathcal{L}^{penalty}_{rejection}$. Moreover, the heuristic is shown to find near optimal solutions with only 10% gap compared to optimal profits as worst performance degradation. For small request sizes (up to 8 VMs), the heuristic performs quite close to optimal with less than 1% degradation in achieved profits. This gap remains lower than 5% for request graphs with less than 15 VMs. For increasing loads in the range of 15 to 30 VMs per request, the profit gap increases up to 10% with the optimal workload value, before stabilizing around the 8% with higher request sizes as depicted in Figure 5.12. Beyond 25 VMs per request in a federation of 8 providers, the exact algorithm reached its limits and was not able to find solutions during several hours.

In fact, the higher performance gaps with large requests of 15 to 30 VMs is due to the limitation of the number of authorized k-cut compared to the request sizes. Indeed, the moderate connectivity of requests enables wide distribution of resources while satisfying

m	2	4	5	6	8	10	12	14	15
$Profit\ Gaps(\%)$	0.6	5.47	6.18	5.76	4.73	2.68	2.28	1.31	0.70

TABLE 5.5: Impact of the federation size on the profit improvements gaps

networking requirements; whereas the maximum k value is not high enough ($k \leq m$) to favor better request partitioning across providers, especially with possible resulting unbalanced k-cut partitions. In fact, the quality of the heuristic's solutions in terms of performance gaps is highly dependent on the problem's inputs including the federation and requests sizes. This explains the performance improvement when the federation size has increased from 5 to 8 providers as shown in Table 5.4. Nevertheless, despite these performance gaps, the profits achieved by the heuristic remain significant with regard to the fast convergence times (milliseconds to several seconds) compared to the time required by the exact model to find solutions (several minutes to hours).

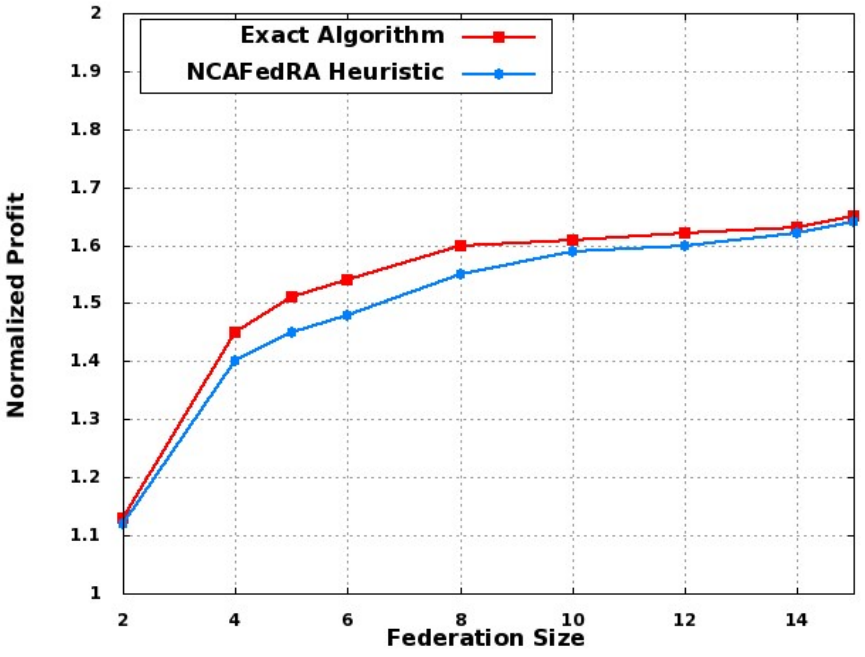


FIGURE 5.13: Impact of the federation size on the profit improvement gaps between Exact and Heuristic algorithms

Impact of federation size on the profit improvement gaps: Figure 5.13 extends the analysis of the heuristic's optimality by evaluating the impact of varying the number of cooperating providers on the solutions quality. For this assessment we hold constant the size of request graphs at 15 VMs with 50% connected topologies and we evaluate the algorithm's behaviour for federations of 2 to 15 providers. The normalized profits depicted in Figure 5.13 confirms the sensibility of the heuristic to the federation size with consistently better performance with increasing size. As shown in Table 5.5, the profit

gaps decrease significantly from around 5% with less than 8 providers to 2% for larger federation of 10 and 12 providers. The heuristic achieves the best profit improvements with 14 and 15 providers with only 1.31% and 0.70% degradation compared to the exact solutions. Indeed, as the upper bound (m) of authorized cuts increases, the heuristic is more likely to converge to optimal partitioning and allocation solutions. The results emphasize the efficiency of our algorithm with large-scale problem instances for which it achieves near optimal performance with significantly reduced computation times.

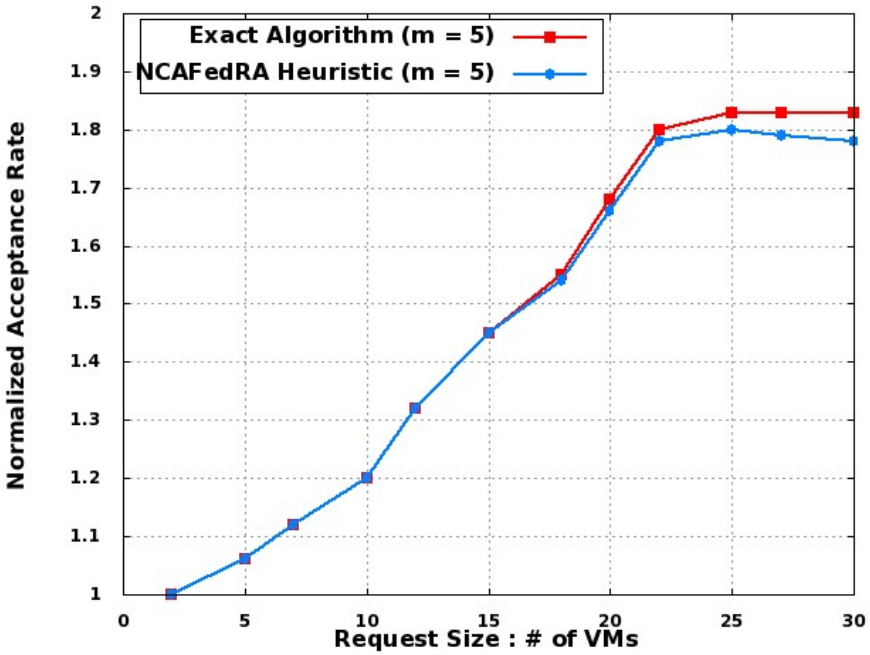


FIGURE 5.14: Exact Versus Heuristic request acceptance rates

Acceptance Ratio: To pursue the performance evaluation of the heuristic, we measure in Figure 5.14 its requests acceptance rate. For this experiment, a federation of 5 providers receiving batches of 20 partially meshed request graphs with different sizes of 2 to 30 VMs is used. Reported results in Figure 5.14 confirm the efficiency of the heuristic in reducing request rejection rate with performance quite close to the optimal. For small requests of up to 15 VMs, the heuristic achieves identical request acceptance improvement as the exact algorithm. For higher load, both algorithms achieve better acceptance rates with a small performance advantage in favor of the exact method. A maximum gap of 5% is experienced for extreme cases in the simulation (30 VMs). This gap is due to the restricted number of authorized cuts leading to large VM-clusters not fitting the remaining capacities, as detailed above. The obtained results confirm and match those of profit improvements reported in Figures 5.12 and 5.13.

Based on all evaluation results, the NCAFedRA heuristic performance is quite good in terms of profit, request acceptance rate and remarkable in terms of convergence time and scalability with large problem sizes.

5.5 Conclusions

This chapter presents a topology-aware heuristic algorithm for profit-driven resource allocation in cloud federations. The proposed solution relies on a Gomory-Hu based min k-cut algorithm and a Best-Fit assignment strategy, which combined together achieve both optimization objectives in terms of revenue maximization and user requirements satisfaction. The proposed heuristic is shown to perform close to the exact ILP formulation in terms of profit improvements and request acceptance rates, with less than 10% of performance gaps in all simulated scenarios. Moreover, the heuristic algorithm scales well with problem size and exhibits fast convergence times not exceeding tens of second, as opposed to the exact algorithm that experiences exponential convergence times. The heuristic is remarkably efficient with large federations and highly connected topologies for which it improves convergence time by 3 to 5 orders of magnitude, while achieving near-optimal solutions with less than 2% of profit gaps. Finally, both exact and heuristic algorithms are exploitable for resource provisioning in distributed and federated Clouds. The exact model is a viable and efficient solution for small and medium problem instances. Beyond the limits of the branch and bound method, the heuristic stands out as a powerful alternative for large problem instances, able to achieve practical response times and performances quite close to optimal. The exact method remains nevertheless requisite as it can serve as a benchmark to assess the quality of approximate and heuristic solutions.

Chapter 6

Conclusions and Perspectives

Contents

This final chapter concludes the work presented in this dissertation and points out future research directions. We first summarize the thesis contributions and highlight the main results regarding providers' profit maximization in Cloud federations. Then, we outline some promising perspectives for future investigations, to address the research limitations and further refine the proposed resource allocation algorithms.

6.1 Results and Discussion

With the rapid development of Internet and hardware/software virtualization technologies, Cloud Computing has rapidly become the de facto model for delivering on-demand cost-effective and large-scale IT solutions over the past few years. This promising paradigm has fundamentally revolutionized the way IT industries conduct their businesses by enabling new multi-tenant third-party hosted scenarios. Despite this success, the growing scale of Cloud infrastructures and the increasing workloads still raise several resource management challenges for Cloud stakeholders. While the functional and economic benefits of moving to the Cloud have been extensively discussed in the literature, much less attention has been devoted to the opportunities and issues faced by Cloud vendors to improve their profitability. Recently, Cloud Federation has emerged as a key solution to help providers build scalable infrastructures through cooperation and resource sharing with others to achieve better performance and revenues.

Defining efficient allocation strategies for workload orchestration in a federation is a challenging task for providers since they also have to deal with cooperation decisions. This involves many factors and criteria, including the diversity of resource and pricing offerings, heterogeneity of demand requirements, applications' topologies and induced networking costs. To take advantage of this multi-Cloud environment and make fruitful collaborations, it is crucial for providers to use advanced optimization mechanisms to automate this challenging and tedious provisioning task. This thesis addresses profit optimization for cloud infrastructure providers engaging in a federation. The overall goal is to provide effective algorithms to find optimal distributed resource allocation plans that achieve the best tradeoffs between user satisfaction, resource utilization and cost minimization.

In line with these objectives, the major contributions of the thesis are listed below:

- An in-depth review of the literature on infrastructure resource provisioning in distributed federated Clouds has been provided. The study includes an overview of the key concepts, enabling technologies and pricing models of Cloud computing. The major motivations and revenue-related challenges for inter-Cloud scenarios have also been studied. Finally, a detailed discussion of prior works on profit-driven allocation models in federated Clouds has been presented. This analysis allowed us to build a deeper understanding of the problem and identify the relevant challenges and constraints to consider and to define the thesis scope and objectives. (Chapters 2 and 3).

- Based on the literature analysis, a generic model for Cloud federation resource management has been designed. The model introduces the federation scenario properties, the graph-based request modeling to better capture users' requirements and support both simple and complex services as well as the resource cost and pricing schemes. (Chapter 4).

- An integer linear program for request allocation across federation has been proposed to achieve cost-effective cooperation and placement decisions. The allocation choices are treated jointly in a global objective function, that combines actions and partitions the request across different providers, while considering the VMs connectivity and the induced networking costs. The evaluation results highlighted the efficiency of the algorithm in improving profit and acceptance ratio, with respectively up to 47% and 69% improvements compared with the non-federated scenario. The exact algorithm is shown to outperform the baseline federation approaches with up to 10% of profit improvements, and to experience practical convergence times with typical federation and request sizes. The achievable profits depend on several parameters namely the received workloads during rounds

and the provider and federation sizes. The reported results can be used to derive guidelines on the favorable conditions for a provider to join or build a federation. (Chapter 4).

