

LEBANESE AMERICAN UNIVERSITY

Game Theoretical Models for Cloud Federations

By

Ahmad Tarek Hammoud

A thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

School of Arts and Sciences

May 2019

THESIS APPROVAL FORM

Student Name: Ahmad Hammoud I.D. #: 201600245

Thesis Title : Game Theoretical Models for Cloud Federations

Program: Masters in Computer Science

Department: Computer Science and Mathematics

School: Arts and Sciences


The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Masters of Science in the major of Computer Science

Thesis Advisor's Name Dr. Azzam Mourad | Signature  | DATE: 02 / 05 / 2019
Day Month Year

Thesis Co-Advisor's Name Dr. Hadi Otork | Signature  | DATE: 02 / 05 / 2019
Day Month Year

Committee Member's Name Dr. Danielle Azar | Signature  | DATE: 02 / 05 / 2019
Day Month Year

Committee Member's Name Dr. Rony Touma | Signature  | DATE: 02 / 05 / 2019
Day Month Year

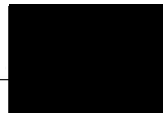
THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants the Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic formats and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: **Ahmad Hammoud**

Signature:



Date: **17** / **04** / **2019**

Day

Month

Year

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: **Ahmad Hammoud**

Signature:



Date: **17** / **04** / **2019**

Day

Month

Year

ACKNOWLEDGMENT

This project would not have been possible without the support of many people. Many thanks to my advisors, Dr. Azzam Mourad and Dr. Hadi Otrouk, who guided me through the process of this thesis. Also thanks to my committee members, Dr. Danielle Azar and Dr. Rony Touma, who offered guidance and support.

Also, I would like to thank Dr. Omar Abdel Wahab for the technical help he offered regarding this thesis.

And last but not least, I would like to thank my friends and family for the love and support they showed throughout the years.

Game Theoretical Models for Cloud Federations

Ahmad Tarek Hammoud

ABSTRACT

Cloud federation is an architecture that allows cloud providers to make use of their unallocated virtual machines, by combining their resources to serve a pool of cloud consumers whose requests cannot be handled by any of these providers alone. A client is willing to rent computing resources from the cloud providers or federations due to the advantages they can provide. The quality of service (QoS) is one of the main factors that attracts or discourages the clients from renting such services. What complicates the process is having the actual QoS delivered being worse than the promised QoS. Such could lead to the client changing federation, and the latter getting dissociated. Some of the main reasons that could result in worsening the QoS are encountering passive malicious cloud providers after the federation formation, and having unstable federation formation. In this thesis, we present solutions for such problems in order to increase the lifespan of the formed federations, by introducing a maximin game to prevent the malicious providers from accomplishing their wicked schemes without getting penalized, and advancing a genetic and an evolutionary game theoretical models for the federation formation process to bypass the dynamicity boundaries. Experiments conducted using CloudHarmony real-world dataset revealed that both of our solutions were able to increase the total profit obtained by the federations and ameliorate the QoS delivered, granting the cloud consumer a great experience with the service.

Keywords: Cloud Federation, Malicious Providers, Genetic Algorithm, Game Theory,

Evolutionary Game Theory, MaxMin Game, Security

TABLE OF CONTENTS

Chapter	Page
I Introduction	1
1.1 Motivation and Problem Statement	1
1.2 Objectives	4
1.3 Methodology and Contributions	4
1.3.1 On the Detection of Passive Malicious Cloud Providers in Cloud Federations	4
1.3.2 Cloud Federation Formation Using Genetic and Evolutionary Game Theoretical Models	5
1.4 Thesis Organization	6
II Background and Related Work	7
2.1 Introduction	7
2.2 Background	7
2.2.1 Cloud Computing	7
2.2.2 Federated Clouds	8
2.2.3 Genetic Algorithms	9
2.2.4 Game Theory	11
2.2.5 Evolutionary Game Theory	12
2.2.6 Hawk-Dove Evolutionary Game Example	14
2.3 Related Work	15
III On the Detection of Passive Malicious Cloud Providers in Cloud Feder- ations	18

3.1	Introduction	18
3.2	Problem Formulation	19
3.3	Detecting Passive Malicious Providers	21
3.4	Numerical Example	23
3.5	Experimental Evaluation	26
3.5.1	Implementation Setup	26
3.5.2	Results and Discussion	27
3.6	Conclusion	30
 IV Cloud Federation Formation Using Genetic and Evolutionary Game The-		
oretical Models		31
4.1	Introduction	31
4.2	Stable Cloud Federation Formation Problem	32
4.2.1	Approach Overview	32
4.2.2	System Model	33
4.2.3	Problem Formulation	34
4.3	Federation Formation using Genetic Algorithm	35
4.3.1	Encoding Scheme	36
4.3.2	Fitness Evaluation	38
4.3.3	Evolution Process	38
4.4	Federation Formation using Evolutionary Game Theory	40
4.4.1	Player Strategy	40
4.4.2	The Evolutionary Formation Game	41
4.5	Numerical Investigation	44
4.5.1	Solution Using Genetic Algorithm	45
4.5.2	Solution Using Evolutionary Game Theory	46
4.6	Experimental Evaluation	47
4.6.1	Experimental Setup	47
4.6.2	Results and Discussion	48
4.7	Conclusion	51

V Conclusion	53
Bibliography	56

LIST OF FIGURES

Figure		Page
1	Formed cloud federations	9
2	Flow chart of genetic algorithm	10
3	Flow Chart of the Evolutionary Game Theory	13
4	The Hawk-Dove Game: Variation of Payoffs	15
5	Detection rate	28
6	False negative	28
7	Profit	29
8	Latency	30
9	Availability	30
10	Cloud Federations: a VM's Dilemma	33
11	Total profit	48
12	VM utility	50
13	Quality of service	51

LIST OF TABLES

Table		Page
1	Prisoner's dilemma: payoff matrix	11
2	Hawk-Dove: payoff matrix	14
1	Probability Distribution of the Broker and the Malicious Provider(s) .	26
1	Requests	44
2	Initial solution	45
3	Solution after several generations	46
4	Solution using evolutionary game theory	46
5	Requests types	47

Chapter One

Introduction

1.1 Motivation and Problem Statement

The concept of cloud computing paved its way to be the first solution that business owners rely on when they are in need of an IT infrastructure, instead of having the burdens of managing their own infrastructure. Such concept has led to a massive increase in the demand, and to an expansion in the online business. Cloud providers have limited resources, which makes them unable to handle a huge stream of demands. Therefore, a new business architecture had arisen, called ‘federated clouds’, to address the management issues. The cloud federation concept aims to help cloud providers take advantage of their available unused virtual machines (VMs), by allowing them to join their resources together in order to serve a larger pool of requests that could not have been served by only one single cloud provider. A customer’s decision of whether to rent resources from a certain provider or not, is based on many factors. The main factor in an environment with competitive prices, is the Quality of Service (QoS). QoS is the measurement of the provided service’s performance. It is evaluated by the customer himself to decide if it is acceptable or not, based on status of the service, such as the availability, and response time. The availability represents the percentage of times the VMs were able to execute their assigned tasks normally, without being unavailable. The response time is the time taken to respond to a request. A customer would always

prefer to have good QoS, which can be obtained by having high availability and low response time. A high availability ensures the VMs being up and ready for processing requests at random times. And the lower the response time is, the faster the packets are received. Vice-versa, high response time and low availability will result in a decreased QoS. A customer might request VMs with good QoS, but once rented, the performance of these VMs might drop due to many reasons. It can be software, hardware, and/or network related factors involved in reducing the promised performance.

In an environment where a customer gets served by a federation instead of one single provider, the chances of having unstable QoS increases. Apart from the factors that affects the QoS delivered by one provider, a federation may confront some additional threats. Some of these threats could be having malicious providers engaged within the federation, or having an unstable federation formation.

In a particular federation, a cloud provider is responsible of dedicating the promised computing resources to it, in exchange of receiving a portion of the total profit. It is considered to be a well-behaved cloud provider in this case. Whereas on the other hand, a cloud provider is said to be malicious, if it has the intention of causing harm to the federation, or to another particular provider. Two types of malicious providers exist, the active ones and the passive ones. An active malicious cloud provider has the motive to launch an attack at the federation, or at another cloud provider(s) in order to directly harm them. Different from active malicious, passive malicious providers aim only at illegally increasing their own benefits and hence gaining advantage over the other providers. Specifically, passive malicious providers might misbehave by promising to provide a certain amount of VMs to the federation(s) and then reneging on their promises to dedicate those VMs to their own requests. The objectives of passive malicious providers in such a scenario are two-fold. On the one hand, they aim at increasing their own profits through outsourcing requests to other providers in the federation but refraining from helping back with the aim of serving resources. On the

other hand, those malicious providers might seek also to decrease the reputation scores of some federations to exclude them from the competition and hence increase their own market share. None of the proposed approaches in the literature has tackled the problem of encountering passive malicious providers in the formed federations [17], [11], [23]. Note that the problem of passive malicious providers has been addressed in [30]. However, the main difference between the work done in [30] and our work is that the former aims at excluding the passive malicious providers at the formation's level, but they do not advance any detection mechanism to detect their presence after federations' formation. This increases the risk of having passive malicious misbehavior in case some providers decide to change their behavior after the federations get formed.

Apart from having malicious providers in the formed federations, the fact of having an unstable federation formation can affect the promised QoS, by increasing the response time, and decreasing the availability of the machines. A stable cloud federation is a federation structure in which no cloud provider has incentive to deviate from its current federation and join another one, or leave the federation in order to form a new one. The absence of stability will cause the QoS to be decreased over time, what leads to the clients being unsatisfied with the service they are getting, and having them considering different cloud providers. The loss of these clients will result in a decrease of reputation and profit for the cloud federation. The real challenge is finding the optimal way to form a stable federation, while maximizing the total revenue. Several approaches have been proposed, however, most of the reported work in the literature did not take into consideration the stability [23], [11]. Whereas the work claimed achieving stability, had restrictions on the participants that prevent some cloud providers from participating in the federations [6]. Other papers ignored the fact that providers might reassign their resources to federations formed based on another set of requests [17].

1.2 Objectives

The main objective of this thesis is to find a proper way of forming stable and passive providers-free federations and monitoring them, where the promised QoS is no different than the actual QoS. Such property will ensure the survivability of these federations. Our objectives can be summarised as follows:

- Preventing the passive malicious providers from accomplishing their wicked schemes without getting penalised. This way they will have to be cautious when planning to be selfish.
- Achieving optimality in profit and stability among federations, which will lead to a better reputation and to an increase in the profit on the long run.

1.3 Methodology and Contributions

In this thesis, we focus on finding solutions for the listed objectives in the previous section. We propose a solution addressing each problem as an attempt to decrease the chances of the federations being destroyed.

1.3.1 On the Detection of Passive Malicious Cloud Providers in Cloud Federations

First, we propose a maxmin game theoretical model that helps the cloud broker, a third-party charged with managing the cloud federations, minimize the attack success chances of the passive malicious providers under a limited amount of resources. The strategy of the passive malicious providers is to distribute their misbehavior over a set of federations with the aim of maximizing their success chances. For example, a passive malicious provider might misbehave 10% of times in one federation, 13% in a second federation, and 7% in a third federation. In this way, the malicious behavior will not be concentrated in only one (or a few) federation(s) to avoid being

easily detected by the broker. On the other hand, given a limited amount of resources that can be dedicated to passive malicious providers detection, the broker's strategy is to distribute the monitoring load over the set of federations to minimize the passive malicious providers' attack success chances maximization, while not exceeding the limited amount of resources. By solving the game, we provide the broker with the optimal distribution of the monitoring load over the set of federations that maximizes the probability of detecting passive malicious misbehavior. This leads to improving the performance of the federations in responding to customers' requests compared to the state-of-the-art, and hence maximizing the overall revenue of the cloud providers. The main contributions of this work can be summarized by the following points:

- Devising a maxmin game theoretical model between the cloud broker and the passive malicious providers. By solving the game, the broker learns the optimal distribution of the monitoring load over the set of federations that maximizes the probability of detecting passive malicious misbehavior. To the best of our knowledge, this work is the first that addresses the existence of passive malicious providers in the cloud federations architecture and proposes an intelligent resource-aware detection mechanism to limit their effect on the profitability of the federation.
- Maximizing the overall monetary profit of cloud providers as well as the QoS delivered to customers' requests.