- To address scalability issues, a new topology-aware heuristic algorithm has been introduced for the revenue maximization problem. The heuristic uses a Gomory-Hu based clustering algorithm to partition requests into smaller subgraphs, and a cost oriented best-fit matching for resource placement across providers. Evaluation results have proven the efficiency of the heuristic, that closely approximates the optimal profit outcome while improving convergence times by several orders of magnitude. The heuristic is shown to scale well with problem size and achieve better performances with complex scenarios (large federation and request sizes, significant graph connectivity) with less than 2% of revenue loss compared to the optimal. (Chapter 5).

6.2 Future Research Directions

Beyond the thesis contributions, we have identified a number of additional investigations that can be pursued in future work to address the issues outside the scope of this study and further enhance the mechanisms of profit maximization in cloud federations. The potential future research directions include:

- **The design and development of advanced pricing strategies** represents a natural extension of the current work. In this thesis, to update the insourcing prices, we have used a simple pricing mechanism from the literature that dynamically adjusts prices according to remaining capacities. The evaluation results have highlighted the importance of considering both current resource utilization and future demands when setting prices and sharing quotas. However, existing pricing models are still relatively abstract and do not provide such advanced policies. As future work, we plan to enhance our allocation algorithms with a load predictor to derive more elaborate pricing schemes to improve profits even more. To this end, we foresee exploring different prediction techniques such as Markov chains and regression approaches that we believe are suitable to characterize and predict Cloud workload fluctuations.

- **Intra- and Inter-Cloud network provisioning**: So far we have only considered the allocation of computational resources. While this is reasonable for traditional VM-based Cloud systems, we believe it is important to extend the work to support further resource types such as network and storage to meet the new trends in

cloud services. In addition, the data-center and inter-Cloud network topologies are important aspects to consider with distributed resource allocation. For the current research, we have assumed that federated providers are interconnected by high performance and capacity links meeting applications' requirements. This differs from real Cloud environments where dynamic network conditions can influence application performance. Network features in terms of bandwidth and latency variations should be integrated to the proposed model to address more thoroughly the resource allocation problem in Cloud federations.

- **Advanced Resource Allocation Policies**: In addition to resource capacities and prices, the allocation model could be enhanced to consider additional constraints and criteria including energy consumption, geographic location and providers' reputation. Moreover, the fault tolerance and SLA enforcement are important aspects to be considered to prevent the application performance degradation. Advanced policies are needed to detect and react to SLA violations through partial or complete update of the current allocation plans (e.g. scaling up/down VM sizes, VM migrations, VM replications ...). Furthermore, the resource placement and consolidation at each provider can be handled jointly with the federation level optimization to provide more generic solutions. Further allocation actions such as dynamic adjustment of local hosting capacity (restarting and shutting down servers in response to the workload) and admission control decisions can be added to the model.

- **Cloud Federation Framework**: An important goal of this thesis is to integrate our profit optimization algorithms into a real federation testbed to confirm their performance and compatibility with cloud infrastructures. We aim to provide a global orchestration framework for inter-Cloud management that automates the resource allocation decisions, assesses insourcing prices and shared quotas, supports QoS monitoring and establishes connectivity between distributed resources. The achievement of such advanced features requires the adoption of efficient protocols and APIs. For interoperability purposes, we can investigate the open Cloud standards, such as the OCCI interface [112] for providers' interaction, the OpenFlow protocol [164] for resource connectivity, the OpenStack [81] and OpenNebula [80] Cloud managers for resource deployment and management, and common monitoring services like Monitis [165] or Amazon CloudWatch [166].

Thesis Publications

International Conferences

- S. Rebai, M. Hadji and D. Zeghlache, "Improving profit through cloud federation," In *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2015, pp. 732-739.
- S. Rebai and D. Zeghlache, "Gomory-Hu based Algorithm for Distributed Resource Allocation in Cloud Federation," (Under Review).

Open source software

- S. Rebai and D. Zeghlache, "CompatibleOne Placement Service," http://gitorious.ow2.org/ow2-compatibleone/dev-cops.

Technical Reports

- I.J. Marshall et al., "CompatibleOne Resource Description System (CORDS)," Technical report, CompatibleOne, http://www.compatibleone.com/community/wp-content/uploads/2014/05/CordsReferenceManualV2.15.pdf, 2013.
- S. Rebai et al., "CompatibleOne Placement Service COPS," Technical report, CompatibleOne, 2012.

Appendix A

French Summary - Résumé Français

A.1 Introduction

L'informatique en nuage (Cloud Computing) est un modèle à grande échelle et en évolution continue, permettant le provisionnement et l'utilisation des ressources informatiques à la demande, selon un modèle rentable de facturation à l'usage "pay-as-you-go". Ce nouveau paradigme a rapidement révolutionné l'industrie IT et a permis de nouvelles tendances en matière de prestation de services informatiques, y compris l'externalisation des infrastructures IT vers des prestataires tiers spécialisés. Cependant, la nature multi-utilisateur des plateformes d'hébergement, ainsi que la complexité des demandes, soulèvent plusieurs défis liés à la gestion des ressources Cloud. Malgré l'attention croissante portée à ce sujet, la plupart des efforts ont été axés sur des solutions centrées sur l'utilisateur, et malheureusement beaucoup moins sur les difficultés rencontrées par les fournisseurs Cloud pour maximiser leurs bénéfices et améliorer leurs affaires dans un tel marché concurrentiel. Les solutions d'allocation traditionnelles basées sur des capacités d'hébergement statiques et limitées, ne sont pas adaptées aux nouvelles tendances Cloud, et empêchent les fournisseurs de réaliser les performances et les revenus souhaités.

Dans ce contexte, la Fédération de Cloud a été récemment proposée comme une solution clé pour répondre à l'augmentation et la fluctuation des charges de travail. Les fournisseurs ayant des besoins complémentaires en ressources au fil du temps, peuvent collaborer et partager leurs infrastructures respectives via l'externalisation ("Outsourcing") et l'internalisation ("Insourcing") des machines virtuelles. Une telle coopération permet aux fournisseurs de faire face à la limitation des ressources et de mieux satisfaire les demandes et exigences des utilisateurs, en leur offrant la possibilité de dépasser

leurs capacités d'hébergement initiales. Toutefois, être membre d'une fédération, rend la procédure d'allocation des requêtes plus complexe à traiter, puisque les fournisseurs doivent également gérer leurs décisions de collaboration et de partage. Ce problème n'a pas été suffisamment abordé par la communauté scientifique. Les travaux de recherche antérieurs ont été principalement focalisés sur la définition de plates-formes et d'architectures pour l'interopérabilité et les interactions entre fournisseurs. Cependant, peu d'attention a été accordée à la problématique de gestion et de distribution des charges de travail au sein d'une fédération. Ceci est d'une importance cruciale pour les fournisseurs de Cloud du point de vue rentabilité, et particulièrement délicat dans une fédération impliquant plusieurs membres et différentes ressources et applications distribuées.

Cette thèse aborde le problème d'optimisation du profit via la fédération et l'allocation optimale des ressources parmi plusieurs fournisseurs d'infrastructures. L'étude examine les principaux défis et opportunités liés à la maximisation des revenus dans une fédération de Clouds, et définit des stratégies efficaces pour diriger les fournisseurs dans leurs décisions d'allocation et de coopération. Le but est de fournir de nouveaux algorithmes qui automatisent la sélection du plan d'allocation le plus rentable, qui satisfait à la fois la demande des utilisateurs et les exigences de mise en réseau dans ce contexte distribué. Pour atteindre ces objectifs, des approches exacte et heuristique sont proposées et évaluées en termes de performance, flexibilité et scalabilité, afin d'identifier les meilleurs conditions et équilibres pour l'amélioration des bénéfices. Nous visons des modèles d'allocation génériques et robustes qui répondent aux nouvelles tendances Cloud, en termes de gamme et de qualité des services fournis. Les travaux de recherche actuels se concentrent principalement sur l'allocation des machines virtuelles indépendantes et séparées. Cependant, les utilisateurs de Cloud s'attendent à des services beaucoup plus avancés avec différentes ressources distribuées et connectées. Notre objectif est d'étendre l'applicabilité des modèles proposés à ces demandes complexes tout en conservant de bonnes performances.

Conformément aux objectifs de la thèse, nous avons mené une étude approfondie des travaux antérieurs traitant la problématique de provisionnement des ressources d'infrastructure dans les environnements Cloud distribués. L'analyse a porté notamment sur les modèles d'allocation ayant pour objectif la maximisation des profits dans les fédérations de Clouds, et les lacunes et défis associés.

Dans un deuxième temps, nous avons proposé un programme linéaire en nombre entiers (ILP), pour aider les fournisseurs de services IaaS à ajuster leurs décisions d'hébergement et de coopération en réponse à leurs charges de travail et aux offres de la fédération. Grâce à une modélisation des demandes utilisateurs par graphes génériques, l'approche proposée s'applique efficacement aux requêtes complexes, exigeant le provisionnement

d'infrastructures virtuelles composites et connectées. Afin de sélectionner les meilleures solutions, nous traitons les différentes décisions d'allocation potentielles conjointement dans une même formule d'optimisation globale. Cette formulation peut résulter en un plan de placement optimal qui combine différentes actions d'externalisation, d'internalisation et d'allocation locale, et partitionne une requête entre plusieurs fournisseurs, tout en satisfaisant les exigences de communication entre les services élémentaires. En plus de la structure (topologie) des graphes de requêtes, ce partitionnement prend en compte les prix et quotas de ressources proposés par les autres membres de la fédération ainsi que les coûts d'hébergement et de mise en réseaux des ressources demandées.

Enfin, pour respecter les attentes de délais de provisionnement des services Cloud, nous avons proposé une heuristique pour faire face à la dégradation des performances du modèle exact avec les instances de grandes tailles. Pour réduire la complexité du processus de partitionnement, l'approche proposée recourt à des méthodes de coupe minimale (min k-cut) pour la décomposition des graphes de requêtes initiaux, et à des stratégies de meilleur ajustement (Best-Fit) pour l'allocation et le placement des sous-graphes résultants. L'utilisation conjointe de ces deux techniques permet de capturer l'essence du problème d'optimisation et de respecter les différents objectifs fixés, tout en améliorant le temps de convergence vers les solutions optimales et proches de l'optimale de plusieurs ordres de grandeur.

A.2 Algorithme Exact d'Allocation et de Fédération

Dans cette section, nous décrivons les modèles conceptuel et analytique proposés pour la résolution du problème d'allocation et de fédération des ressources. Nous présentons en premier lieu la modélisation des paramètres de conception de l'algorithme exact de maximisation de profit, y compris le scénario de fédération, les requêtes utilisateurs, les exigences de coopération et les coûts d'allocation à prendre en compte. Ensuite, nous présentons la formulation mathématique du problème, basée sur un programme linéaire en nombres entiers, et ayant pour objectif d'aider les fournisseurs à optimiser leurs décisions d'allocation et de coopération selon les offres de la fédération.

A.2.1 Modélisation du problème

A.2.1.1 Modélisation de l'environnement de Fédération

La figure A.1 décrit notre modèle de fédération impliquant m fournisseurs d'infrastructure Cloud, $F = \{cp_1, cp_2, ..., cp_j, ..., cp_m\}$, en coopération selon un mode d'interaction peer-to-peer [79]. Ces fournisseurs sont supposés être connectés par des liens réseaux de haute performance capables de satisfaire continuellement les exigences de communication et d'interaction des applications distribuées. Chaque fournisseur dispose d'une quantité limitée de ressources à répartir entre l'utilisation interne de son centre de données, et la contribution à la fédération en tant que quotas de ressources partagés. A chaque cycle d'allocation, le fournisseur exécute notre algorithme pour déterminer la répartition optimale de ses ressources locales et répondre aux demandes des utilisateurs et celles des autres membres. L'algorithme lui permet également de fixer la partie des requêtes à externaliser à la fédération pour sous-traitance, en fonction des prix proposés. Pour accentuer la contrainte de limitation des ressources, CPU_j et MEM_j définissent les capacités maximales de ressources de calcul (CPU) et de mémoire disponibles dans le Cloud domestique ("Home Cloud") cp_j. Nous utilisons également les notations CPU_f^{Avail} et MEM_f^{Avail} pour représenter les quotas de CPU et de mémoire partagés par chaque fournisseur fédéré cp_f en guise de coopération.