1.3.2 Cloud Federation Formation Using Genetic and Evolutionary Game Theoretical Models

Then, we address the problem of constructing cloud federations by introducing a search heuristic genetic algorithm (GA), inspired by Darwin's natural evolution theory. However, despite the fact that it has a good performance as will be shown in the simulation section, it does not lead into stable federations that may affect the long term revenue. In addition, the convergence time was too long. Thus, we extend it by modeling the

formation process as a learning-based evolutionary game theoretical model, in which the conflict and cooperation between the cloud providers are considered in the presence of the dynamic strategy change [1], [15], [7], and [19]. The purpose of applying this approach is to reach a state where no one has incentive to break from his currently chosen strategy, known as evolutionary stable strategy. Therefore, by reaching such state, all cloud providers will be fully dedicated to their federations without the intention of leaving them and reallocating their VMs to other ones.

The main contributions of this work can be summarized by the following points:

- Maximizing the profit yielded by the cloud federations using Genetic Algorithms. Such model can guarantee that the federations are in continuous increase in the profit until reaching maximality.
- Modeling the stable formation problem as an Evolutionary Game Theory to bypass the dynamicity boundaries. By reaching evolutionary stable strategy, the population can survive any small mutant invasion.

1.4 Thesis Organization

The remaining of the thesis is organized as the following. Chapter Two provides background and related work addressing the federations' formation and their security. Chapter Three addresses the malicious providers problem, and how to reduce their effect on the formed federations by using a maxmin game theoretical model. Chapter Four focuses on forming highly stable federations, while maintaining a high profit, using genetic and evolutionary game theoretical models. Finally, Chapter Five concludes the thesis and presents suggestions for future work.

Chapter Two

Background and Related Work

2.1 Introduction

In this chapter, we present the background and related work needed for this thesis.

2.2 Background

In this section, we discuss the concepts of cloud computing, federated clouds, genetic algorithms, game theory, and finally, evolutionary game theory and its stability property.

2.2.1 Cloud Computing

As defined by the National Institute of Standards and Technology (NIST), ‘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’ [14]. In other words, it is the practice of delivering computation resources through the internet. The demand on cloud computing resources has increased in the past few years due to the various advantages

it provides [20] including relieving the burdens of hosting their own IT infrastructure, getting rid of the maintenance cost, and saving money by only paying for the resources and workloads used. Moreover, the performance of such services is typically monitored by experts, which leads to an increased QoS. It is typically offered and managed by cloud service providers. These providers offer three types of services; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

- IaaS is a model in which compute resources, such as servers and storages, would be made available for the clients. Amazon Web Services is considered to be a leading provider when it comes to offering infrastructure.
- PaaS consists of offering a platform for a client, on which he can deploy the software he needs. In this form, the operating system provided along with the hardware used are taken care of. Google App Engine is an example of a PaaS, it is a web framework used for hosting web applications.
- SaaS is a form where a software is offered on the cloud for consumers. An example of such type can be ‘Dropbox’, a service where a consumer can find contentment in storing his personal files online, since it is managed by experts.

One of the fundamentals of cloud computing’s IaaS form, is virtualization. It allows more than one virtual machine to be hosted on the same physical machine. With the help of a hypervisor, the main machine (host) can create several guest machines that provide the functionality needed to execute entire operating systems, each by which can be rented out independently. Statistics have shown that some of the cloud providers are in a continuous increase in profit like Amazon Web Service, which is estimated to acquire 49% more profit than last year [18], [8].

2.2.2 Federated Clouds

Federated clouds is a new architecture that allows cloud providers to make use of their unallocated virtual machines. It consists of merging the resources of two or more cloud service providers together, allowing them to increase their capacity of handling

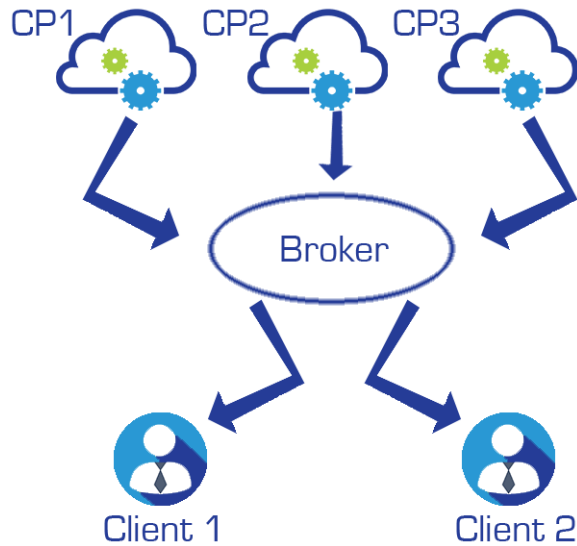


Figure 1: Formed cloud federations

such large requests instead of dropping them [4]. Such an architecture is beneficial for both parties; i.e. the client who's in need of compute resources, and the cloud provider having additional resources. It improves the QoS for the clients' requests due the interoperability among providers from one side, and from another side, it allows the providers to rent out their unused resources, which will increase their profit, and to expand their geographical footprints without the need of new points of presence. A cloud broker is responsible for the process of managing these cloud providers, and assuring that the service the client requested is being provided [14].

2.2.3 Genetic Algorithms

Genetic algorithm (GA) has been introduced by John Holland in 1960, and has been extended by his student Goldberg in 1989 [25]. It is a metaheuristic based on the Darwinism theory, formed by Charles Darwin. Like any other metaheuristic, GA is a search technique used to solve complex optimization problems. Typically, GAs attempt to find a near optimal solution in a relatively short time.

The main components of GA, as shown in Fig. 2, are (1) encoding, (2) evaluation, (3) selection, and (4) genetic operators.

Before solving the optimization problem, chromosomes (i.e. possible solutions) must

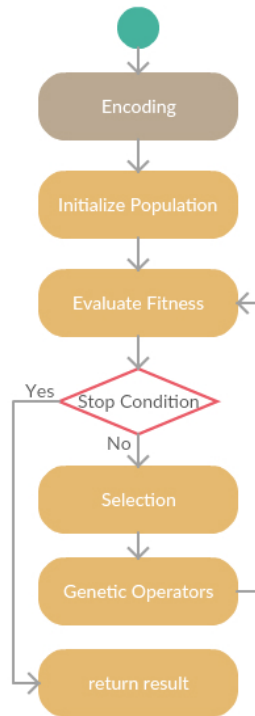


Figure 2: Flow chart of genetic algorithm

be encoded in a specific way. Each problem may depend on a special type of encoding in order to be solved. Some of the encodings that have already shown great success in practice are the binary encoding and the permutation encoding.

The evaluation step requires having a fitness function $f(c)$, such that a candidate c could be passed to that function, and the function would return a score for that candidate in order to be evaluated.

The selection process takes place after having all candidates' fitness evaluated. A portion of these candidates would get selected to breed a new generation. The fitter the candidate, the more likely to get selected, and vice-versa. Many methods exist for the selection of the best candidates, such as the roulette wheel selection, rank selection, tournament selection, etc...

The last component of the genetic algorithm is genetic operators. Genetic operators (i.e. crossover and mutation) are used to generate offsprings in order to be introduced to the next generation. Offsprings have mixed properties from both parents. Crossover is applied to create better chromosomes than the parents, where a cross-site is selected, and one of the sub-sequences (before or after the cross-site) of one parent is swapped

Table 1: Prisoner's dilemma: payoff matrix

		Prisoner A	
		betray	remain silent
Prisoner B	betray	-5, -5	-10, 0
	remain silent	0, -10	-1, -1

with the same subsequence of the other.

Genetic Algorithms has been applied in many fields such as computer gaming [16], investment strategies [5], digital circuits [29], and placement problems [28]. What makes GA attractive is its simplicity and effectiveness [22].

2.2.4 Game Theory

Game theory is the study of optimizing the outcome by mathematically determining the best strategy for the players under their circumstances. All possible outcomes of the games played by two or more can be represented by the decision tree, or what is called payoff matrix [10]. A payoff matrix is composed of three components; players, strategies, and outcomes. The players are those in charge of making a decision, the strategies represents all possible decisions that a player can make. The outcome is the intersection cell of the two selected strategies. It contains 2 variables; the left and right variables are the outcomes of the first and second player respectively.

Nash Equilibrium

Nash Equilibrium is a state where no player can obtain a better outcome by changing his strategy, knowing that the other participants are going to stick with their currently chosen decisions. Thus, there are no incentives for players to deviate from their current strategy. Some games might have more than one Nash Equilibrium, and some others might not have any. The prisoner's dilemma is an example analysed in game theory (Table 1). It consists of two criminals getting arrested and investigated on a particular crime. Each criminal can either betray the other, or remain silent during investigation. There are three possible cases for such dilemma. The first one consists of the two

criminals betraying each other, leading to both serving five years in jail. If a prisoner remains silent, while the other betrays, then the former serves ten years in jail, while the other is set free. In the last case, they both serve one year in jail by remaining silent. Even though serving one year in jail might be the best solution possible for both criminals in a cooperative game, one of the criminals might act selfish and decide to betray the other in order to get away from the punishment, therefore, this strategy doesn't constitute Nash equilibrium. Nevertheless, both criminals betraying each other is considered to constitute Nash Equilibrium because none of these players has incentives to remain silent, knowing the other is betraying. A strict Nash Equilibrium consists of the strategy selected being strictly more profitable for both players to stay on their current strategy, and not having one of them deviating into an alternative one [13].

2.2.5 Evolutionary Game Theory

Evolutionary game theory was introduced by Price and Maynard Smith in 1973 [26]. That notion shows that the analysis of game theory can be applied even if a player exhibits different forms of behavior, and doesn't always have to be reasonable. It focuses on the dynamics of strategy change, and which forms of behavior have the ability to persist among the players. In evolutionary game theory, the fitness of the individuals has to be evaluated in the context of the full population, in order to find out whether a particular player's strategy is successful or not. Individuals that are more fit in the population, tend to have their strategy being replicated by others [7].

Evolutionary stable strategy

Evolutionary stable strategy (ESS) is a strategy that once adopted by a population, it can survive even if it gets invaded by any small group of invaders. Once that strategy is reached, then it won't change in time. In case of an invasion, the invaders will die after a few generations, due to the stability of that strategy. Meaning that if a population was using strategy A , and a few mutants came to this population with an alternative strategy B - knowing that B being less profitable than A - then A is ESS if it can survive

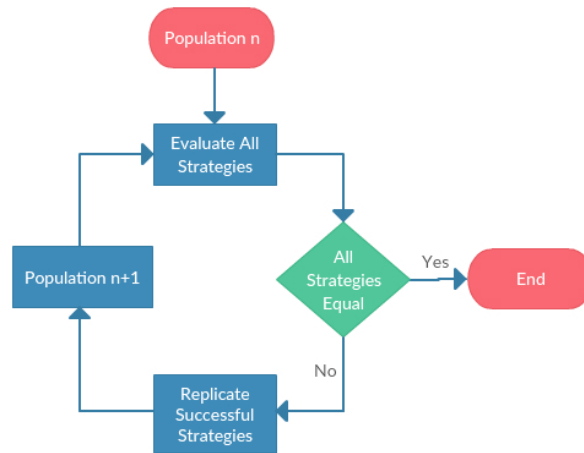


Figure 3: Flow Chart of the Evolutionary Game Theory

that invasion, and force the mutants to switch their strategy into A due to the natural selection.

Let $O(A, B)$ represent the outcome of an individual choosing strategy A meeting another with strategy B . A is stable if it constitute a strict nash equilibrium ($[O(A, A)] > [O(A, B)]$), or if $O(A, A) = O(B, A)$ and $O(A, B) > O(B, B)$. In such cases, no individual has incentives to break from his current strategy, even if the population got invaded by a few mutants.