A.2.1.2 Modélisation des requêtes de ressources

Durant chaque cycle d'allocation, un fournisseur cp_j peut recevoir plusieurs lots de requêtes R. Chaque lot est composé de plusieurs demandes de la part des utilisateurs finaux et/ou des fournisseurs membres. Chaque requête reçue i est modélisée par un graphe non orienté $G_i = (V_i, Tr^i)$, où les sommets V_i représentent les ressources demandées en termes d'instances de VMs, et les arêtes $Tr^i = (tr^i_{l,l'})_{1 \leq l,l' \leq |V_i|}$ reflètent les exigences de communication et d'échange de trafic entre les VMs l et l'. Nous supposons aussi que toutes les machines virtuelles l ($l \in V_i$) associées à une requête donnée i, restent actives durant toute sa période d'activité $d_i = d_{i,l}, \forall l \in V_i$.

A.2.1.3 Modèle de tarification des ressources

Concernant les bénéfices réalisés par le fournisseur, ce dernier touche un certain revenu unitaire $P_{i,l}$ pour toute acceptation d'une instance de VM l associée à une requête i. Afin de favoriser la coopération au sein de la fédération, les fournisseurs devraient facturer l'hébergement des requêtes d'internalisation des autres membres moins cher que

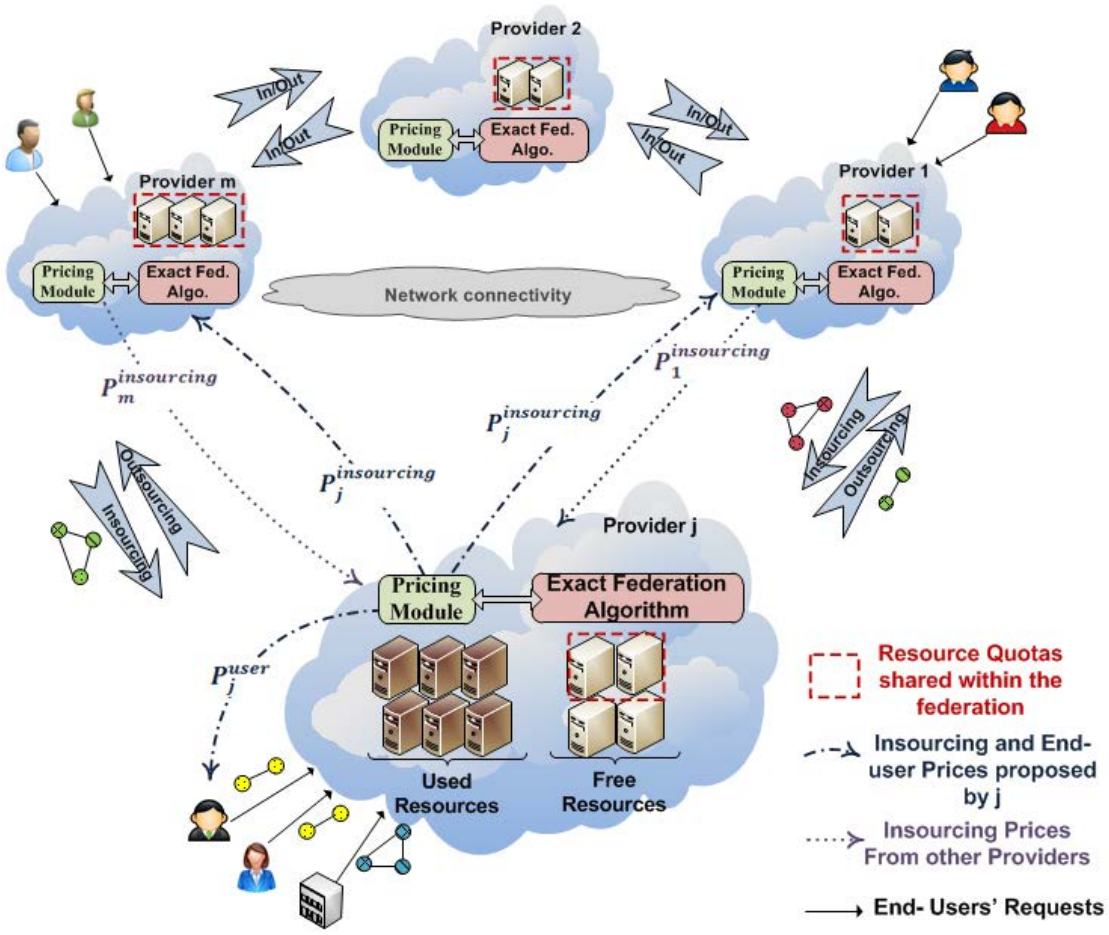


FIGURE A.1: Le Scénario de fédération de Cloud

le prix payé par les utilisateurs finaux. Pour établir ces prix avantageux, nous avons opté pour un simple et efficace mécanisme de tarification [105], qui permet d'ajuster dynamiquement les prix d'internalisation en fonction des conditions actuelles du système, selon l'expression mathématique ci-dessous:

$$P_{type}^{insourcing} = \frac{Cap_{type}^{max} - Cap_{type}^{idle}}{Cap_{type}^{max}} * (P_{type}^{user} - Cost_{type}) + Cost_{type} \quad (A.1)$$

L'expression prend en compte le coût d'hébergement des instances de VMs ($Cost_{type}$), le prix d'allocation fixe facturé aux utilisateurs (P_{type}^{user}), les capacités d'hébergement maximale (Cap_{type}^{max}) et inactive (Cap_{type}^{idle}) chez le fournisseur, et détermine son prix d'internalisation selon le taux d'utilisation de ses ressources. L'équation (A.1) assure l'équilibrage de charge entre les membres fédérés, en diminuant le prix des fournisseurs ayant des capacités restantes importantes pour encourager l'internalisation des requêtes.

Cependant, un prestataire de services supporte plusieurs coûts qu'il faut prendre en considération pour évaluer ses revenus potentiels. Outre le coût d'utilisation des ressources

locales ($C^{local}_{j,l}$), le fournisseur encaisse un coût d'externalisation $C^{out}_{f,l}$ pour chaque VM l déléguée à un autre fournisseur cp_f. Le coût de mise en réseau $C^{net}_{f,f'}$ pour toutes VMs l et l' distribuées entre différents membres cp_f et $cp_{f'}$ de la fédération doit être également pris en compte. Finalement, vu que le rejet des requêtes affecte directement la réputation et les bénéfices des fournisseurs, nous avons introduit une pénalité $\mathcal{L}^{penalty}_{rejection}$ pour refléter la perte moyenne de revenu pour chaque demande rejetée.

A.2.2 Formulation en programme linéaire en nombres entiers

Il convient de noter que l'algorithme proposé sera exécuté par chaque membre de la fédération. Les fournisseurs se serviront du modèle de tarification (A.1) pour établir leurs prix d'internalisation à pratiquer durant chaque cycle d'allocation. Comme le montre la Figure A.2, l'algorithme aide chaque fournisseur cp_j $(j = 1, ..., m)$ à partager les requêtes reçues en sous-ensembles à héberger localement (chez cp_j) ou à externaliser à d'autres fournisseurs cp_f $(f \neq j)$, ainsi que de sélectionner les demandes d'internalisation à accepter. Cette décision prend en compte les prix et quotas proposés par les membres de la fédération et les coûts des ressources et leurs mise en réseau. Le but est de trouver la distribution optimale des demandes à travers la fédération, en maximisant les revenus et minimisant les coûts de chacun des fournisseurs afin d'améliorer leurs bénéfices.

Pour la résolution du problème, nous dérivons un programme linéaire en nombres entiers (ILP) centré sur un membre arbitraire cp_j de la fédération, puisque l'algorithme est exécuté indépendamment par chacun des fournisseurs. Le couplage est assuré par l'équation de tarification (A.1). La fonction objective de notre programme linéaire doit:

1. maximiser le profit réalisé par le fournisseur cp_j à travers l'allocation optimale des requêtes reçues (typiques et/ou d'internalisation) sur sa propre infrastructure locale;
2. améliorer ses revenus en externalisant des requêtes vers d'autres membres proposant des prix d'allocation avantageux en comparaison avec ses coûts d'hébergement en local et les prix facturés à ses utilisateurs;
3. minimiser les coûts de mise en réseau nécessaires pour la connectivité et l'interaction des machines virtuelles réparties sur plusieurs infrastructures dans la fédération;
4. maintenir une bonne réputation pour les fournisseurs en minimisant le nombre des requêtes rejetées.

Pour atteindre ces objectifs, nous définissons un certain nombre de variables booléennes et entières, énumérées dans le tableau A.1. La variable de décision bivalente $x_{j,f,l}$ indique

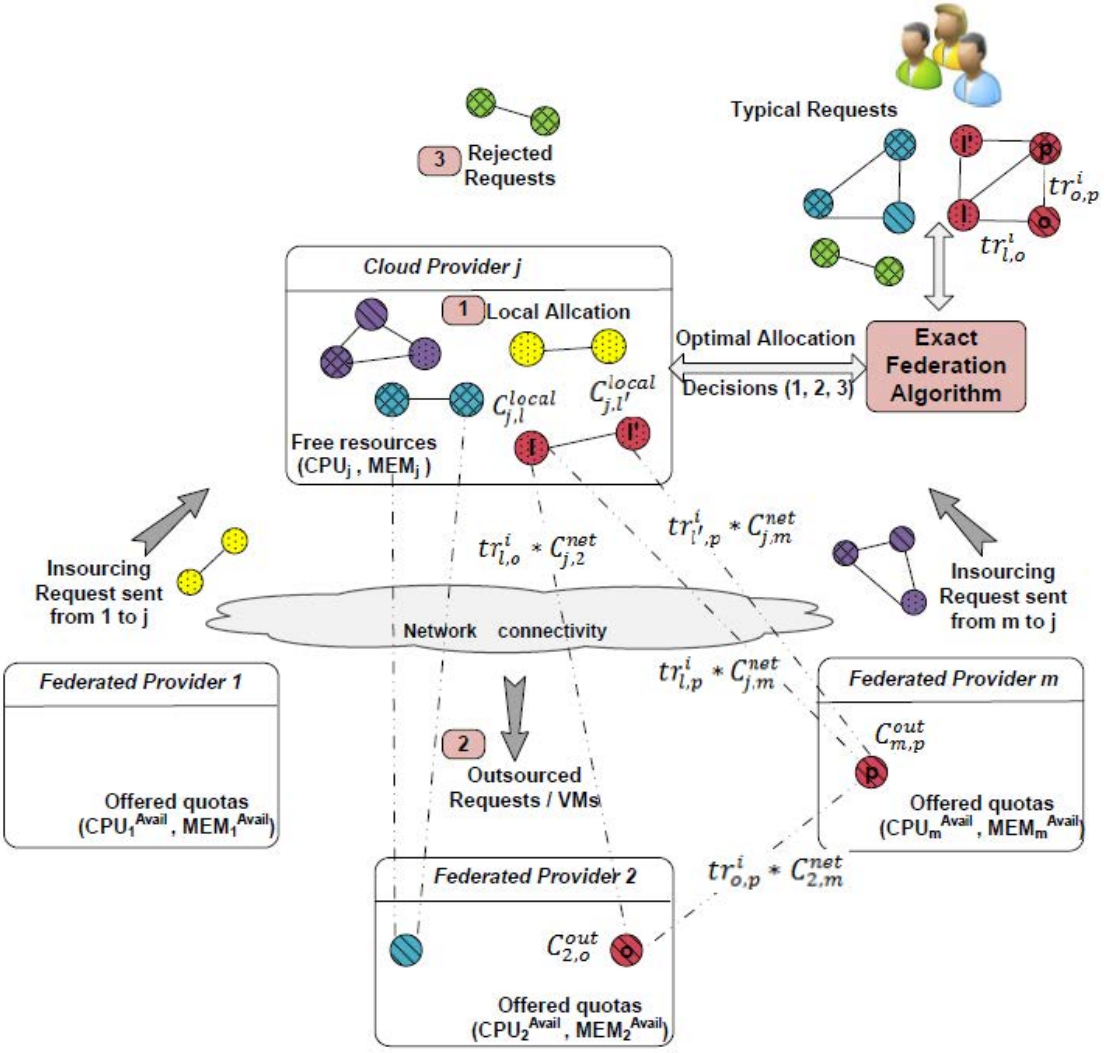


FIGURE A.2: Decision Making Process

si une VM l reçue de la part d'un fournisseur cp_f est acceptée par le fournisseur cp_j et servie localement sur son propre infrastructure. Il convient de noter que la variable $x_{j,j,l}$ (quand $f = j$) représente les requêtes d'utilisateurs finaux reçues et servies en local par cp_j. Afin de différencier les requêtes utilisateurs de celles des autres fournisseurs ($f \neq j$), nous introduisons l'ensemble S_i. Cet ensemble est égal à $\{j\}$ dans le cas d'une requête utilisateur, et égal à $\{f/f = 1, ..., m; f \neq j\}$ dans le cas d'une requête d'internalisation de la part de cp_f. L'ensemble S_i est utilisé également pour contrôler le prix facturé $P_{i,l}$, qui est fixe pour les utilisateurs finaux mais dynamiquement ajusté pour les requêtes de "insourcing". Pour les décisions d'externalisation, nous utilisons la variable $x_{f,j,l}$ pour indiquer si une VM l est confiée par le fournisseur cp_j à un autre membre cp_f de la fédération. La variable a_i est utilisée pour indiquer si la demande d'allocation de ressources i est acceptée ou rejetée.

En utilisant ces notations, la fonction objectif globale est formulée par l'équation (A.2),

Notation	**Signification**
m	Nombre de fournisseurs dans la fédération.
j, f, f'	Sont utilisés pour désigner les fournisseurs de la fédération. cp_j fait référence au Cloud domestique ("Home Cloud"), cp_f et $cp_{f'}$ aux autres membres fédérés.
i	Une requête de ressources reçue de la part d'un utilisateur final ou un autre fournisseur fédéré.
R	Un lot de requêtes contenant plusieurs demandes reçues (de la part des utilisateurs finaux et des fournisseurs) à traiter simultanément, $R = \cup_i$.
V_i	L'ensemble des VMs demandées par une requête i. l et l' font référence à deux VMs quelconques dans cet ensemble ($l, l' \in V_i$).
cpu_l, mem_l	Les exigences en ressources requises par la VM l en termes de puissance de calcul (CPU) et de mémoire (RAM) respectivement.
d_i	La durée de vie (de service) de la requête i.
$tr^i_{l,l'}$	Le trafic à échanger entre les VMs l et l' de la requête i.
CPU_j	La capacité maximale de CPU sur l'infrastructure locale de cp_j.
MEM_j	La capacité maximale de mémoire (RAM) sur l'infrastructure locale de cp_j.
CPU_f^{Avail}	Le quota de CPU partagé par le fournisseur cp_f dans la fédération.
MEM_f^{Avail}	Le quota de RAM partagé par le fournisseur cp_f dans la fédération.
d_f^{Avail}	La durée de disponibilité des quotas offerts par le fournisseur cp_f.
$C^{net}_{f,f'}$	Le coût unitaire de mise en réseau entre les fournisseurs cp_f et $cp_{f'}$.
$C^{local}_{j,l}$	Le coût d'hébergement en local d'une VM l.
$C^{out}_{f,l}$	Le coût d'externalisation (outsourcing) d'une VM l chez le fournisseur cp_f.
$P_{i,l}$	Le prix unitaire facturé par le fournisseur lors de la satisfaction d'une VM l demandée par la requête i.
$\mathcal{L}^{penalty}_{rejection}$	La perte moyenne de revenu (pénalité) pour chaque requête rejetée par le fournisseur.
S_i	Est l'ensemble des acteurs ayant soumis la requête au fournisseur cp_j. Selon notre modélisation, $S_i = \{j\}$ si i est une requête typique d'un utilisateur final, et $S_i = \{f = 1, ...m; f \neq j\}$ si i est une requête d'internalisation (insourcing) de la part du fournisseur cp_f.
Variables	**Définition**
$x_{j,f,l}$	Est une variable binaire. $x_{j,f,l} = 1$ si la VM l a été reçue par le fournisseur cp_j de la part de cp_f et allouée en local, et 0 sinon.
$x_{f,j,l}$	Est une variable binaire. $x_{f,j,l} = 1$ si la VM l a été externalisée (outsourced) par le fournisseur cp_j à un autre membre cp_f, et 0 sinon.
$y^j_{f,f',l,l'}$	Est une variable binaire. $y^j_{f,f',l,l'} = x_{f,j,l} \cdot x_{f',j,l'}$.
a_i	Est une variable binaire. $a_i = 1$ si la requête i a été acceptée, et 0 sinon.

TABLE A.1: Notations et Variables

et est soumise à un ensemble de contraintes linéaires et d'intégrité exprimées respectivement par les équations (A.3) à (A.13) et l'équation (A.14). Les deux premières contraintes (A.3) et (A.4) formulent la condition sur la limitation des ressources locales pour que les allocations ne dépassent pas les capacités maximales de cp_j. Les inégalités (A.5) et (A.6) assurent que les allocations d'externalisation demeurent en deçà des quotas mis à disposition par les autres fournisseurs. L'inégalité (A.7) garantit que les quotas sont disponibles pendant toute la durée de vie des requêtes externalisées. L'inégalité (A.8) et l'égalité (A.9) assurent que les requêtes acceptées sont bien satisfaites. La contrainte (A.8) garantit que chaque VM est allouée à un seul et unique hôte (fournisseur); alors que la contrainte (A.9) garantit qu'une requête est acceptée seulement si elle est entièrement allouée. La contrainte (A.10) empêche les boucles sur les opérations d'externalisation et d'internalisation. les inégalités (A.11) et (A.12) définissent les relations entre les variables de décisions $x_{j,f,l}$, $x_{f,j,l}$ et $y^j_{f,f',l,l'}$. Finalement, pour garantir au fournisseur cp_j un certain gain en participant à la fédération, la contrainte (A.13) assure que les solutions sélectionnées mènent toujours à un revenu supérieur à un seuil minimal R_0.

$$\begin{aligned}\max Z_j = \Bigg\{ \Bigg[\Bigg(\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} \sum_{f \in S_i} (P_{i,l} - C^{local}_{j,l}) \cdot x_{j,f,l} \cdot d_i \Bigg) + \\ \Bigg(\sum_{i=1}^{|R|} \sum_{f=1, f \neq j}^{m} \sum_{l=1}^{|V_i|} (P_{i,l} - C^{out}_{f,l}) \cdot x_{f,j,l} \cdot d_i \Bigg) - \\ \Bigg(\sum_{i=1}^{|R|} \sum_{f=1}^{m} \sum_{f'=1}^{m} \sum_{l=1}^{|V_i|} \sum_{l'=1, l'>l}^{|V_i|} tr^i_{l,l'} \cdot C^{net}_{f,f'} \cdot y^j_{f,f',l,l'} \Bigg) \Bigg] - \\ \Bigg((|R| - \sum_{i=1}^{|R|} a_i) \cdot \mathcal{L}^{penalty}_{rejection} \Bigg) \Bigg\} \cdot \Delta t\end{aligned} \tag{A.2}$$

Sous Contraintes:

$$\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} \sum_{f \in S_i} cpu_l \cdot x_{j,f,l} \leq CPU_j \tag{A.3}$$

$$\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} \sum_{f \in S_i} mem_l \cdot x_{j,f,l} \leq MEM_j \tag{A.4}$$

$$\sum_{i=1}^{|R|} \sum_{l=1}^{|V_i|} cpu_l \cdot x_{f,j,l} \leq CPU^{Avail}_f \quad \forall f = 1, ..., m;\ f \neq j \tag{A.5}$$

$$\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|} mem_l \cdot x_{f,j,l} \leq MEM_f^{Avail} \quad \forall f = 1, ..., m;\ f \neq j \tag{A.6}$$

$$\begin{gathered} d_i \cdot x_{f,j,l} \leq d_f^{Avail} \\ \forall i = 1 \in R;\ \forall l = \in V_i;\ \forall f = 1, ..., m; f \neq j \end{gathered} \tag{A.7}$$

$$\sum_{f=1}^{m} x_{f,j,l} \leq 1 \quad \forall i \in R;\ \forall l \in V_i \tag{A.8}$$

$$\sum_{f=1}^{m}\sum_{l=1}^{|V_i|} x_{f,j,l} = |V_i| \cdot a_i \quad \forall i \in R \tag{A.9}$$

$$x_{f,j,l} = 0 \quad \forall i, \{i \in R \mid S_i \neq \{j\}\};\ \forall l \in V_i;\ \forall f \in S_i \tag{A.10}$$

$$\begin{gathered} x_{f,j,l} + x_{f',j,l'} - y^j_{f,f',l,l'} \leq 1 \\ \forall i \in R;\ \forall l = 1, ..., |V_i|;\ \forall l' = 1, ..., |V_i|; l' > l; \\ \forall f = 1, ..., m;\ \forall f' = 1, ..., m \end{gathered} \tag{A.11}$$

$$\begin{gathered} \sum_{f'=1}^{m} y^j_{f,f',l,l'} \leq x_{f,j,l} \\ \forall i \in R;\ \forall l = 1, ..., |V_i|;\ \forall l' = 1, ..., |V_i|; l' > l; \\ \forall f = 1, ..., m; \end{gathered} \tag{A.12}$$

$$\begin{gathered} \left[\left(\sum_{i=1}^{|R|}\sum_{l=1}^{|V_i|}\sum_{f\in S_i}(P_{i,l} - C^{local}_{j,l}) \cdot x_{j,f,l} \cdot d_i\right) + \right. \\ \left(\sum_{i=1}^{|R|}\sum_{f=1,f\neq j}^{m}\sum_{l=1}^{|V_i|}(P_{i,l} - C^{out}_{f,l}) \cdot x_{f,j,l} \cdot d_i\right) - \\ \left.\left(\sum_{i=1}^{|R|}\sum_{f=1}^{m}\sum_{f'=1}^{m}\sum_{l=1}^{|V_i|}\sum_{l'=1,l'>l}^{|V_i|} tr^i_{l,l'} \cdot C^{net}_{f,f'} \cdot y^j_{f,f',l,l'}\right)\right] \geq R_0 \end{gathered} \tag{A.13}$$

$$\begin{gathered} x_{j,f,l}\ ,\ x_{f,j,l}\ ,\ y^j_{f,f',l,l'} \in \{0, 1\} \\ \forall i \in R;\ \forall l = 1, ..., |V_i|;\ \forall l' = 1, ..., |V_i|; l' > l; \\ \forall f = 1, ..., m;\ \forall f' = 1, ..., m \end{gathered} \tag{A.14}$$

Il convient de noter que les prix $P_{i,l}$ dans le premier et le deuxième terme de la fonction objectif, sont établis par le biais du modèle de tarification exprimé par l'équation (A.1). Ce prix est mis à jour par le fournisseur cp_j au début de chaque cycle d'allocation pour fixer le prix facturé aux autres membres pour l'hébergement de leurs requêtes d'internalisation. $P_{i,l}$ reste fixe pour les utilisateurs finaux. Dans le second terme, exprimant le revenu réalisé par les opérations d'externalisation, $C^{out}_{f,l}$ est le coût de sous-traitance appliqué par les autres fournisseurs à cp_j. Ce coût n'est autre que le prix d'insourcing proposé par les autres membres ($f \neq j$), et qui est également fixé par ces derniers selon l'équation (A.1) en fonction de l'utilisation de leurs ressources respectives.