The idea behind applying evolutionary game theory, is reaching the ESS, where stability occurs. Fig. 3 shows the flow chart of the game, where it can initially starts from any generation. A generation represents the set of players along with their current chosen strategies. The evaluation process takes place to weight all of the strategies that the current players are holding on to, then it classifies these strategies as good or bad, according to the utility of the player choosing it. Good strategies are more likely to be replicated by players who have chosen bad ones. After the replication phase occurs, a new generation will be born, such that strategies' utilities are slightly different due to the replication process. At any time, the evaluation process might lead to no changes as a result of having all strategies providing same payoff to their corresponding players; this is where ESS is established, therefore, all players are not dissatisfied about their chosen strategies, leading to a state where no player has any incentives of switching to a different one.

Table 2: Hawk-Dove: payoff matrix

		meets	
		Hawk	Dove
if	Hawk	-4, -4	2, 0
	Dove	0, 2	1

2.2.6 Hawk-Dove Evolutionary Game Example

The Hawk-Dove game (a.k.a the chicken game) is a model of conflict between players contesting over a certain shareable resource. Maynard has analysed this game in the presence of the dynamics of strategy change. The player can either be a dove or a hawk. Table 2 represents the payoff matrix of such game. Three different types of competitions may take place:

- If a hawk meets a dove, the former wins the fight, and takes the resource for himself (payoff = 2), while the latter can't benefit anything (payoff = 0)
- If two hawks meet, they both show aggressiveness seeking for the resource until the defeat of one of them, thus the average payoff, due to the damage dealt, is -4
- If two doves meet, they both share the same resource, thus their payoff is 1 each

Considering a population of birds playing this evolutionary game. As we can notice from Fig. 4, for every chosen strategy, the fitness of both species vary. Supposing we have no hawks entered the population, then the payoff of every dove is guaranteed to be 1. If we assume that 10% of the population became hawks after a certain invasion, then the probability of a hawk meeting a dove is going to be 90%. Meaning that the average payoff of a hawk will become $-4 \times 0.1 + 2 \times 0.9 = 1.4$, while the average payoff of a dove would be $1 \times 0.9 = 0.9$. The problem is that the percentage will always varies with no stable payoffs for both types. By applying evolutionary game theory, the ESS can be adopted such as, the population will maintain its current portion of the existing species as is, with time evolvement. Once the fitness of both types are equal,

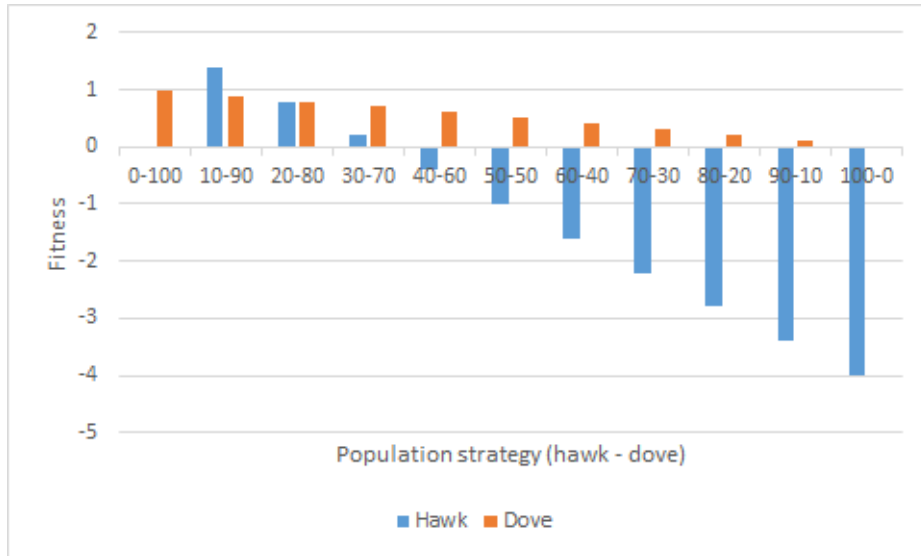


Figure 4: The Hawk-Dove Game: Variation of Payoffs

the population will adapt the current strategy even if it gets invaded by a few mutants attempting to change it. Such state can be noticed in Fig. 4, when the percentage of the hawks and doves are 20% and 80% respectively.

2.3 Related Work

In this section, we discuss some of the solutions proposed in the literature to handle the threats that negatively affect the survivability of cloud federations. In [24], the authors presented the open cloud federation model by merging computational resources provided by different cloud providers, as part of the Reservoir project. The project addressed similar problems such as the lack of interoperability among cloud providers, and how limited a cloud provider can be in terms of scalability. The focus, however, was on the architecture and functionality of such concept, with no formation mechanism for the federations.

In [11], the authors focused on enhancing the profit of cloud providers. They advanced a set of mathematical equations for a provider to make an optimal decision on where and when to allocate the computing resources. Their main objective was to maximize the provider's profit, and not the federation's. In [23], the authors derived a linear

optimization program whose solution helps providers in a certain federation to regulate their hosting and cooperation decisions on the basis of the encountered workload and the available pool of resources. In [3], the authors worked on maximizing the revenue of the cloud service providers by addressing the provider's resource selection process from the shared resource pool; the approach aimed to satisfy the users by serving them with the desired QoS level. The approach was based on a multi-choice multi dimension knapsack algorithm to optimize the resource selection process. In [27], the authors sought to assist providers in overcoming the resource limitation problem. They provided decision-making policies to help the providers decide whether to outsource requests to other federation members or to terminate spot VMs in order to free resources for more profitable VMs. [12] proposed a formation mechanism as a hedonic coalitional game based on factors like security level and reputation. The main contribution consists of minimizing the loss in security for cloud providers in order to avoid insecure federations. The approach didn't consider the impact of this proposed mechanism on the profit of the formed federations.

In [17], the authors proposed a formation mechanism for the cloud federations. The formation framework is based on an algorithm that relies on merging and splitting federations until finding the near optimal solution. They claimed that their mechanism can lead to a stable formation. However, they didn't take into consideration having new requests taking place after the federations have being formed, thus leading to providers leaving their federations and joining new ones, for seeking more profit as rational decision makers. [6] worked on forming the federation formation using trust as a main key among providers. They claimed that their formation mechanism would lead to stability, fairness, and maximization in the overall profit. Unfortunately, their policy places a constraint on the minimum number of VMs that should be part of a cloud provider if the provider intends to be part of the federation. Such a policy can exclude small cloud providers who can't meet that constraint, or cloud providers with a small number of VMs available.

In [2], the authors proposed a genetic approach for cloud brokering, called 'QBROK-

AGE'. The approach addresses the problem of forming a federation that meets the QoS requirements of the application needed. They did not take into consideration how this will affect the profit, nor how stable the formation is going to be.

Problem Statement: The limitations of such works consists of the following:

- The absence of a mechanism that deals with having passive malicious cloud service providers allocating their resources in some federations.
- Neglecting the presence of dynamic strategies when it comes to allocating the virtual machines within federations.

Such problems can lead to an inconsistency in the Quality of Service provided for the client, which will affect the reputation and outcome of the cloud federation.

Chapter Three

On the Detection of Passive Malicious Cloud Providers in Cloud Federations

3.1 Introduction

In this chapter we address the problem of having passive malicious cloud service providers allocating their resources in the cloud federations. Although plenty of solutions have been proposed trying to ensure the optimal formation of cloud federations, these approaches ignore the problem of encountering malicious providers that join federations to destroy them from inside and exclude some strong competitors from the market. To tackle this challenge, we propose in this letter a maximin game theoretical model which assists the broker, responsible for creating and managing federations, with maximizing the detection of such malicious providers. The challenge here is to deal with providers that try to minimize the detection maximization through distributing their misbehavior over several federations and changing their identities from time to time.

Experiments conducted using real data from the CloudHarmony dataset reveal that our solution maximizes the detection of malicious providers and improves the profit and Quality of Service of the federations compared to the sky federation model. The rest of the chapter is organized as follows. Section 3.2 represents the formulation of the

problem. Section 3.3 explains the approach. Section 3.4 provides a numerical example. Section 3.5 shows experiment results. Finally section 3.6 concludes the chapter.

3.2 Problem Formulation

We consider a set of federations $F = \{f_1, f_2, \dots, f_n\}$ that are formed to serve a pool of requests from cloud users. Each federation consists of a set of providers $P = \{p_1, p_2, \dots, p_k\}$ offering a set of virtual machines $V = \{v_1, v_2, \dots, v_l\}$, where each cloud provider is considered to be part of a certain federation if it dedicates one or more virtual machines to that federation. A cloud provider, according to NIST, is responsible of providing services to the cloud consumers in terms of infrastructure, computing, and storage [14], [8]. Providers may be either well-behaving or malicious. A cloud provider is considered to be well-behaving if it actually dedicates the proclaimed virtual machines to the federations it is member of. While, on the other hand, a cloud provider is said to be malicious if it lies about the number of VMs it will provide with the aim of saving resources and/or dedicating more resources toward fulfilling requests coming from its own users. Note that in this chapter we consider only the passive malicious misbehavior in which providers try to gain illegal advantage over other providers and increase their own market share of requests. Thus, active malicious attacks (e.g., Denial of Service) are out of the scope of this work. For example, a passive malicious provider might claim (upon federations formation) that it will provide 10 VMs to a certain federation, but in fact, it dedicates only 4 VMs and keeps the remaining 6 VMs to serve its own users. On the other hand, such a malicious provider would benefit from other providers' VMs when its resources become insufficient to keep up with its incoming requests. Such a malicious behavior would harm the federations by decreasing their payoff and reputation toward users.

To complicate the process of detection, a malicious provider can split its misbehavior over several federations in the sense that it can join federations with multiple

identities and misbehave (i.e., provide VMs less than promised) with a small probability and with different identities in more than one federation instead of concentrating its misbehavior on one federation, where it can be easily captured. To deal with such a type of malicious providers, the broker of the cloud federations is responsible, in our architecture, of monitoring the behavior of the cloud providers in the federations. The broker however has a limited amount of resources to be spent on the detection process. The reason is that the broker gets paid by the providers forming the federations to perform this task, where obviously these providers put a certain limit on the amount of money they will spend on this process in such a way that does not greatly affect their overall profit. Thus, the broker must exploit the available resources in an efficient manner by splitting the detection load over the existing federations so as to maximize the detection of malicious providers and respect at the same time the available resources constraints.

In order to guarantee the honesty of the detection process and avoid punishing some providers for uncontrollable circumstances (e.g., some providers might not be able to provide the proclaimed VMs for technical reasons rather than for being malicious), the broker sets a misbehavior threshold x above which a certain provider would be deemed to be malicious. Thus, if a provider is detected to be misbehaving more than $x\%$ of the times, it will get penalized by banning it from participating in further federations. Formally, let $\beta_{t,F} = \{\beta_t(f_1), \beta_t(f_2), \dots, \beta_t(f_n)\}$ denote the misbehavior probability distribution over the set F of joined federations at time t . It depicts the probability that a federation will misbehave. In order to cope with that, and with the limited amount of available resources that the broker possesses to be used for detection, he will have to set a mixed scheme corresponding of the optimum detection load probability distribution vector $\alpha_{t,F} = \{\alpha_t(f_1), \alpha_t(f_2), \dots, \alpha_t(f_n)\}$ over the set of federations F at time t , such that $\sum_{f_i \in F} \alpha_t(f_i) = 1$.