L'évaluation des performances du modèle exact et les différents gains identifiés, ont confirmé la pertinence de la fédération des ressources et du modèle proposé, pour l'amélioration des bénéfices des fournisseurs et de la satisfaction des utilisateurs. L'étude a mis en exergue également les conditions les plus favorables pour la participation et/ou la construction d'une fédération.

A.3 Approche Heuristique basée sur les arbres de Gomory-Hu

Le modèle exact réalise de bonnes performances avec les instances de problème de taille moyenne et de complexité modérée. Toutefois, comme tout problème NP-difficile, les solutions exactes ne passeront pas à l'échelle et leurs performances se dégradent avec les instances de grande tailles. Notre algorithme exact entraîne des temps de convergence inacceptables, notamment avec l'augmentation du nombre de VMs composant les requêtes et leurs degrés de connectivité. Vu que le temps de réponse est une préoccupation cruciale pour les utilisateurs de Cloud, les décisions d'allocation de ressources doivent être prises dans un temps opportun. Ceci nous oblige à concevoir des algorithmes heuristiques efficaces, permettant de trouver des solutions optimales et proches de l'optimale dans un temps polynomial.

L'heuristique proposée est basée sur la clusterisation des graphes de requêtes, et se déroule selon ces trois étapes principales décrites dans l'algorithme 7:

1. La décomposition des requêtes en un ensemble de grappes de VMs (clusters) disjoints, ayant de faibles flux de trafic de communication inter-grappes, selon une approche de *minimum k-cut*.
2. L'affectation de ces partitions candidates aux membres de la fédération suivant un algorithme de *meilleur ajustement (Best-Fit)* basé sur les coûts d'allocation.

3. L'identification de la meilleure k-coupe (k-cut) qui résulte en une partition de requête optimale ayant le coût minimum d'hébergement et de mise en réseau.

Dans les sections suivantes, nous détaillons les différentes étapes de l'algorithme proposé.

A.3.1 Décomposition des graphes de requêtes

La décomposition des requêtes est une variante du problème de partitionnement de graphes et de k-coupe minimale ("minimum k-cut"). Une k-cut est un ensemble d'arêtes $S \in E_G$, dont la suppression partage un graphe non-orienté $G = (V_G, E_G)$ en k composants connectés. Le problème de "minimum k-cut" consiste à trouver l'ensemble S ayant le poids global le plus faible. Ce problème est résoluble en temps polynomial ($\mathcal{O}(|V|^{k^2})$) pour des valeurs de k fixées [151], mais il est NP-complet avec des k faisant partie des variables d'entrée [152]. Vu que le nombre optimal k de partitions des requêtes n'est pas connu à l'avance, nous utilisons un algorithme heuristique populaire et efficace basé sur les arbres de Gomory-hu [154].

Definition A.1. Un arbre de Gomory-Hu $T^{GH} = (V_T, E_T)$ associé à un graphe non-orienté G, est une représentation compacte de la connectivité entre tous ses sommets. Il s'agit d'un arbre pondéré ayant les mêmes noeuds que G ($V_T = V_G$), mais ses arêtes $|E_T| = (|V_G| - 1)$ représentent les coupes minimales entre toutes les paires de sommets dans le graphe d'origine.

Cet arbre peut être construit en temps polynomial, en se basant sur $(|V_G|-1)$ évaluations des flux maximaux entre les sommets du graphe [154]. La Figure A.3-(b) montre un exemple d'un arbre Gomory-Hu T_i^{GH} associé au graphe G_i représenté par A.3-(a). Par exemple, le poids 6 de l'arête reliant les sommets (3) et (5) de l'arbre T_i^{GH}, correspond à la 3-5 coupe minimale dans le graphe d'origine G_i. La coupe minimale entre n'importe quelle paire de sommets dans G_i est égale au poids minimum sur les arêtes composant le chemin entre les deux noeuds dans T_i^{GH}. Par exemple, la 2-9 coupe minimale est égale à $(min\ \{10, 6, 9\} = 6)$. Pour en savoir davantage au sujet des arbres de Gomory-Hu, les lecteurs peuvent se référer au [154] et [155].

Afin de sélectionner le plan d'allocation le plus rentable, l'heuristique débute par l'application de l'algorithme de Gomory-Hu au graphe de requête reçu. Cela permet d'obtenir une représentation concise T_i^{GH} des flux maximaux échangés entre toutes les VMs, de sorte que n'importe quelles $k \in [1; m]$ partitions peuvent être facilement obtenues en cas de besoin. Une k-coupe de la requête G_i en k sous-graphes, est obtenue simplement en enlevant les $(k-1)$ arêtes de T_i^{GH} ayant le poids le plus faible. Cette élimination d'arêtes

basée sur un tri croissant de poids assure que les VMs fortement connectées feront partie de la même partition et seront allouées sur la même infrastructure, alors que les grappes de VMs résultantes peuvent être réparties dans la fédération. La Figure A.3 illustre un exemple de partition d'une requête complexe G_i, en 3 sous-graphes de VMs, en éliminant les deux arcs de plus faible poids dans T_i^{GH}.

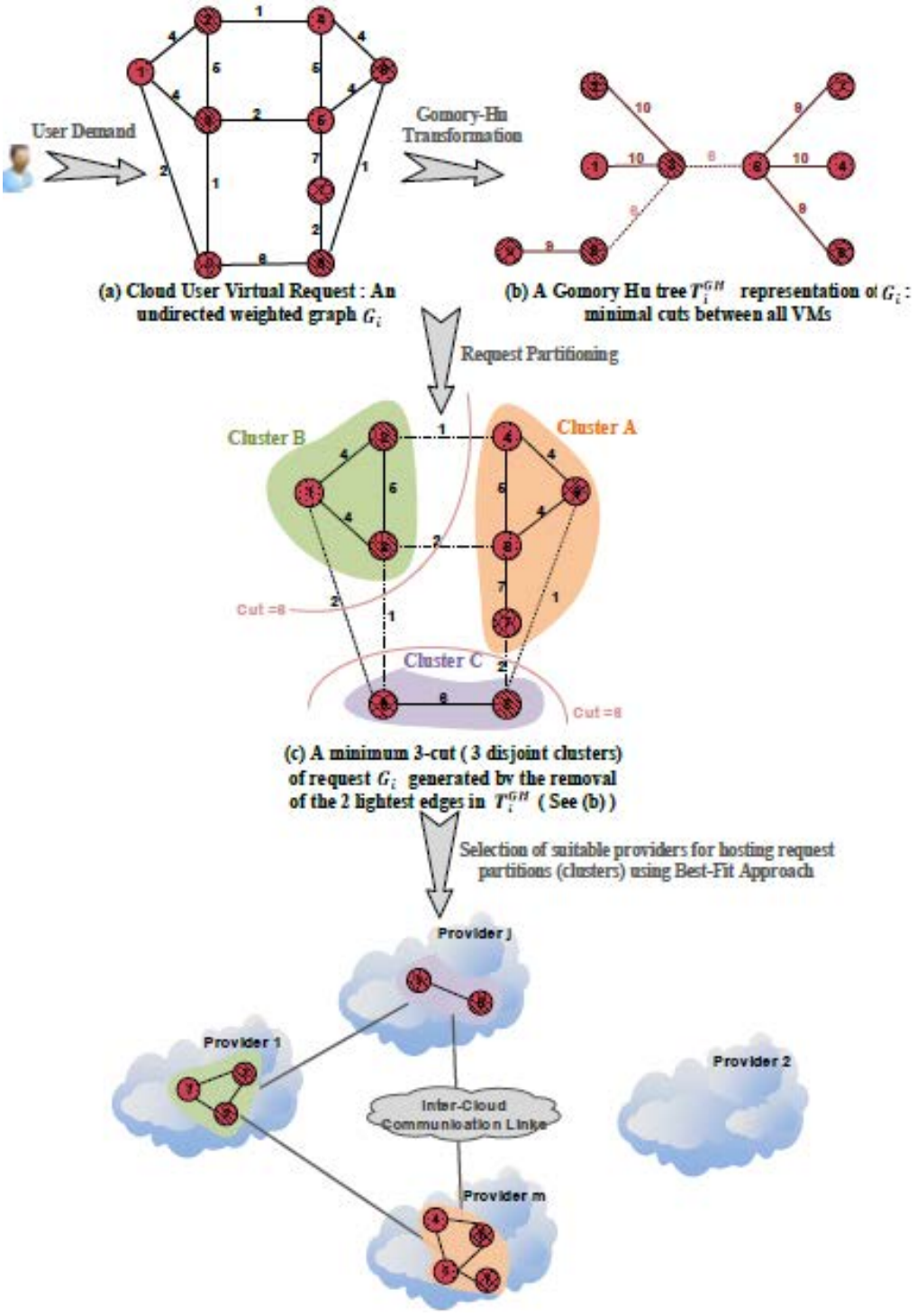


FIGURE A.3: Partition et Allocation des requêtes au sein de la Fédération

Après la décomposition du graphe, l'heuristique doit sélectionner pour chaque partition candidate, le fournisseur le plus approprié pour l'allocation des ressources selon les coûts d'hébergement et de mise en réseau. Cette étape d'affectation est basée sur une approche de meilleur ajustement (Best-Fit), détaillée dans les sections suivantes.

A.3.2 Calcul de la métrique de coût générique

Le traitement des requêtes complexes nécessite une attention particulière concernant les exigences et coûts de mise en réseau entre les composants distribués. Même avec le modèle de partitionnement minimisant le trafic sur les liens du réseau inter-Cloud, il serait plus bénéfique de considérer également les coûts de communication inter-fournisseurs pour la prise de décision. La gestion de ces coûts devient plus compliquée avec l'augmentation du nombre des fournisseurs et celui des partitions à allouer. Pour résoudre ce problème, nous définissons une métrique de coût global, estimant les frais totaux d'allocation d'une grappe de VMs donnée, en matière de ressources de calcul et de communication. L'algorithme 5 récapitule les étapes d'évaluation de cette métrique de coût:

L'estimation du coût global $C^{Aggregate}_{f,V_c^{clus}}$ d'une grappe de VMs V_c^{clus} en cas d'allocation chez un fournisseur cp_f, est exprimée par l'équation (A.15). Ce coût correspond à la somme du coût d'hébergement des différentes VMs y appartenant $C^{Hosting}_{f,V_c^{clus}}$ (A.16), et du coût approximatif $C^{Networking}_{f,V_c^{clus}}$ de mise en réseau avec ses voisins distribués dans la fédération. Le terme $cost(l)$ dans l'expression (A.16) fait référence au coût d'allocation de la VM l, qui correspond soit au coût d'allocation en local $C^{local}_{j,l}$, soit au coût d'externalisation $C^{out}_{f,l}$. Le coût $C^{Networking}_{f,V_c^{clus}}$ est fixé à 0 s'il y a un seul cluster à allouer; sinon il est évalué par l'équation (A.17), comme détaillé dans l'algorithme 5.

$$C^{Aggregate}_{f,V_c^{clus}} = C^{Hosting}_{f,V_c^{clus}} + C^{Networking}_{f,V_c^{clus}} \tag{A.15}$$

$$C^{Hosting}_{f,V_c^{clus}} = \sum_{l \in V_c^{clus}} cost(l) \cdot d_i \tag{A.16}$$

$$C^{Networking}_{f,V_c^{clus}} = \sum_{V_n^{clus} \in N_c^{assign}} bw^{cut}_{c,n} \cdot C^{net}_{f,Ass(n)} + AC^{net}_f \cdot BW^{Ngh}_c \tag{A.17}$$

$$AC^{net}_f = \frac{\sum_{f'=1;f' \neq f}^{m} C^{net}_{f,f'}}{(m-1)} \tag{A.18}$$