3.3 Detecting Passive Malicious Providers

The profit of the federations is mainly dependent on users' satisfaction, which is practically reflected through the monetary payment and the reputation score given by users to the federations. This means that the broker should always make sure that malicious providers are being detected in order to maintain high satisfaction levels from users. Mathematically, the payoff of the federation set F at the discrete time window $[t_1, t_2]$ is determined as follows:

$$U_{t_2+1}(F) = \sum_{f_i \in F} \delta(f_i) \times R(f_i) \times \gamma(f_i)_{[t_1, t_2]} \quad (\text{III.1})$$

where $\delta(f_i)$ is the profit obtained through renting out the virtual machines of federation f_i which can be calculated by subtracting the total cost of the VMs from the total revenue, $R(f_i)$ is the reputation score of the federation f_i which is computed proportionally to the availability of the federation, and $\gamma(f_i)_{[t_1, t_2]}$ is the average detection rate at the time window $[t_1, t_2]$ that can be computed as the following:

$$\gamma(f_i)_{[t_1, t_2]} = \sum_{\varepsilon=t_1}^{t_2} \frac{\beta_\varepsilon(f_i) \times \alpha_\varepsilon(f_i)}{t_2 - t_1} \quad (\text{III.2})$$

The calculations in the rest of the chapter will be all done at the current time $t_2 + 1$, so we can simplify $U_{t_2+1}(F)$, $\alpha_{t_2+1, F}$, and $\beta_{t_2+1, F}$ by referring to them as $U(F)$, α_F , and β_F respectively. Since the malicious providers' objective is to cause damage to the federations to increase their own market shares, the payoff of the malicious providers can be modeled as being the negation of the federations' utility, i.e.,

$$U(M) = -U(F) \quad (\text{III.3})$$

where $U(M)$ and $U(F)$ represent the utilities of the malicious providers and the federations respectively. This leads us to zero-sum games in which one player's payoff is the negation of the other player's payoff [21], since the objective of the broker is to

maximize profit, while the objective of the malicious provider is to harm and reduce the federations' payoff, this implies that the utility of the former is the negation of the latter's. In order for the malicious provider to succeed with its passive attack and minimize the federation's payoff, it must choose its probability distribution β_F over the federations' set wisely in such a way to complicate the broker's detection process and hence minimize the federations' utility, i.e.,

$$\arg \min_{\beta_F} U(F) \quad (\text{III.4})$$

On the other hand, the broker must choose the detection probability distribution α_F over the federations' set in order to maximize the malicious providers' minimization and hence maximize the federations utility, i.e.,

$$\arg \max_{\alpha_F} \min_{\beta_F} U(F) \quad (\text{III.5})$$

This forms a maximin game theoretical model in which the malicious providers are trying to minimize the federations' payoff to increase their own market share, and the broker, acting on behalf of the federations, is trying to choose the optimal detection strategy that maximizes the providers' minimization. The solution of the game can be obtained using Linear Programming, where the objective function of the broker can be rewritten as follows:

$$\begin{aligned} & \text{maximize} \quad \min_{\beta_F} \sum_{f_i \in F} \alpha(f_i) \times U(F) \\ & \text{subject to} \quad \sum_{f_i \in F} \alpha(f_i) = 1, \\ & \quad \quad \quad \alpha(f_i) \geq 0, \forall f_i \in F \end{aligned} \quad (\text{III.6})$$

To linearize the above equation, we define a variable y such that $y \leq \min_{\beta_F} \sum_{f_i \in F} \alpha(f_i) \times U(F)$ and try to make y as large as possible. The problem

becomes:

$$\begin{aligned}
& \text{maximize} && y \\
& \text{subject to} && y \leq \sum_{f_i \in F} \alpha(f_i) \times U(F), \\
& && \alpha(f_1) + \alpha(f_2) + \dots + \alpha(f_n) = 1, \\
& && \alpha(f_i) \geq 0, \forall f_i \in F
\end{aligned} \tag{III.7}$$

To make it easier, we assume that $y > 0$ and $x(f_i) = \frac{\alpha(f_i)}{y}$, which transforms the constraint $\alpha(f_1) + \dots + \alpha(f_n) = 1$ into $x(f_1) + \dots + x(f_n) = \frac{1}{y}$. Now, y can be eliminated by minimizing $x(f_1) + \dots + x(f_n)$ instead of maximizing y , since maximizing y is equivalent to minimizing $\frac{1}{y}$. The problem becomes:

$$\begin{aligned}
& \text{minimize} && x(f_1) + x(f_2) + \dots + x(f_n) \\
& \text{subject to} && 1 \leq \sum_{f_i \in F} x(f_i) \times U(F), \\
& && x(f_i) \geq 0, \forall f_i \in F
\end{aligned} \tag{III.8}$$

This problem can be solved using the simplex method in order to derive the optimal monitoring load probability distributions $\alpha(f_i)$ over the set of federations [9]. Our solution will still work well even if the percentage of the malicious providers was 100% because we are able to distribute the detection load to maximize the detection rate regardless of that percentage in one single federation.

3.4 Numerical Example

Consider a set of three federations f_1 , f_2 , and f_3 , functioning normally by processing the received requests. They yield 12.5\$, 7.5\$, and 10\$ respectively. Suppose that the detection rate at the current time is 0.8, and the reputation score is 1 for each of the three federations. The following matrix is obtained after computing the utility (using Eq. III.1, and III.3).

$$U = \begin{matrix} & f_1 & f_2 & f_3 \\ f_1 & \begin{pmatrix} 10 & -6 & -8 \end{pmatrix} \\ f_2 & \begin{pmatrix} -10 & 6 & -8 \end{pmatrix} \\ f_3 & \begin{pmatrix} -10 & -6 & 8 \end{pmatrix} \end{matrix}$$

A federation's utility can be calculated by multiplying the profit, the detection rate, and the reputation score together. For example, f_1 's gain for broker will be $12.5 \times 0.8 \times 1 = 10$. The malicious provider(s) in this case would lose 10 for not being able to bypass the federation's defensive strategy. The problem of finding the optimal way to distribute the monitoring load over the set of federations can be solved by using the simplex method, as the following:

Step 1: increment all the numbers inside matrix by the smallest number in it, to make sure that there are no negative numbers left. Thus, the matrix will be:

$$U' = \begin{matrix} & f_1 & f_2 & f_3 \\ f_1 & \begin{pmatrix} 20 & 4 & 2 \end{pmatrix} \\ f_2 & \begin{pmatrix} 0 & 16 & 2 \end{pmatrix} \\ f_3 & \begin{pmatrix} 0 & 4 & 18 \end{pmatrix} \end{matrix}$$

Step 2: create a table of size 3 containing the data in the matrix. Insert a column of 3 rows with value 1 to the right, and a row of 3 columns with value -1 at the bottom. Put a 0 in the remaining empty cell (i.e. right-bottom cell). Label the first 3 rows and 3 columns from x_1 to x_3 and from y_1 to y_3 respectively. Consider x_n to represent the broker's strategy for federation n , and y_n to represent the attacker's strategy for federation n .

	y_1	y_2	y_3	
x_1	20	4	2	1
x_2	0	16	2	1
x_3	0	4	18	1
	-1	-1	-1	0

Step 3: Select a positive value from a cell that (1) produces the smallest ratio among its column's cells, and (2) has a negative value on the last row of its column. A ratio can be obtained by dividing the last element of the row, by the pivot. Once selected, replace it by its reciprocal, and divide every other cell located on the row or the column of the pivot, by the pivot. And for each other remaining cell $T(i, j)$, subtract it by the product of $T(a, j)$ by $T(i, b)$, divided by the pivot, where a and b represents the row and column number of the pivot.

The pivot would be the first cell (i.e. the one with value = 20) in this case since its ratio is 0.05, which is the smallest among the others.

	y_1	y_2	y_3	
x_1	0.05	0.2	0.1	0.05
x_2	0	16	2	1
x_3	0	4	18	1
	-0.05	-0.8	-0.9	0.05

Step 4: Switch the labels of the row and column of the pivot with each other.

	x_1	y_2	y_3	
y_1	0.05	0.2	0.1	0.05
x_2	0	16	2	1
x_3	0	4	18	1
	-0.05	-0.8	-0.9	0.05

Step 5: Repeat steps 3 and 4 until no negative number exists in the last row of the table. The table will be the following after 2 more loops:

	x_1	x_2	x_3	
y_1	0.05	-0.0114	-0.0043	0.0343
y_2	0	0.0643	-0.0071	0.0571
y_3	0	-0.0143	0.0571	0.0429
	0.05	0.0386	0.0457	0.1343

To calculate the probability distribution for both of the broker and the malicious provider(s), we divide the numbers located on the borders by the intersection of the two borders (i.e. 0.1343). For example, the broker’s strategy for the first federation is $x_1 = 0.05$, to obtain the allocated monitoring load, we divide it by 0.1343, to obtain $\alpha_t(f_1) = 0.372$. Whereas the probability allocated by the attacker for the same federation is $\beta_t(f_1) = \frac{0.0343}{0.1343} = 0.255$. Table 1 presents the optimal strategy for the broker on how to distribute the monitoring load over the set of federations, and the optimal strategy for the attackers on how to distribute their misbehavior over the same set, at time t .

Table 1: Probability Distribution of the Broker and the Malicious Provider(s)

Player	f_1	f_2	f_3
Broker	0.372	0.288	0.340
Malicious Provider(s)	0.255	0.425	0.320

3.5 Experimental Evaluation

In this section, we evaluate the performance of our solution through comparing it experimentally with the sky federation model [17] since the authors use related configurations and metrics, and to examine the performance of their approach under a harsh environment wherein multiple malicious providers are involved.

3.5.1 Implementation Setup

The simulations have been conducted in a 64-bit windows 10 environment having an i7-4720 HQ CPU, 2.6 GHZ, and 16 GB of memory. We used MATLAB as a programming language to implement the two studied models. To compare our solution with the sky federation model, we used a similar setup in terms of VMs prices and configurations (i.e., small, medium, large, and extra-large). We simulated ten cloud providers and varied the percentage of malicious providers from 10% to 50% out of the ten considered providers. Malicious providers would drop some requests coming from the federations with a probability varying from 50% to 99%. The simulations

have been run for 100 iterations to maximize the accuracy of the obtained results. To study the performance of our solution w.r.t the sky federation model, five performance metrics have been evaluated, namely those of false negative, attack detection rate, overall profit, latency, and availability. False negative represents the proportion of times in which the model wasn't able to detect a passive malicious action, and which can be calculated as follows:

$$\theta_{[t_1, t_2]} = \sum_{\varepsilon=t_1}^{t_2} \sum_{f_i \in F} \frac{\beta_{\varepsilon}(f_i) \times (1 - \alpha_{\varepsilon}(f_i))}{t_2 - t_1} \quad (\text{III.9})$$

for each $\beta_{\varepsilon}(f_i) > \alpha_{\varepsilon}(f_i)$

On the other hand, attack detection rate represents the proportion of times in which the model was successful in detecting the passive malicious actions. The overall profit represents the monetary gain yielded by a certain provider as a result of joining a federation. Latency describes the delay between the submission of a user request to a certain VM and the receipt of the response. Finally, availability represents the percentage of times in which a VM was available to serve users' requests.

To populate the QoS metrics of the VMs, we used data obtained from CloudHarmony¹, which records the promised as well as the actual QoS metrics of different cloud services (measured for a period of 30 days) pertaining to different well-known providers such as Amazon and Agile Cloud. This allows us to have an idea about the behavior of the providers in terms of committing to their QoS promises.

3.5.2 Results and Discussion

We notice from the set of figures presented in this section (Fig. 5, 6, 7, 8, and 9), that when the percentage of malicious providers increases, the performance of the detection and the QoS decreases. The reason is that the more the malicious providers we have, the larger the set of virtual machines harming the federations will be. Consequently it becomes harder for the broker to distribute the same budget of detection load over a

¹<http://cloudharmony.com/>

larger set of attacking virtual machines.

In the first series of experiments, we evaluate the performance of our solution in terms of passive malicious misbehavior detection (Fig. 5). Note that for this set of experiments the sky federation model wasn't included in the comparisons since this latter model is entirely business-oriented and does not account for the malicious misbehavior when forming federations. By looking at Fig. 5, we can notice that our solution achieves high detection rates while being scalable to an increased percentage of malicious providers. For example, when the percentage of malicious providers was 10% of the total number of considered providers, the attack detection percentage was 97%. When the percentage of malicious providers jumped to 50%, the attack detection percentage remained high (i.e., 93%). Similarly, Fig. 6 reveals that our solution entails low levels of false negative and its performance is scalable to the increase in the percentage of malicious providers.