Algorithm 5 Approximation of the aggregate cost metric

function: **Aggregate-Cost-Cluster**$(V_c^{clus}, cp_f, LCs_i^k, cut_{remov}^k)$;

Input: A cluster V_c^{clus}, a provider cp_f, the list of all k VM-clusters $LCs_i^k = \{V_1^{clus}, V_2^{clus}, \ldots, V_k^{clus}\}$, the set of removed $(k-1)$ cuts $cut_{remov}^k = \{e\_remov_{(c,c')}^{cut}; c, c' \leq k\}$ partitioning the original graph into k clusters

Output: The approximate overall cost $C_{f,V_c^{clus}}^{Aggregate}$ for serving the cluster V_c^{clus} on provider cp_f

Initialize: $C_{f,V_c^{clus}}^{Aggregate} \leftarrow 0$; $C_{f,V_c^{clus}}^{Hosting} \leftarrow 0$; $C_{f,V_c^{clus}}^{Networking} \leftarrow 0$
// Evaluate the hosting cost
calculate $C_{f,V_c^{clus}}^{Hosting}$ using equation (A.16)
// Evaluate the networking cost
if ($size(LCs_i^k) = 1$) **then**
  $C_{f,V_c^{clus}}^{Networking} \leftarrow 0$
else
  $N_c \leftarrow$ **List-of-neighbors**$(V_c^{clus}, LCs_i^k, cut_{remov}^k)$
  $N_c^{assign} \leftarrow$ **List-of-assigned**(N_c)
  $N_c^{unassign} \leftarrow$ **List-of-Unassigned**(N_c)
  if ($size(N_c^{assign}) = 0$) **then**
    calculate AC_f^{net} using equation (A.18)
    calculate BW_c^{Ngh} using equation (A.19)
    $C_{f,V_c^{clus}}^{Networking} \leftarrow AC_f^{net} * BW_c^{Ngh}$
  else
    // Evaluate the networking cost with assigned neighbors
    for ($V_n^{clus} \in N_c^{assign}$) **do**
      $Ass(n) \leftarrow$ **get-assigned-provider**(V_n^{clus})
      $C_{f,V_c^{clus}}^{Networking} \leftarrow C_{f,V_c^{clus}}^{Networking} + bw_{c,n}^{cut} * C_{f,Ass(n)}^{net}$
    end for
    // Evaluate the networking cost with non-assigned neighbors
    calculate AC_f^{net} using equation (A.18)
    calculate BW_c^{Ngh} using equation (A.19)
    $C_{f,V_c^{clus}}^{Networking} \leftarrow C_{f,V_c^{clus}}^{Networking} + AC_f^{net} * BW_c^{Ngh}$
  end if
end if
$C_{f,V_c^{clus}}^{Aggregate} \leftarrow C_{f,V_c^{clus}}^{Hosting} + C_{f,V_c^{clus}}^{Networking}$
return $C_{f,V_c^{clus}}^{Aggregate}$

$$BW_c^{Ngh} = \sum_{V_n^{clus} \in N_c^{unassign}} bw_{c,n}^{cut} \tag{A.19}$$

L'étape suivante consiste à sélectionner les meilleurs fournisseurs pour servir les grappes de VMs, suivant une approche de meilleur ajustement (Best-Fit) [162] basée sur la métrique du coût global.

A.3.3 Algorithme du meilleur ajustement: Cost-Aware Best-Fit Matching

Cette approche d'allocation a été sélectionnée vu sa pertinence et efficacité en termes de performances et qualité des solutions, y compris la rapidité du temps de convergence, l'optimisation de l'utilisation des ressources et la maximisation des requêtes acceptées. Les capacités d'hébergement disponibles dans la fédération (ressources locales et quotas proposés par les autres membres) représentent les ***boîtes "bins"*** à remplir. L'ensemble des k sous-graphes issus de la décomposition des requêtes et ayant des exigences variées en ressources, sont les ***objets "items"*** à empaqueter. Pour assurer un meilleur profit, nous avons adapté l'algorithme du meilleur ajustement en y intégrant la métrique du coût global, comme illustré par la procédure 6.

Pour l'affectation des ressources, l'algorithme 6 commence par trier les k sous-graphes de la requête en ordre décroissant de leurs capacités demandées, et estime pour chacun son coût d'allocation global chez les différents fournisseurs selon l'algorithme 5. Chaque sous-graphe est assigné au fournisseur le moins cher ayant des capacités suffisantes. Le processus se répète jusqu'à ce que tous les composants de la requête soient placés et empaquetés autant que possible chez les meilleurs fournisseurs. Si un sous-groupe donné ne peut pas être satisfait, le processus d'affectation s'arrête pour passer à la prochaine valeur de partitions k (voir Algorithme 7). Finalement, la matrice d'affectation résultante est retournée au programme principal pour valider l'allocation. Il convient de noter que cette étape représente juste un phase de sélection de fournisseurs sans aucune allocation effective. Ces decisions seront validées par l'algorithme principal (7), seulement si les critères de coûts sont vérifiés et la partition optimale (k-coupe) de la requête est déterminée.

A.3.4 Description de l'approche heuristique

L'heuristique proposée utilise les procédures décrites par les algorithmes 5 et 6 pour déterminer le partitionnement optimal des requêtes. L'algorithme sélectionne les meilleurs

Algorithm 6 Providers selection for hosting requests partitions

function: **Cost-Aware-Best-Fit-Assignment**$(LCs_i^k, F, cut_{remov}^k)$;

Input: The list of k clusters $LCs_i^k = \{V_1^{clus}, V_2^{clus}, \ldots, V_k^{clus}\}$ to allocate, the federation providers $F = \{cp_1, cp_2, ..., cp_f, ..., cp_m\}$ and their offerings, the set of removed $(k-1)$ cuts partitioning the graph $cut_{remov}^k = \{e\_remov_{(c,c')}^{cut}; c, c' \leq k\}$

Output: A feasible cost-effective assignment plan $assign\_matrix[k]$,if exists, specifying the list of best providers to host the k candidate clusters.

1: **Initialize:** $assign\_matrix[k] \leftarrow null$; $cost\_matrix[k,m] \leftarrow null$; $remain\_capacity\_matrix[m] \leftarrow null$; $nb_{Satisfied}^{clus} \leftarrow 0$;
2: **for** ($cp_f \in F$) **do**
3: $remain\_capacity\_matrix[f] \leftarrow$ **remaining-capacity**(f)
4: **end for**
5: Sort the list of VM-clusters LCs_i^k in decreasing order of their total needed resources.
6: **for** ($V_c^{clus} \in LCs_i^k$) **do**
7: boolean $assigned \leftarrow false$
8: **for** ($cp_f \in F$) **do**
9: $cost\_matrix[c,f] \leftarrow$ **Aggregate-Cost-Cluster**$(V_c^{clus}, cp_f, LCs_i^k, cut_{remov}^k)$
10: **end for**
11: Sort the list of providers F in increasing order of their aggregate allocation cost for cluster V_c^{clus} and in increasing remaining hosting capacities in case of equal costs.
12: **for** ($cp_f \in F$) **do**
13: **if** cp_f has enough resources to host V_c^{clus} **then**
14: $assigned \leftarrow true$
15: $assign\_matrix[c] \leftarrow f$
16: update $remain\_capacity\_matrix[f]$
17: $nb_{Satisfied}^{clus} \leftarrow nb_{Satisfied}^{clus} + 1$
18: **break**
19: **end if**
20: **end for**
21: **if** ($assigned = false$) **then**
22: **break**
23: **end if**
24: **end for**
25: **if** ($nb_{Satisfied}^{clus} \neq k$) **then**
26: $assign\_matrix \leftarrow null$
27: **end if**
28: **return** $assign\_matrix$

décisions d'internalisation et d'externalisation maximisant le profit des fournisseurs, selon les étapes décrites dans 7.

L'heuristique commence par trier les requêtes R selon un ordre décroissant de leurs revenus potentiels afin de privilégier les plus rentables. Pour chacune des requêtes i, l'algorithme détermine les meilleurs fournisseurs pour l'allocation selon les étapes suivantes. Tout d'abord, l'arbre de Gomory-Hu associé au graphe de VMs reçu $G_i = (V_i, Tr_i)$ est établi pour définir les éventuelles décisions de son partitionnement, comme le décrit la Figure A.3. Ensuite, l'algorithme itère sur les valeurs de k partitions possibles ($k \in [1, \max\{m, |V_i|\}]$), pour évaluer leurs coûts d'allocation et sélectionner la meilleure solution. Pour chaque valeur de k, l'heuristique fait appel à l'algorithme du meilleur ajustement (*Cost-Aware-Best-Fit-Assignment* 6) pour déterminer la liste des meilleurs fournisseurs satisfaisant les exigences des k sous-graphes de la requête LCs_i^k. Si aucune solution n'est faisable, l'algorithme passe directement à la valeur suivante de k pour partitionner davantage la requête et obtenir des sous-graphes plus petits, adaptés aux capacités et quotas des fournisseurs. Autrement, l'algorithme 7 évalue la qualité de la solution obtenue *assign_sol* en termes de coût d'allocation global avant de l'appliquer. Pour chaque itération, le coût actuel de la k-coupe $alloc\_cost_i^k$ est comparé à celui de la $(k-1)$-coupe $alloc\_cost_i^{best}$, jusqu'à identifer le meilleur coût minimum et le nombre optimal K_{opt} de partitions à utiliser pour distribuer la requête i à travers la fédération.

Algorithm 7 NCAFedRA heuristic

Input: A batch of requests $R = \cup_i$, the list of federation providers $F = \{cp_1, cp_2, ..., cp_f, ..., cp_m\}$ and their offerings

Output: A distributed allocation plan for requests R minimizing the overall costs.

Initialize: $accept\_matrix[\ |R|\ ] \leftarrow null$; $VM\_alloc\_plan[\ |R|, max(|V_i|)\ ] \leftarrow null$; $assign\_sol[m] \leftarrow null$; $LCs_i^{best} \leftarrow null$; $assign^{best}[m] \leftarrow null$;
Sort requests i in R in decreasing order of their profitability (selling prices)
for ($i \in R$) **do**
  if (**total-capacity**$(i) \geq$ **total-remaining-quotas**(F)) **then**
    $accept\_matrix[i] \leftarrow rejected$
    $VM\_alloc\_plan[i, l] \leftarrow -1$, for all $l \in V_i$
  else
    boolean $solution\_found \leftarrow false$
    $alloc\_cost_i^{best} \leftarrow \infty$
    $profit_i^{best} \leftarrow -\infty$
    Construct the **Gomory-Hu tree** of i and obtain T_i^{GH} containing V_i VMs (nodes) and $(|V_i| - 1)$ links $cuts_i^{GH}$
    Sort $cuts_i^{GH}$ by increasing weights
    for ($k \in [1, m]$) **do**
      $cut_{remov}^k \leftarrow$ the lightest $(k-1)$ links from $cuts_i^{GH}$ if exists, else **break**
      $LCs_i^k \leftarrow$ the k disjoints clusters resulted by the removal of cut_{remov}^k
      $assign\_sol \leftarrow$ **Cost-Aware-Best-Fit-Assignment**$(LCs_i^k, F, cut_{remov}^k)$
      if ($assign\_sol \neq null$) **then**
        $solution\_found \leftarrow true$
        // Calculate this resulted allocation cost C(k) for this k-cut partitions
        $alloc\_cost_i^k \leftarrow$ **total-allocation-cost**$(LCs_i^k, assign\_sol, i)$ (5.6)
        if ($alloc\_cost_i^k > alloc\_cost_i^{best}$) **then**
          // Exit iterations : optimal solution is found for $(k-1)$ *partitions*
          break
        else
          $alloc\_cost_i^{best} \leftarrow alloc\_cost_i^k$
          // Memorize this allocation solution and evaluate the next k value
          $LCs_i^{best} \leftarrow LCs_i^k$
          $assign^{best} \leftarrow assign\_sol$
        end if
      end if
    end for
    $profit_i^{best} \leftarrow$ (**total-selling-revenue**$(i) - alloc\_cost_i^{best}$)
    if ($solution\_found$ & ($profit_i \geq R_0$)) **then**
      $accept\_matrix[i] \leftarrow accepted$
      // Effective Allocation: update resources and quotas
      $VM\_alloc\_plan[i, |V_i|] \leftarrow$ **allocate-request**$(LCs_i^{best}, assign^{best})$
      update-profit(cp_j)
    else
      $accept\_matrix[i] \leftarrow rejected$
      $VM\_alloc\_plan[i, l] \leftarrow -1$, for all $l \in V_i$
    end if
  end if
end for
return $accept\_matrix$, VM_alloc_plan

Bibliography

[1] F. Gens. Worldwide and regional public it cloud services 2014–2018 forecast. White Paper, http://www.idc.com/getdoc.jsp?containerId=251730, 2015.