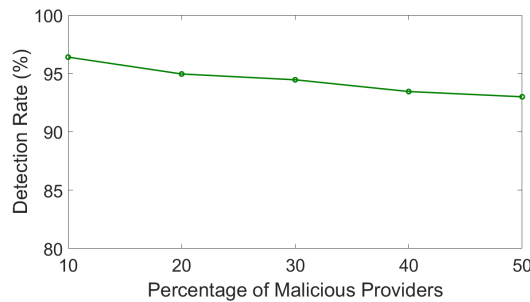


Figure 5: Detection rate

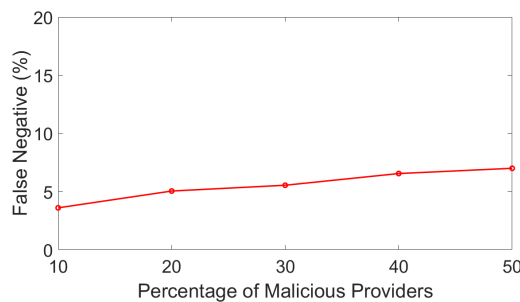


Figure 6: False negative

In Fig. 7, we measure the profit per hour yielded by providers compared to the sky federation model. To do so, we injected some malicious providers in the sky federation model and implemented their merge-and-split federation formation algorithms.

We can notice from Fig. 7 that our model can increase the profit of the providers in the presence of passive malicious providers compared to the sky federation model. The reason is that we advance a detection strategy to capture the passive malicious misbehavior whose presence leads to decreased performance and hence decreased profit for the federations. In contrary, the sky federation model is purely business-oriented and ignores the existence of malicious providers, which might lead to the generation of federations consisting of a large number of malicious members.

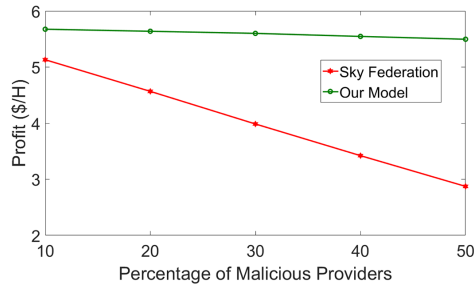


Figure 7: Profit

Thereafter, we measure the QoS (i.e., latency and availability) delivered by the formed federations while serving users' requests. Latency can be defined as the delay from the submission of the user's request to the federation to the submission of the response back to the user. Fig. 8 shows that our solution stabilizes the latency compared to the sky federation model since the malicious providers will be motivated, due to the detection mechanism, to not misbehave under the threat of being penalized. In contrary, the sky federation model gives those malicious providers the freedom to carry out their selfish misbehavior without being detected. Finally, we measure in Fig. 9, the percentage of availability for the formed federations, where it shows that our solution can improve the availability of the federations compared to the sky model, due to our detection mechanism, while being resilient to an increased percentage of malicious providers. Specifically, we can notice that the availability percentage decreased by 4% only when the percentage of malicious providers increased from 10% to 50%.

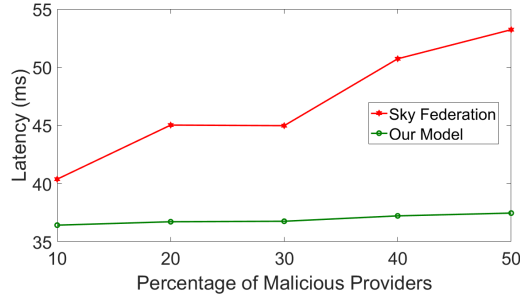


Figure 8: Latency

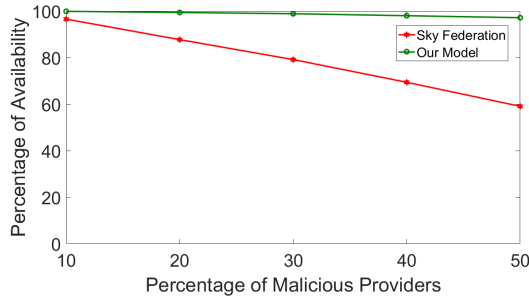


Figure 9: Availability

3.6 Conclusion

In this chapter, we addressed the problem of detecting passive malicious providers who join cloud federations to harm the performance, profit, and reputation of those federations. As a solution, we proposed a maximin game theoretical model which allows the cloud broker to maximize the detection of such malicious providers. Experiments conducted using the CloudHarmony dataset show that our solution achieves a high detection rate up to 92% and improves the profit, availability, and latency of the federations up to 25% compared to the sky federation model.

Chapter Four

Cloud Federation Formation Using Genetic and Evolutionary Game Theoretical Models

4.1 Introduction

In this chapter, we propose an approach based on genetic algorithms and evolutionary game theory in order to study the problem of forming highly profitable federated clouds, while maintaining stability among the members in the presence of dynamic strategies (i.e. cloud providers joining and/or leaving federations) that might result in decreased Quality of Service (QoS). The problem may arise after the federation formation where many cloud providers, due to the dynamicity, may be tempted to reallocate their resources into other federations for seeking better payoff. Such an act may lead to a decrease in the QoS and cause a drop in the profit earned by the federations. Thus, we extend the genetic model as an evolutionary game, which aims at improving profit while maintaining stability among federations.

Experiments were conducted using CloudHarmony real-world dataset and benchmarked with Sky federation model previously introduced in the literature. Both the genetic and evolutionary game theoretical models outperform the benchmarked one. The evolu-

tionary game model gave better results in terms of profit and QoS due to its mechanism of reaching a stable state, in which no provider has incentive to reallocate his resources into different federations.

The remainder of this Chapter is organized as follows. Section 4.2 presents and formulates the problem. Section 4.3 proposes a solution for the formation problem using a Genetic Algorithm. Section 4.4 explains the evolutionary game theoretical approach, and models the formation problem as a game. Section 4.5 provides a numerical investigation about the used models, while Section 4.6 states the experimental setup and analyses the experimental results. Finally, Section 4.7 concludes the chapter.

4.2 Stable Cloud Federation Formation Problem

In this section, we provide the details of our solution by highlighting briefly our approach, describing the environment, and stating the problem and the utility functions used in it.

4.2.1 Approach Overview

As shown in Fig. 10, the main components of the cloud federation architecture are the cloud providers, virtual machines, clients, and brokers. A cloud broker is responsible for managing the cloud providers, and ensuring that the service requested by the client is being provided [14]. In a nutshell, cloud providers collaborate by merging together their virtual machines in order to serve a client who has requested a number of VMs that no single provider could afford. Once a federation is formed, the client has to pay the broker in exchange for the service. In return, the broker shall distribute the profit among the cloud providers. What complicates the process is the fact that payments may differ from a federation to another depending on the size of the request and the duration of the service. Such a discrepancy might motivate some providers to change their strategies and reallocate their VMs in other (potentially) more profitable federations. By reallocating the resources from one federation to another, the QoS of the federation

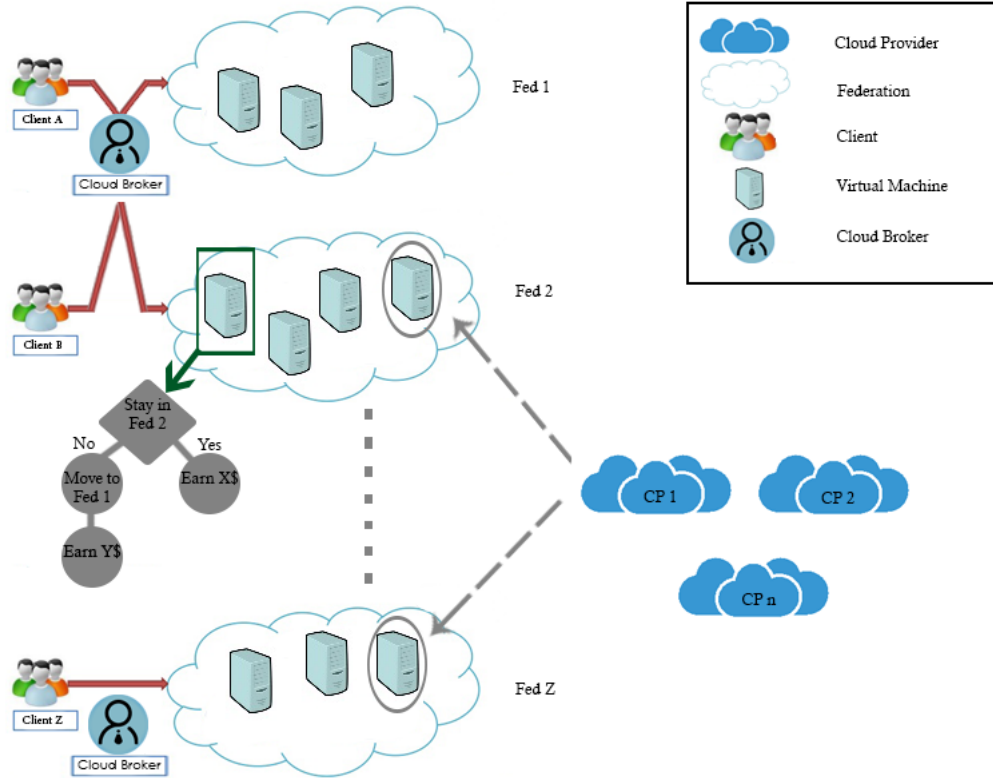


Figure 10: Cloud Federations: a VM's Dilemma

will drop, which will affect its reputation. The objective of this chapter is to derive a mechanism that maintains the QoS of the federations in such dynamic situations by addressing the stability of these federations using genetic and evolutionary game theoretical models.

4.2.2 System Model

Let $CP = \{cp_1, cp_2, \dots, cp_i\}$ denote the set of cloud providers, and let $F = \{f_1, f_2, \dots, f_n\}$ represent the set of federations that serve a set of requests R . Each federation has a number of virtual machines such that $V_{f_n} = \{v_1(f_n), v_2(f_n), \dots, v_m(f_n)\}$, each of which is provided by a particular cloud provider. In other words, a federation f_i can be represented as a new temporary provider that has VMs from different cloud providers for a period of time.

4.2.3 Problem Formulation

As stated earlier, the main purpose of our work is to find a formation that can yield the maximum profit. Such maximality will not occur if stability is not reached since the reallocation of cloud providers resources to different federations will affect the QoS in terms of availability and response times, thus leading to a decrease in the accrued payment. Formally, the payoff of the set of federations F at time t can be represented as follows:

$$U_t(F) = \sum_{f_i \in F} \delta_t(f_i) \quad (\text{IV.1})$$

where $\delta_t(f_i)$ represents the profit of federation f_i

$$\delta_t(f_i) = Rev_{f_i} \times Rep_t(f_i) - \sum_{v_m \in V_{f_i}} C_{v_m} \quad (\text{IV.2})$$

such that Rev_{f_i} is the revenue of federation f_i , C_{v_m} is the cost of virtual machine v_m , and $Rep_t(f_i)$ represents the reputation score of federation f_i at time t , derived as follows:

$$Rep_t(f_i) = \frac{\theta_{f_i} \vee \gamma_{f_i}}{\gamma_{f_i}} \times \frac{\alpha_{f_i}}{100} \quad (\text{IV.3})$$

where γ_{f_i} represents the average response time of the virtual machines allocated in federation f_i (i.e. the time a federation takes to respond to a certain request), θ_{f_i} is a static number that represents the response time promised by the federation, and α_{f_i} is the percentage of availability of the federation (i.e. the proportion of time the federations were available to serve requests).

We define the function $Pay_t(f_i)$ to be the payoff collected by a provider per one virtual machine allocated in federation f_i at time t as follows:

$$Pay_t(f_i) = \delta_t(f_i) \times \frac{1}{\eta_{f_i}} \quad (\text{IV.4})$$

where η_{f_i} is the number of VMs inside f_i .

Finally, the average payment is calculated using the following:

$$AvgPay_t = \sum_{f_i \in F} Pay_t(f_i) \times \frac{1}{\kappa} \quad (IV.5)$$

where κ is the number of federations.