[2] C. Oppenheimer. Which is less expensive: Amazon or self-hosted? https://gigaom.com/2012/02/11/which-is-less-expensive-amazon-or-self-hosted/, Feb. 2012.

[3] J. Wilkes and C. Reiss. Cluster workload traces. https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md, Aug. 2015.

[4] P. Mell and T. Grance. Effectively and securely using the cloud computing paradigm. http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-computing-v26.ppt, 2009.

[5] A.N. Toosi, R.N. Calheiros, and R. Buyya. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1):7:1–7:47, May 2014.

[6] O. Rogers and W. Fellows. The cloud pricing codex–2013, November 2013.

[7] M. Mihailescu and Y.M. Teo. Dynamic resource pricing on federated clouds. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 513–517, May 2010.

[8] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[9] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[10] M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. Cloud computing: Distributed internet computing for it and scientific research. *Internet Computing, IEEE*, 13(5):10–13, Sept 2009.

[11] Cisco. Cisco global cloud index: Forecast and methodology, 2014–2019. White Paper, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf, 2015.

[12] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 500–507, July 2011.

[13] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow. Blueprint for the intercloud - protocols and formats for cloud computing interoperability. In *Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on*, pages 328–336, May 2009.

[14] Global Inter-Cloud Technology Forum (CICTF). Use cases and functional requirements for inter-cloud computing. White Paper, http://www.ttc.or.jp/files/8614/1214/5480/GICTF_Whitepaper_20100809.pdf, Aug. 2010.

[15] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. 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*, 41(1):23–50, January 2011.

[16] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, June 2009.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[18] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1): 50–55, December 2008.

[19] L. Schubert, K.G. Jeffery, and B. Neidecker-Lutz. *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010:–expert Group Report.* European Commission, Information Society and Media, 2010.

[20] P.M Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.

[21] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. Sp 500-292. nist cloud computing reference architecture. Technical report, Gaithersburg, MD, United States, 2011.

[22] M. Cafaro and G. Aloisio. Grids, clouds, and virtualization. In *Grids, Clouds and Virtualization*, pages 1–21. Springer London, 2011.

[23] J.E. Smith and R. Nair. The architecture of virtual machines. *Computer*, 38(5): 32–38, May 2005.

[24] M. Nelson, B. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '05, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association.

[25] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[26] M. Rosenblum and T. Garfinkel. Virtual machine monitors: current technology and future trends. *Computer*, 38(5):39–47, May 2005.

[27] Understanding full virtualization, paravirtualization, and hardware assist. whitepaper, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf, 2007.

[28] Kvm (kernel-based virtual machine). http://http://www.linux-kvm.org/, 2015.

[29] Vmware esxi. https://www.vmware.com/products/esxi-and-esx/, 2015.

[30] Xen. http://www.xenproject.org/, 2015.

[31] Microsoft hyper-v. http://www.microsoft.com/Hyper-V, 2015.

[32] S. Soltesz, H. Pötzl, M.E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287, March 2007.

[33] M.G. Xavier, M.V. Neves, F.D. Rossi, T.C. Ferreto, T. Lange, and C.A.F. De Rose. Performance evaluation of container-based virtualization for high performance

computing environments. In *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, PDP '13, pages 233–240, Washington, DC, USA, 2013. IEEE Computer Society.

[34] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. *technology*, 28:32, 2014.

[35] Linux containers. https://linuxcontainers.org/, 2015.

[36] Docker. https://www.docker.com/, 2015.

[37] Openvz. https://openvz.org/, 2015.

[38] M.J. Scheepers. Virtualization and containerization of application infrastructure: A comparison. volume 21. University of Twente, 2014.

[39] 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*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association.

[40] I. Goiri, F. Julià, J.O. Fitó, M. Macías, and J. Guitart. Resource-level qos metric for cpu-based guarantees in cloud providers. In Jörn Altmann and Omer F. Rana, editors, *GECON*, volume 6296 of *Lecture Notes in Computer Science*, pages 34–47. Springer, 2010.

[41] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P.Y. Wang. Seamless live migration of virtual machines over the man/wan. *Future Gener. Comput. Syst.*, 22(8):901–907, October 2006.

[42] G. Kecskemeti, G. Terstyanszky, P. Kacsuk, and Z. Neméth. An approach for virtual appliance distribution for service deployment. *Future Gener. Comput. Syst.*, 27(3):280–289, March 2011.

[43] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, Nov 2008.

[44] Salesforce. www.salesforce.com, 2015.

[45] Box inc. https://www.box.com/, 2015.

[46] Dropbox. https://www.dropbox.com/, 2015.

[47] Google apps. http://www.google.com/intx/fr/enterprise/apps/business/, 2015.

[48] Google app engine. https://cloud.google.com/appengine/, 2015.

[49] Microsoft azure cloud services. http://azure.microsoft.com/services/cloud-services/, 2015.

[50] Pivotal cloud foundry. https://pivotal.io/platform-as-a-service/pivotal-cloud-foundry, 2015.

[51] S. Bhardwaj, L. Jain, and S. Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1):60–63, 2010.

[52] Amazon ec2. http://aws.amazon.com/ec2/, 2015.

[53] Microsoft azure iaas. http://azure.microsoft.com/en-us/services/virtual-machines/, 2015.

[54] Google compute engine. https://cloud.google.com/compute/, 2015.

[55] Ibm smartcloud enterprise. http://www.ibm.com/cloud-computing/iaas.html, 2015.

[56] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I.M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmerich, and F. Galán. The reservoir model and architecture for open federated cloud computing. *IBM J. Res. Dev.*, 53(4):535–545, July 2009.

[57] R. Buyya, R. Ranjan, and R.N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 13–31, Berlin, Heidelberg, 2010. Springer-Verlag.

[58] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. Three-phase cross-cloud federation model: The cloud sso authentication. In *Second International Conference on Advances in Future Internet (AFIN)*, pages 94–101, July 2010.

[59] A.J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R.M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S.K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan. Optimis: A holistic approach to cloud service provisioning. *Future Gener. Comput. Syst.*, 28(1):66–77, January 2012.

[60] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. How to enhance cloud architectures to enable cross-federation. In *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 337–345, July 2010.

[61] Amazon EC2. Summary of the amazon ec2 and amazon rds service disruption in the us east region. aws.amazon.com/message/65648, Apr. 2011.

[62] Amazon EC2. Summary of the aws service event in the us east region. aws.amazon.com/message/67457, July 2012.

[63] Google App Engine. Post-mortem for february 24th, 2010 outage. https://groups.google.com/forum/#!topic/google-appengine/p2QKJ0OSLc8, 2010.

[64] B. Laing. Windows azure service disruption update. https://azure.microsoft.com/blog/windows-azure-service-disruption-update/, Feb. 2012.

[65] D. Petcu. Portability and interoperability between clouds: Challenges and case study. In *Proceedings of the 4th European Conference on Towards a Service-based Internet*, ServiceWave'11, pages 62–74, Berlin, Heidelberg, 2011. Springer-Verlag.

[66] F. Gonidis, I. Paraskakis, and D. Kourtesis. Addressing the challenge of application portability in cloud platforms. pages 565–576, 2012.

[67] F. Gonidis, A.J.H. Simons, I. Paraskakis, and D. Kourtesis. Cloud application portability: An initial view. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 275–282, New York, NY, USA, 2013. ACM.

[68] L. Rodero-Merino, L.M. Vaquero, V. Gil, F. Galán, J. Fontán, R.S. Montero, and I.M. Llorente. From infrastructure delivery to service management in clouds. *Future Gener. Comput. Syst.*, 26(8):1226–1240, October 2010.

[69] T. Bittman. The evolution of the cloud computing market. http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market/, 2015.

[70] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. Masoud Sadjadi, and M. Parashar. Cloud federation in a layered service model. *J. Comput. Syst. Sci.*, 78(5):1330–1344, September 2012.

[71] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muñoz, and G. Tofetti. Reservoir - when one cloud is not enough. *Computer*, 44(3):44–51, March 2011.

[72] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti. Cloud federations in contrail. In *Proceedings of the 2011 International Conference on Parallel Processing*, Euro-Par'11, pages 159–168, Berlin, Heidelberg, 2012. Springer-Verlag.

[73] T. Aoyama and H. Sakai. Inter-cloud computing. *Business & Information Systems Engineering*, 3(3):173–177, 2011.

[74] J. Tordsson, R.S. Montero, R. Moreno-Vozmediano, and I.M. Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Gener. Comput. Syst.*, 28(2):358–367, February 2012.

[75] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T.D. Nguyen. Reducing electricity cost through virtual machine placement in high performance computing clouds. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 22:1–22:12, New York, NY, USA, 2011. ACM.

[76] R. Moreno-Vozmediano, R.S. Montero, and I.M. Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45(12): 65–72, Dec 2012.

[77] D. Petcu. Consuming resources and services from multiple clouds. *J. Grid Comput.*, 12(2):321–345, June 2014.

[78] T. Subramanian and N. Savarimuthu. A study on optimized resource provisioning in federated cloud. *CoRR*, abs/1503.03579, 2015.

[79] N. Grozev and R. Buyya. Inter-cloud architectures and application brokering: Taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2012.

[80] The opennebula project. http://opennebula.org/, 2015.

[81] The openstack project. https://www.openstack.org/, 2015.

[82] Z. Zhang, C. Wu, and D.W.L. Cheung. A survey on cloud interoperability: Taxonomies, standards, and practice. *SIGMETRICS Perform. Eval. Rev.*, 40(4):13–22, Apr. 2013.

[83] J.L. Lucas-Simarro, R. Moreno-Vozmediano, R.S. Montero, and I.M. Llorente. Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In *International Conference on High Performance Computing and Simulation (HPCS)*, pages 1–7, July 2011.

[84] Gartner. Gartner says cloud consumers need brokerages to unlock the potential of cloud services. http://www.gartner.com/newsroom/id/1064712, July 2009.

[85] F. Jrad, J. Tao, and A. Streit. Sla based service brokering in intercloud environments. In Frank Leymann, Ivan Ivanov, Marten van Sinderen, and Tony Shan, editors, *CLOSER*, pages 76–81. SciTePress, 2012.

[86] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski. Introducing stratos: A cloud broker service. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 891–898, June 2012.

[87] A. Cuomo, G. Di Modica, S. Distefano, A. Puliafito, M. Rak, O. Tomarchio, S. Venticinque, and U. Villano. An sla-based broker for cloud infrastructures. *Journal of Grid Computing*, 11(1):1–25, 2013.

[88] J.L. Lucas-Simarro, R. Moreno-Vozmediano, R.S. Montero, and I.M. Llorente. Scheduling strategies for optimal service deployment across multiple clouds. *Future Gener. Comput. Syst.*, 29(6):1431–1441, August 2013.