Based on the above, the profit maximization problem is formulated for a given set F of federations at time t as follows:

$$maximize U_t(F) \quad (IV.6)$$

and the stable formation problem is formulated to be the following:

$$minimize \sum_{f_i \in F} (Pay_t(f_i) - AvgPay_t)^2 \quad (IV.7)$$

which implies minimizing the variability of the made payments; the less the difference in payments, the more satisfied cloud providers about their VM placements. Such satisfaction will affect their resources reallocation decision to other federations. The wider the gap in terms of payments is, the more unstable the federations would be.

In the sequel, we provide solutions for the aforementioned problem using both genetic and evolutionary game theoretical models.

4.3 Federation Formation using Genetic Algorithm

Genetic algorithm is a metaheuristic approach used to efficiently solve complex optimization problems. Typically, GAs attempt to find a near optimal solution in a relatively short time. The main components of GA are (1) encoding scheme, (2) fitness evaluation, and (3) evolution process.

4.3.1 Encoding Scheme

A chromosome is an encoding of a candidate solution. GAs start usually from a set of chromosomes that are generated randomly or using a particular heuristic. They form the initial population. Typically, encoding is problem specific and vary from one problem to the other. In our problem, we are interested in figuring out where each virtual machine should be allocated. Therefore, we use a permutation-based encoding and we model the formation problem as an ordering problem, where VMs are assigned unique numbers from 1 to n . Different VMs that are assigned to different federations are separated by 0's. An example of a candidate solution can be '0 2 4 6 0 1 3 5 0'; this chromosome implies that there are two federations formed since 3 zeroes exist, where the first federation has VMs number 2, 4, and 6, and the other federation has VMs number 1, 3, and 5. For the initial population, instead of generating a set of random chromosomes, we use a heuristic technique so that the GA can start from a good point. The heuristic would then aim at assigning cloud providers to the federations that give them the highest profit.

The aforementioned heuristic is presented in Algorithm 1, which takes as arguments the required population size ($pSize$), the probability of a cloud provider being careless to whatever federation he joins ($pRatio$), the number of requests arrived simultaneously ($numberOfReq$), the set of prices the clients are willing to pay ($prices$), and finally the set of virtual machines that are being allocated for the forthcoming federations (vms). It outputs a set of chromosomes ($pArray$), in which, each individual might be a candidate solution. The algorithm starts by calculating the total payoff (Line 2 to Line 5) since it will be needed later on during the decision making phase. It will loop $pSize$ times to produce $pSize$ different individuals (ind). In each chromosome, and for all virtual machines, a random decision will be taken (Lines 9 to Line 12) based on the variable $pRatio$ to check whether that virtual machine will be assigned randomly to a federation, or not. If not, the virtual machine will follow a federation according to the profit it can yield (Line 14 to Line 24).

Algorithm 1 Initial Population Generator

Input: $pSize$, $pRatio$, $numberOfReq$, $prices[]$, $vms[]$

Output: $pArray$

```
1:  $pArray =$   
2:  $totalPayoff = 0$   
3: for all  $price \in prices$  do  
4:    $totalPayoff = totalPayoff + price$   
5: end for  
6: for  $i = 0$  to  $pSize$  do  
7:    $ind = array[numberOfReq]$   
8:   for all  $vm \in vms$  do  
9:      $r = random(0, 1)$   
10:    if  $pRatio \leq r$  then  
11:       $randFed = random(0, numberOfReq)$   
12:       $ind[randFed] = ind[randFed] \cup vm$   
13:    else  
14:       $choice = random(0, totalPayoff)$   
15:       $temp = 0, counter = 0$   
16:      for all  $price \in prices$  do  
17:        if  $choice \leq price + temp$  then  
18:           $ind[counter] = ind[counter] \cup vm$   
19:          break  
20:        else  
21:           $temp = temp + price$   
22:           $counter = counter + 1$   
23:        end if  
24:      end for  
25:    end if  
26:  end for  
27:   $pArray = pArray \cup ind$   
28: end for  
29: return  $pArray$ 
```

4.3.2 Fitness Evaluation

The evaluation step requires a fitness function $f(c)$ which can be used in order to evaluate a candidate solution c . The fitness function returns a score for the candidate solution which represents the federations' payoff and the variance of the payments.

4.3.3 Evolution Process

The evolution process consists of 3 steps, which are (1) selection, (2) crossover, and (3) mutation. A set of solutions is produced in each generation, but not all of them are worth reproducing. The selection phase consists of selecting only the few chromosomes that are likely to be fruitful. The selected portion of the candidates will breed a new generation. Our selection mechanism consists of selecting the best half of the population and apply the genetic operators (i.e. crossover and mutation) on them to produce new candidates for the following generation.

A chromosome consists of a set of genes, and in our problem, a gene is either an identifier of a virtual machine, or a 0 that refers to the start/end of a new federation. The crossover can be achieved by exchanging genes between two selected chromosomes as an attempt to breed a better offspring. For each two chromosomes, we generate a random number r between 1 and the length of 1 individual. We split the two chromosomes at position r in order to obtain 2 subsequences of genes. Then we append the right-sided subsequence of the second chromosome to the left-sided subsequence of the first one to obtain the first offspring, and repeat the process in reverse to obtain the other offspring. For example, if the first chromosome is '123456' and the second is '321564', after crossing at position 3, the offsprings A and B will be '123**564**' and '321**456**' respectively.

Mutation can be applied by altering randomly one or more values of the offspring to change the latter into a totally different solution that could be better or worse. In our implementation, we swap 2 genes together to result in a modified individual. For

instance, offspring A can have the third and the fourth values switched to be ‘125364’.

Algorithm 2 Genetic Algorithm Pseudocode

```
1:  $t \leftarrow 0$ 
2:  $initialize\_population(P)$ 
3:  $evaluate\_fitness(P)$ 
4:  $order\_candidates(P)$ 
5: while termination condition not met do
6:   for all  $\{i, j\} \in P$  do
7:      $\{i', j'\} \leftarrow crossover(i, j)$ 
8:      $mutate(i', j')$ 
9:      $P \leftarrow \{P, i', j'\}$ 
10:  end for
11:  $evaluate\_fitness(P)$ 
12:  $order\_candidates(P)$ 
13:  $remove\_worse\_half(P)$ 
14:  $t \leftarrow t + 1$ 
15: end while
16: return  $P[0]$ 
```

Algorithm 2 depicts the formation mechanism using genetic algorithm. At Line 2, the initial population is initialized as shown in Algorithm 1. At Line 3, each candidate solution inside P is evaluated based on the predefined fitness function (the two objective functions in Eq. IV.6 and IV.7). Then, the chromosomes are sorted from best to worst based on their fitness. The loop at Line 5 will only terminate when a satisfying solution is reached. The selection process at Line 6 requires selecting each two candidates together (with proportion to their fitness), and applying genetic operators on them (i.e. crossover and mutation at lines 7 and 8). At Lines 11, the new population is evaluated by the same fitness function that was used at Line 3. Then the population is sorted again based on the fitness. At Line 13 the worst $n/2$ candidates are eliminated from the list, where n represents the total number of candidates in the population. This step is done in order to reduce the storage utilization of the algorithm. When the stopping condition is met, the algorithm will return the first element (i.e. candidate) of the list, which is the fittest so far (Line 16).

4.4 Federation Formation using Evolutionary Game Theory

Although GA performs well on improving the profit, as will be shown in Section 4.6, however, it is not able to form stable coalitions which will affect the profit and reputation on the long run. In addition, its convergence time is too long due to the fact that the solutions it provides are based on finding a better candidate from one iteration to another. Thus, to address the aforementioned problem, we propose in this section an evolutionary game theoretical model, in which we treat the cloud formation and VMs reallocation as a game and solve it to overcome the dynamism problem.

4.4.1 Player Strategy

As a rational decision maker, a cloud provider always seeks for gaining more profit. Some federations may provide more profit than others. Such federations are more likely to have more contributors than others. Therefore, the set of federations would become unstable every time a cloud provider decides to reallocate one or more of its VMs to another federation, hence causing a decrease in the outcome and in the reputation for the whole set. The idea is to have a strongly built set of federations such that no one has incentive to deviate from its current federation. Suppose that we have two sets of virtual machines V_{f_i} and V_{f_j} allocated into two federations f_i and f_j . These federations guarantee the amounts X and Y of profit for providers respectively for each VM contributed to the federation such that X is higher than Y . Being rational, the owners of the virtual machines dedicated for f_j (i.e. $v_1(f_j), v_2(f_j), \dots, v_m(f_j)$) are more likely to start thinking whether they should reallocate their VMs into f_i due to the extra profit they can obtain by switching into the latter. It is worth mentioning that having extra virtual machines joining f_i will cause the profit given to each single player to be less than X due to the increase of players while yielding the same amount of profit obtained by the federation f_i . This can happen if the client was already satisfied with what he had (i.e. client needed a certain number of VMs, but got allocated more than

what he needs). Also, f_j might not yield the same profit (i.e. Y) if the resources remaining in this federation do not meet the requirements set by the client. Therefore, the total profit of the whole set of federations will be reduced.

4.4.2 The Evolutionary Formation Game

We study the evolutionary behavior among the cloud providers who will decide to which federation they are going to allocate their resources. In evolutionary game theory, a player is more worried about the payoff that comes from his current federation. Therefore, we study the problem of how to allocate the virtual machines owned by the cloud providers into the set of available federations while seeking the highest possible payoff that can be acquired. If a federation contains a large number of VMs in such a way that the supplied resources are more than what was requested, then the profit of a single VM (Section 3, equation IV.4) will decrease because of the increase in terms of cost. Many rational players, as a consequence, would change their strategy seeking for a better payoff by joining another federation. Changing the federation of a certain virtual machine may repeat many times until the provider is assured that he is getting in return the best possible outcome. In other words, federations are chosen based on which one can guarantee the maximum profit possible for each VM. Since it is hard to reach immediately an optimal decision because of the huge number of virtual machines that would be allocated in the federations, we apply the evolutionary game framework in order to analyse such interactions. The main components of such a game are the following: (1) players, (2) population, (3) strategy, and (4) utility. The players in this game are the virtual machines, controlled by the cloud providers. The population is the set of all allocated virtual machines. The set of strategies available are the federations which a virtual machine can be assigned to. The utility of a player is the payoff (i.e. what he is getting on behalf of what he is offering). Whenever a cloud provider expects to find a better payoff for a virtual machine, he would change his strategy by joining the federation offering more payoff; therefore, a successful strategy is more likely to be replicated. The notion of evolution in this evolutionary game theory can be

represented by that replication. To describe this evolution with time, we use a model called replicator dynamics.

Let $x = \{x_1, x_2, \dots, x_n\}$ represent the vector of distribution of strategies in the population, where n is the number of available federations, such that

$$\sum_{i=1}^n x_i = 1 \quad (\text{IV.8})$$

The general form of the replicator equation is as follows

$$\dot{x}_i = x_i[f_i(x) - \bar{x}] \quad (\text{IV.9})$$

where $f_i(x)$ is the fitness function of selecting a strategy i (i.e. the i^{th} federation), which is equation (IV.4), and \bar{x} is the average fitness by the population. This average fitness can be calculated by multiplying each strategy fitness by its proportion in the whole population, as described in equation (IV.10).

$$\bar{x} = \sum_{j=1}^n x_j f_j(x) \quad (\text{IV.10})$$

Equation (IV.9) shows that the proportion of the population for choosing a successful strategy would increase with time. When the fitness of every strategy becomes equal, the proportion will not change anymore. This strategy would become the Evolutionary Stable Strategy (ESS), since no VM can find a better payoff in any other federation. Mathematically, the key is to solve $\dot{x}_i = 0$ for all strategies in order to reach equilibrium. Algorithm 3 shows how the game works. First, the initialization of a random solution should take place, where all VMs are assigned into the set of federations (Line 1). We decided to use the same heuristic used in populating the GA, such that we pick the best chromosome from the set generated out of Algorithm 1 since the evolutionary game theory needs to start from only one solution and not from many like the GA. The game actually starts at Line 3, where we dive into an infinite loop. Lines 4 – 6 state that the payments issued by the federations should be calculated in order to calculate

Algorithm 3 Evolutionary Game Theory Pseudocode

```
1: initialize federations
2:  $t = 0$ 
3: while true do
4:   for all  $f_i \in F$  do
5:     calculate  $Pay_t(f_i)$ 
6:   end for
7:   calculate the average utility ( $x$ )
8:   for all  $vm_j \in VMs$  do
9:     if  $(x) > f_{strat(vm_j)}(x)$  then
10:      change strategy with probability p
11:    end if
12:  end for
13:   $ESS\_Reached = true$ 
14:  for all  $x_i \in x$  do
15:    if  $\dot{x}_i \neq 0$  then
16:       $ESS\_Reached = false$ 
17:    end if
18:  end for
19:  if  $ESS\_Reached == true$  then
20:    break
21:  end if
22:   $t = t + 1$ 
23: end while
```

the average utility (x) at line 7. Decisions are made at lines 8-12, $strat(vm_j)$ is the strategy that the cloud provider has chosen for vm_j . Every single virtual machine can compare its own payoff with the average utility (x) to check whether it is getting paid above or below average, and based on that comparison, it may decide to switch into another federation that can provide it with better profit. The switching part is based on a probability p , which is relative to how much the current payoff is far from the best strategy. At lines 13-21, we check if a stable set of federations got established or not by calculating \dot{x}_i for all available federations. ESS will be established when \dot{x}_i is 0, for all $x_i \in x$.

Theorem 1 *Algorithm 3 leads to a stable set of federations.*

Proof 1 *The algorithm solves $\dot{x}_i = 0$ for all $x_i \in x$. Therefore, the evolutionary equilibrium can be obtained. There are no incentives to reallocate any of the virtual machines into different federations in the evolutionary equilibrium since the rate of selecting a strategy i is zero.*

4.5 Numerical Investigation

In this section, we investigate numerically the process of forming the cloud federations using our two proposed models, i.e. genetic algorithm and evolutionary game theory. Consider six cloud providers, where each one of them has only two available VMs that are not rented out yet. For simplicity, the VMs have the same specifications in terms of processors (2 cores each), and they all have good reputations. Three clients request computing resources, such that the requirements are 6 cores, 8 cores, and 10 cores, and they are willing to pay 12\$, 16\$, and 20\$ respectively (Table 1). No cloud provider can

Table 1: Requests

Request #	Required Cores	Price (\$ per day)
1	6	12
2	8	16
3	10	20

serve any of these requests alone, therefore, an optimal and stable formation should exist to satisfy both parties (i.e. the providers and the clients). For simplicity, we assume that the reputation score $Rep_t(f_i)$ for all federations is 1. An initial solution is generated randomly as a starting point for both models (Table 2), where the first and the third federations have each five VMs contributing within, and the second one has two VMs only. The actual profit that the federation is going to make will be less than what was expected if the resources do not meet the requirements. We consider the profit to be a percentage of what is given. That is, if the client gets half of what he requested, he pays only 50% of the amount of money agreed on. However, if he gets served by more resources than what he actually needs, he only pays the promised amount. The expected profit differs from a federation to another, which will make the

Table 2: Initial solution

Fed. ID	# of VMs	Exp. Profit/VM	Act. Profit/VM
1	5	2.4	2.4
2	2	8	4
3	5	4	4

federations unstable. To calculate the expected profit per virtual machine of a certain request, we should divide the price by the number of virtual machines assigned to serve that request. Whereas the actual profit per virtual machine in a federation is the actual payoff of a virtual machine, which may differ from the expected, as previously discussed. The total payoff of the initial solution is the sum of the actual profit per VM times the number of contributing VMs per federation, that is $5 * 2.4 + 2 * 4 + 5 * 4 = 40$.

4.5.1 Solution Using Genetic Algorithm

We try to stabilize the federations by finding the optimal formation using Algorithm 2. After the first generation, GA seeks a better formation as a solution for the second generation by finding a way to increase the payoff (Section 3, Equation IV.1). Due to the randomness, a possible solution produced by the GA after several generations could be as shown in Table 3. The number of virtual machines allocated for the first and last request decreased by 1 each, thus reaching 4. The second federation got an

Table 3: Solution after several generations

Fed. ID	# of VMs	Exp. Profit/VM	Act. Profit/VM
1	4	3	3
2	4	4	4
3	4	5	4

increase of 2 extra virtual machines coming from the other 2 federations, to make the current number of VMs 4. The total payoff is now $4 * 3 + 4 * 4 + 4 * 4 = 44$. The solution is acceptable since it provides a better payoff than all the previous solutions.

4.5.2 Solution Using Evolutionary Game Theory

Evolutionary game theory focuses more on the profit of the players when switching from a strategy to another. As Algorithm 2 implies, a cloud provider may reassign his VMs based on a probability p if a different federation promised more profit. Mathematically, the fitness of the three strategies are 2.4, 8, and 4. The average fitness is $2.4 \times \frac{5}{12} + 8 \times \frac{2}{12} + 4 \times \frac{5}{12} = 4$. The cloud providers allocating their VMs in the second federation are expected to remain satisfied since it is currently the best possible strategy. However, the members of the first federation are more likely to switch their strategy and join other federations with probability p , where p is calculated based on the gap between both of the player and average fitness; $p = (4 - 2.4)/4 = 0.4$. That is, 40% of the VMs allocated in the first federation will switch to a more profitable one. The final solution after a few generations would be in Table 4, where 3 VMs are

Table 4: Solution using evolutionary game theory

Fed. ID	# of VMs	Exp. Profit/VM	Act. Profit/VM
1	3	4	4
2	4	4	4
3	5	4	4

allocated to the first request, 4 VMs to the second, and 5 to that last. In this solution, no cloud provider has incentive to switch his VMs into another federation since all federations are providing the same payoff. All payoffs are equal to the average, which is 4. This strategy is considered to be evolutionary stable.

4.6 Experimental Evaluation

In this section, we investigate the performance of our two proposed models compared to the Sky federation model proposed in [17].

4.6.1 Experimental Setup

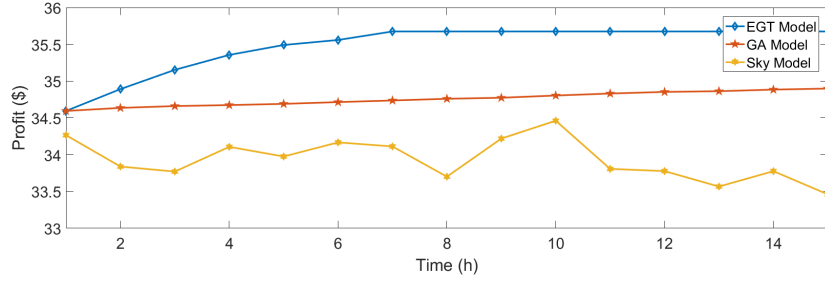
We implemented our models using Matlab 9.0 in a 64-bit windows 10 environment on a machine equipped with Intel Core i7-4720 HQ having 2.60 GHz as base frequency, 3.60 GHz as max turbo frequency, and 16 GB RAM. We set up an environment with 8 cloud providers having different numbers of available virtual machines varying from 100 to 300. At any time, a number of clients can request a specific number of CPU cores. Requests have been categorized as small, medium, and large [17]. Small requests consist of 3 federations with less than 240 requested VMs distributed among them. Medium requests are of size 5 with less than 400 requested VMs. Large requests consist of 7 federations with a total requested VMs less than 560 units. Table 5 describes the requests. The challenge is to formulate a stable set of federations for the

Table 5: Requests types

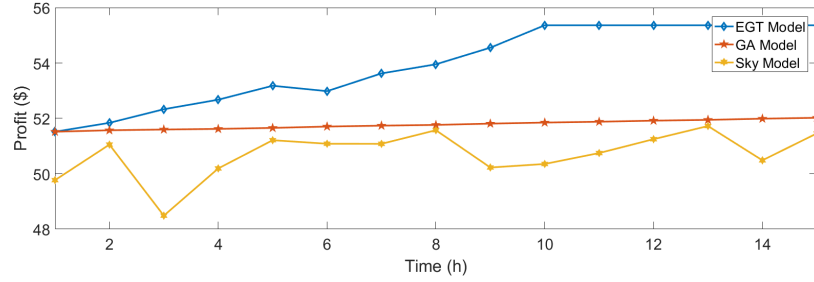
type	number of federations	number of VMs
small	3	240
medium	5	400
large	7	560

received requests. We compared our two models with the Sky model in [17] in terms of total profit, utility, response time, and availability. The total profit figures are generated based on all requests types, whereas we considered only the medium type for the rest of the figures (i.e. utility, response time, and availability) due to their similarity. We ran the algorithms of the two models 100 times and took the average results. We populated the QoS metrics by importing data from CloudHarmony dataset¹ in terms of response time and availability, knowing that CloudHarmony records QoS data for well-known cloud services, such as Amazon Web Service and Rackspace.

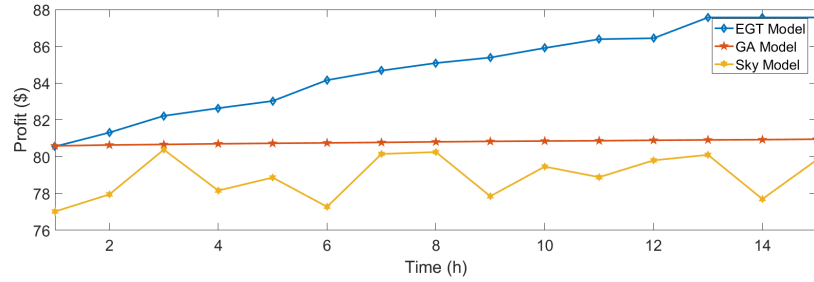
¹<http://cloudharmony.com/>



(a) small requests



(b) medium requests



(c) large requests

Figure 11: Total profit

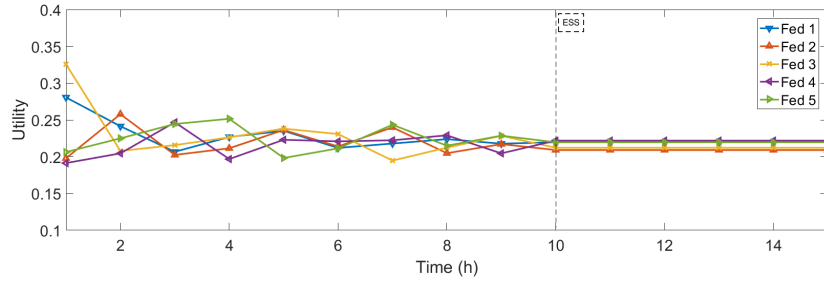
4.6.2 Results and Discussion

The first set of experiments aims to study the total profit obtained by the set of federations (Fig. 11). In Fig. 11a, 11b, and 11c, we compare the three models in terms of profit while forming federations based on small, medium, and large requests respectively. The x-axis represents the time in hours, which in each, a payment by client is made. The y-axis represents the profit amount in dollars. We notice from these figures that by applying evolutionary game theory, the federations maximize the profit in a very short period. For small requests, the total profit increased from 34.6\$ to 35.7\$ in 7 hours (i.e. 7 generations). It started as 51.5\$ for medium requests and reached 55.5\$ in 10 hours. For large requests, the profit increased from 80.5\$ to 87.5\$ in 13 hours. The more the number of federations increases the longer the proposed evolutionary

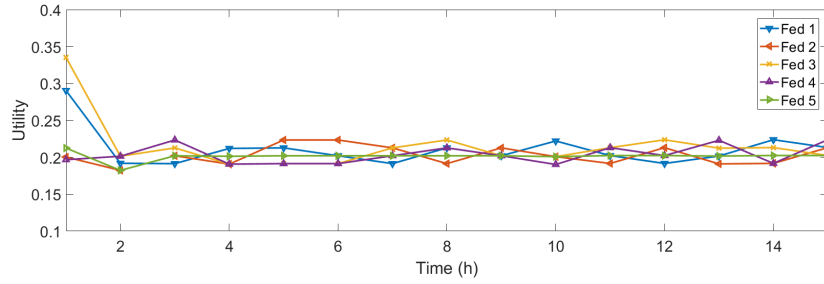
game takes to reach maximality since the unsatisfied cloud providers will most likely be having more strategies to switch to. Genetic algorithm starts from the same point as the EGT (34.6\$, 55.5\$, and 87.5\$ for small, medium, and large requests respectively) as they both share the same heuristic mentioned in Algorithm 1. In all three figures, GA seems to always increase the total profit since it is based on the idea of keeping the current federation structure unless a new structure that provides a better profit is found. However, because of the lack of any smart decision making like the evolutionary game theory, the improvement pace brought by GA is quite slow compared to our other model. In fact, it couldn't maximize profit in the first 15 generations for the three types of requests, and was always steps behind our evolutionary game. It could only reach 34.9\$, 52\$, and 81\$ for the three types of requests respectively. The performance of the sky model was the poorest among the three models because their algorithm consists of only forming the federations from the first hour, and neglecting the fact that the virtual machines will change federation with time, which makes it easier for a cloud provider to leave and join other federation in a very random way, hence leading to an inconsistent state in the total profit chart; i.e. always increasing and decreasing.