[89] S. Chaisiri, Bu-Sung Lee, and D. Niyato. Optimal virtual machine placement across multiple cloud providers. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 103–110, Dec 2009.

[90] S. Chaisiri, B.S. Lee, and D. Niyato. Optimization of resource provisioning cost in cloud computing. *Services Computing, IEEE Transactions on*, 5(2):164–177, Apr. 2012.

[91] W. Li, J. Tordsson, and E. Elmroth. Modeling for dynamic cloud scheduling via migration of virtual machines. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CLOUDCOM '11, pages 163–171, Washington, DC, USA, 2011. IEEE Computer Society.

[92] mOSAIC Project. Open-source api and platform for multiple clouds. http://www.mosaic-cloud.eu/, 2010-2013.

[93] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu. An analysis of mosaic ontology for cloud resources annotation. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 973–980, Sept 2011.

[94] Compatibleone project. http://www.compatibleone.org/, 2010-2013.

[95] S. Yangui, I.J. Marshall, J.P. Laisne, and S. Tata. Compatibleone: The open source cloud broker. *Journal of Grid Computing*, 12(1):93–109, 2014.

[96] Slasoi project. http://sla-at-soi.eu/, 2008-2011.

[97] E. Badidi. A cloud service broker for sla-based saas provisioning. In *International Conference on Information Society (i-Society)*, pages 61–66, June 2013.

[98] Optimis project. http://www.optimis-project.eu/project, 2010-2013.

[99] S.K. Nair, S. Porwal, T. Dimitrakos, A.J. Ferrer, J. Tordsson, T. Sharif, C. Sheridan, M. Rajarajan, and A.U. Khan. Towards secure cloud bursting, brokerage and aggregation. In *IEEE 8th European Conference on Web Services (ECOWS)*, pages 189–196, Dec 2010.

[100] F. Fowley, C. Pahl, and L. Zhang. A comparison framework and review of service brokerage solutions for cloud architectures. In AlessioR. Lomuscio, Surya Nepal, Fabio Patrizi, Boualem Benatallah, and Ivona Brandić, editors, *Service-Oriented Computing – ICSOC 2013 Workshops*, volume 8377 of *Lecture Notes in Computer Science*, pages 137–149. Springer International Publishing, 2014.

[101] N.M Calcavecchia, A. Celesti, and E. Di Nitto. Understanding decentralized and dynamic brokerage in federated cloud environments. In *Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice*, pages 36–56. IGI Global, Nov. 2012.

[102] I.J. Marshal. The compatibleone accords platform. http://www.compatibleone.com/community/wp-content/uploads/2014/05/AccordsPlatformv1.4.pdf, Mar. 2012.

[103] T. Metsch and A. Edmonds. Open cloud computing interface - restful http rendering. https://www.ogf.org/documents/GFD.185.pdf, June 2011.

[104] I.J. Marshal and J.P. Laisné. The compatibleone resource description system (cords). http://www.compatibleone.com/community/wp-content/uploads/2014/05/CordsReferenceManualV2.15.pdf, Dec. 2013.

[105] A.N. Toosi, R.N. Calheiros, R.K. Thulasiram, and R. Buyya. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, pages 279–287, Sept 2011.

[106] I. Goiri, J. Guitart, and J. Torres. Characterizing cloud federation for enhancing providers' profit. In *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 123–130, July 2010.

[107] A.I. Avetisyan, R. Campbell, I. Gupta, M.T. Heath, S.Y. Ko, G.R. Ganger, M.A. Kozuch, D. O'Hallaron, M. Kunze, T.T. Kwan, K. Lai, M. Lyons, D.S. Milojicic, H.Y. Lee, Y.C. Soh, Ng.K. Ming, J.Y. Luke, and H. Namgoong. Open cirrus: A global cloud computing testbed. *Computer*, 43(4):35–43, April 2010.

[108] M.M. Hassan, B. Song, and E.N. Huh. A market-oriented dynamic collaborative cloud services platform. *annals of telecommunications - annales des télécommunications*, 65(11-12):669–688, 2010.

[109] Í. Goiri, J. Guitart, and J. Torres. Economic model of a cloud provider operating in a federated cloud. *Information Systems Frontiers*, 14(4):827–843, 2012.

[110] D. Niyato, A.V. Vasilakos, and Zhu Kun. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 215–224, May 2011.

[111] W. Li, P. Svard, J. Tordsson, and E. Elmroth. A general approach to service deployment in cloud environments. In *Second International Conference on Cloud and Green Computing (CGC)*, pages 17–24, Nov 2012.

[112] The open cloud computing interface (occi). http://occi-wg.org/about/specification/, 2015.

[113] The open virtualization format (ovf). https://www.dmtf.org/standards/ovf, 2015.

[114] The cloud data management interface (cdmi). http://www.snia.org/cdmi, 2015.

[115] The apache libcloud standard. https://libcloud.apache.org/, 2015.

[116] A.N. Toosi, R.K. Thulasiram, and R. Buyya. Financial option market model for federated cloud environments. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 3–12, Washington, DC, USA, 2012. IEEE Computer Society.

[117] M. Mihailescu and Y. Teo. Strategy-proof dynamic resource pricing of multiple resource types on federated clouds. In *Algorithms and Architectures for Parallel Processing*, volume 6081 of *Lecture Notes in Computer Science*, pages 337–350. Springer Berlin Heidelberg, 2010.

[118] H. Li, C. Wu, Z. Li, and F.C.M. Lau. Profit-maximizing virtual machine trading in a federation of selfish clouds. In *Proceedings of IEEE INFOCOM*, pages 25–29, April 2013.

[119] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad. Cloud computing pricing models: A survey. *International Journal of Grid & Distributed Computing*, 6(5): 93–106, 2013.

[120] Prices of amazon on-demand instances. http://aws.amazon.com/ec2/pricing/.

[121] Gogrid - a datapipe company. http://www.gogrid.com/, 2015.

[122] Amazon ec2 spot instances. https://aws.amazon.com/ec2/spot/, 2015.

[123] Dedicated server-arsys cloud. http://www.arsys.net/servers/dedicated, 2015.

[124] Vmware vcloud pricing. http://vcloud.vmware.com/uk/service-offering/pricing-guide, 2015.

[125] Cloudsigma pricing. https://www.cloudsigma.com/pricing/, 2015.

[126] Elastichosts cloud server pricing. https://www.elastichosts.com/pricing/, 2015.

[127] A. Singh, M. Korupolu, and D. Mohapatra. Server-storage virtualization: Integration and load balancing in data centers. In *SC - International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, Nov 2008.

[128] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128, May 2007.

[129] D. Borgetto, M. Maurer, G. Da-Costa, J. M. Pierson, and I. Brandic. Energy-efficient and sla-aware management of iaas clouds. In *Third International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy)*, pages 1–10, May 2012.

[130] H. Nguyen Van, F. Dang Tran, and J.M. Menaud. Autonomic virtual resource management for service hosting platforms. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD 2009)*, pages 1–8, May 2009.

[131] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, CLOUD '10, pages 228–235, Washington, DC, USA, 2010. IEEE Computer Society.

[132] J.O. Fito, I. Goiri, and J. Guitart. Sla-driven elastic cloud hosting provider. In *18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 111–118, Feb 2010.

[133] B. Javadi, J. Abawajy, and R. Buyya. Failure-aware resource provisioning for hybrid cloud infrastructure. *J. Parallel Distrib. Comput.*, 72(10):1318–1331, October 2012.

[134] B. Javadi, P. Thulasiraman, and R. Buyya. Cloud resource provisioning to extend the capacity of local resources in the presence of failures. In *IEEE 14th International Conference on High Performance Computing and Communication and IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, pages 311–319, June 2012.

[135] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Multicloud deployment of computing clusters for loosely coupled mtc applications. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):924–930, June 2011.

[136] R. Moreno-Vozmediano, R.S. Montero, and I.M. Llorente. Elastic management of web server clusters on distributed virtual infrastructures. *Concurr. Comput. : Pract. Exper.*, 23(13):1474–1490, September 2011.

[137] Y.C. Lee, C. Wang, J. Taheri, A.Y. Zomaya, and B.B. Zhou. On the effect of using third-party clouds for maximizing profit. In *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 381–390, Berlin, Heidelberg, 2010. Springer-Verlag.

[138] X. Zuo, G. Zhang, and W. Tan. Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. *IEEE Transactions on Automation Science and Engineering*, 11(2):564–573, April 2014.

[139] D. Breitgand, A. Marashini, and J. Tordsson. Policy-driven service placement optimization in federated clouds. *IBM Research Division, Tech. Rep*, 2011.

[140] E. Casalicchio and L. Silvestri. An inter-cloud outsourcing model to scale performance, availability and security. In *IEEE Fifth International Conference on Utility and Cloud Computing (UCC)*, pages 151–158, Nov 2012.

[141] N. Samaan. A novel economic sharing model in a federation of selfish cloud providers. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):12–21, Jan 2014.

[142] X. Xu, H. Yu, and X. Cong. A qos-constrained resource allocation game in federated cloud. In *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 268–275, July 2013.

[143] A. Amokrane, M.F. Zhani, R. Langar, R. Boutaba, and G. Pujolle. Greenhead: Virtual data center embedding across distributed infrastructures. *IEEE Transactions on Cloud Computing*, 1(1):36–49, Jan 2013.

[144] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, and A. Yumerefendi. Embedding virtual topologies in networked clouds. In *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, pages 26–29, New York, NY, USA, 2011. ACM.

[145] M.F. Zhani, Q. Zhang, G. Simona, and R. Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 18–25, May 2013.

[146] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue. Survivable virtual infrastructure mapping in virtualized data centers. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 196–203, June 2012.

[147] M.G. Rabbani, R.P. Esteves, M. Podlesny, G. Simon, L.Z. Granville, and R. Boutaba. On tackling virtual data center embedding problem. In *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 177–184, May 2013.

[148] M. Alicherry and T.V. Lakshman. Network aware resource allocation in distributed clouds. In *Proceedings IEEE INFOCOM*, pages 963–971, March 2012.

[149] M.M. Hassan, B. Song, and E.N. Huh. Distributed resource allocation games in horizontal dynamic cloud federation platform. In *IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, pages 822–827, Sept 2011.

[150] Ibm cplex solver. www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[151] O. Goldschmidt and D.S. Hochbaum. Polynomial algorithm for the k-cut problem. In *29th Annual Symposium on Foundations of Computer Science.*, pages 444–451, Oct 1988.

[152] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[153] H. Saran and V.V. Vazirani. Finding k-cuts within twice the optimal. In *32nd Annual Symposium on Foundations of Computer Science*, pages 743–751, Oct 1991.

[154] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[155] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155, 1990.

[156] A.V. Goldberg and K. Tsioutsiouliklis. Cut tree algorithms: An experimental study. *Journal of Algorithms*, 38(1):51–83, 2001.

[157] J. Cohen, L.A. Rodrigues, F. Silva, R. Carmo, A.L.P. Guedes, and E.P. Duarte. Parallel implementations of gusfield's cut tree algorithm.

[158] J. Cohen, L.A. Rodrigues, and E.P. Duarte. A parallel implementation of gomory-hu's cut tree algorithm. In *IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 124–131, Oct 2012.

[159] G.S. Rao, H.S. Stone, and T.C. Hu. Assignment of tasks in a distributed processor system with limited memory. *Computers, IEEE Transactions on*, C-28(4):291–299, April 1979.

[160] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, Nov 1993.

[161] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA, 2007. ACM.

[162] E.G. Coffman Jr., J. Csirik, and G.J. Woeginger. Approximate solutions to bin packing problems. Technical report, WOE-29, INSTITUT FR MATHEMATIK B, TU GRAZ, STEYRERGASSE 30, A-8010, 1999.

[163] A.V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM*, 45 (5):783–797, September 1998.

[164] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[165] Monitis monitoring service. http://www.monitis.com/, 2015.

[166] Amazon cloudwatch. http://aws.amazon.com/fr/cloudwatch/, 2015.