We compare the utility of a VM provider in the federations in our second set of experiments (Fig. 12). The x and y axes represent the time and utility a VM provider located in a certain federation gets. Five federations exist since we considered the medium request size, each of which is colored differently. In Fig. 12a, the first few generations (1 to 9) formed by the EGT model were chaotic due to the changes applied to the payoff of each virtual machine from a generation to another. The utilities vary from 0.2 to 0.335. Stability will take over when all utilities are the same, and no provider has incentive to switch his virtual machine to another federation. Things will stabilize at time 10 when all utilities will almost be equal to 0.22. By then ESS will be reached. However, none of the other two models (i.e. GA and Sky in Fig. 12b and 12c) were able to reach stability (at least not during the first 15 generations). Such an act may lead to a drop in terms of reputation and QoS.

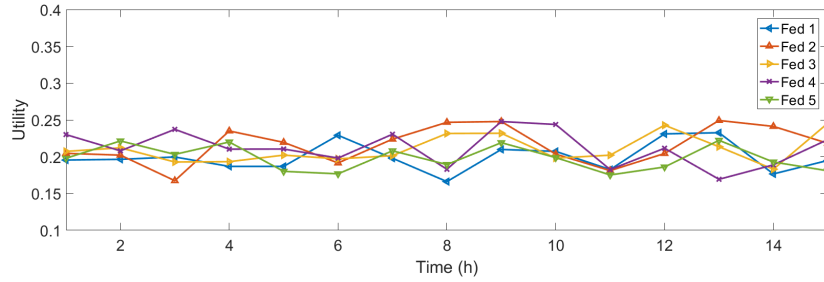
The last set of experiments consists of the QoS of the formed federations (Fig. 13).



(a) using EGT model



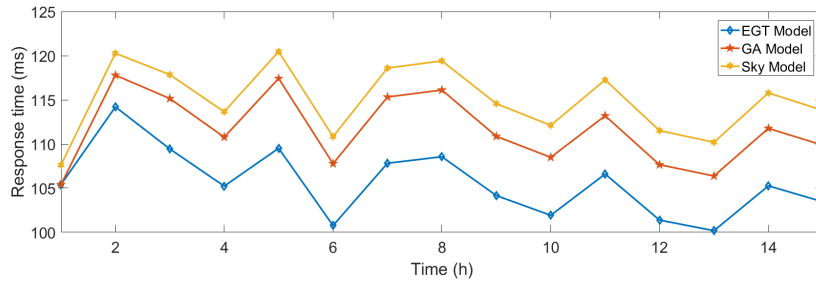
(b) using GA model



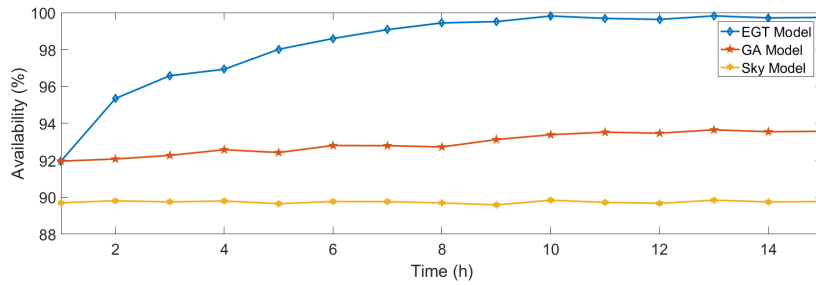
(c) using Sky model

Figure 12: VM utility

In Fig. 13b and 13a, we evaluate the availability and response time of the federations respectively using the three models. The availability represents the percentage of times the VMs were able to execute their assigned tasks normally without being unavailable due to switching federations. The response time is the time a federation takes to respond to a request. Due to the lack of continuous algorithm of the sky model, the availability couldn't get better than 89%, whereas the GA and EGT models started at 92%, and kept increasing in terms of availability. When the EGT model reaches ESS (i.e. time = 10), the federations become almost 100% available. Also, the EGT model can reduce the response time due to its stable formation, whereas the sky model has no learning mechanism; therefore, the response time would remain high. The GA model keeps on improving slowly in terms of availability and response time as time evolves.



(a) response time



(b) availability

Figure 13: Quality of service

4.7 Conclusion

In this chapter, we presented two new models to form cloud federations using genetic algorithm and evolutionary game theory. The genetic algorithm works on enhancing the total payoff of the federations from a generation to another by exploring the search space and finding a better cloud formation. The evolutionary game theory is somehow different in the sense that it doesn't take into account whether the next generation is better or worse. The level of satisfaction of each cloud provider is the key factor for the strategy selection. Therefore, it works on reducing the difference between utilities/profits among providers in order to reach the evolutionary stable strategy. The results showed that our models yield greater profit from one generation to another until reaching maximality. We improved the profit by up to 10% compared to the Sky model [17]. In addition, the models can enhance the QoS of the currently formed federations, such as the availability and response time, which makes these federations more appealing for clients on the long run. The experiments also revealed that the evolutionary game theoretical model outperformed the genetic algorithm in terms of profit and QoS due to the evolutionary stable strategy. In this context, no provider has

incentive of reassigning any of his virtual machines, which led to raise the availability of the VMs to reach 100% and reduce the response time by almost 10%.

Chapter Five

Conclusion

Cloud Federation is an architecture which consists of combining unused resources of cloud providers in order to serve a client that couldn't have been served otherwise. In this thesis, we presented defensive mechanisms for the problems threatening to worsen the promised Quality of Service by cloud federations. The first problem consists of having passive malicious cloud providers in the formed federations. They can harm the federations from the inside by reneging on their promises to dedicate their VMs to their own requests. We advanced a maxmin game theoretical model between the broker, a third party, and the malicious providers, in order to maximize their detection, and reduce the losses. We were able to achieve a high detection rate, hence maintain the profit and reputation of the guarded federations.

The second tackled problem consists of having unstable federation formation, where the cloud providers keep on reallocating their VMs from a federation to another. Such dilemma may reduce the availability of the machines and increase the response time. We proposed two methods for the formation of a highly profitable set of federations, using genetic and evolutionary game theoretical models. Both models increase the total profit earned for the formed federations. However, the evolutionary model was able to stabilize the federations, hence improve the Quality of Service provided.

Fog computing extends the concept of cloud computing to the network edge, making it ideal for IoT and other applications that require real-time interactions. For future

work, we would like to study the challenges that tackle the fog computing structure, leaving it vulnerable to internal and/or external attacks.

List of Publications

The following is the list of publications derived from the thesis work:

Conference Paper

- On the Detection of Passive Malicious Providers in Cloud Federations. IEEE Communications Letters, 23(1), 64-67.

Journal

- Cloud Federation Formation Using Genetic and Evolutionary Game Theoretical Models. Future generation computer systems. *Submitted.*

Bibliography

- [1] Ahmed A Alabdel Abass, Liang Xiao, Narayan B Mandayam, and Zoran Gajic. Evolutionary game theoretic analysis of advanced persistent threats against cloud storage. *IEEE Access*, 5:8482–8491, 2017.
- [2] Gaetano F Anastasi, Emanuele Carlini, Massimo Coppola, and Patrizio Dazzi. Qos-aware genetic cloud brokering. *Future Generation Computer Systems*, 75:1–13, 2017.
- [3] Sazzad Hossain Bhuiyan and Md Mahmudul Hasan. Revenue maximization in cloud federation based on multi-choice multidimensional knapsack problem. In *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2018.
- [4] Rajkumar Buyya, James Broberg, and Andrzej M Goscinski. *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons, 2010.
- [5] Jiah-Shing Chen, Jia-Li Hou, and Shih-Min Wu. Building investment strategy portfolios by combination genetic algorithms. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 3, pages 776–780. IEEE, 2007.
- [6] Anand Dhole, Manoj V Thomas, and K Chandrasekaran. An efficient trust-based game-theoretic approach for cloud federation formation. In *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 1–6. IEEE, 2016.

- [7] David Easley, Jon Kleinberg, et al. Networks, crowds, and markets: Reasoning about a highly connected world. *Significance*, 9:43–44, 2012.
- [8] Bob Evans. Top cloud vendors will crush \$100 billion in 2018 revenue; microsoft, amazon, ibm hit \$75 billion?, May 2018. URL: <https://www.forbes.com/sites/bobevans1/2018/05/21/top-cloud-vendors-will-crush-100-billion-in-2018-revenue-microsoft-amazon/#3aaed51c7548>.
- [9] T. S. Ferguson. Game theory. Technical report, Los Angeles, CA, USA: UCLA, Mathematics Department, 2008.
- [10] Drew Fudenberg and Jean Tirole. Perfect bayesian equilibrium and sequential equilibrium. *journal of Economic Theory*, 53(2):236–260, 1991.
- [11] Inigo Goiri, Jordi Guitart, and Jordi Torres. Characterizing cloud federation for enhancing providers’ profit. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 123–130. IEEE, 2010.
- [12] Talal Halabi and Martine Bellaiche. Towards security-based formation of cloud federations: A game theoretical approach. *IEEE Transactions on Cloud Computing*, 2018.
- [13] John C Harsanyi. Oddness of the number of equilibrium points: a new proof. *International Journal of Game Theory*, 2(1):235–250, 1973.
- [14] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. Nist cloud computing standards roadmap. *NIST Special Publication*, 35:6–11, 2011.
- [15] Chonho Lee, Junichi Suzuki, Athanasios Vasilakos, Yuji Yamamoto, and Katsuya Oba. An evolutionary game theoretic approach to adaptive and stable application deployment in clouds. In *Proceedings of the 2nd workshop on Bio-inspired algorithms for distributed systems*, pages 29–38. ACM, 2010.

- [16] Sushil J Louis and Chris Miles. Playing to learn: Case-injected genetic algorithms for learning to play computer games. *IEEE Transactions on Evolutionary Computation*, 9(6):669–681, 2005.
- [17] Lena Mashayekhy, Mahyar Movahed Nejad, and Daniel Grosu. Cloud federations in the sky: Formation game and mechanism. *IEEE Transactions on Cloud Computing*, 3(1):14–27, 2015.
- [18] Networkers. Sky-high market growth driving demand for cloud infrastructure specialists, May 2018. URL: <https://www.networkerstechnology.com/growth-cloud-demand-infrastructure-specialists>.
- [19] Jonathan Newton. Evolutionary game theory: a renaissance. *Games*, 9(2):31, 2018.
- [20] Jordan Novet. Amazon cloud revenue jumps 45 percent in fourth quarter, Feb 2018. URL: <https://www.cnbc.com/2018/02/01/aws-earnings-q4-2017.html>.
- [21] Peter C Ordeshook. *Game theory and political theory: An introduction*. Cambridge University Press, 1986.
- [22] Sanguthevar Rajasekaran and GA Vijayalakshmi Pai. *Neural networks, fuzzy logic and genetic algorithm: synthesis and applications (with cd)*. PHI Learning Pvt. Ltd., 2003.
- [23] Salma Rebai, Makhoul Hadji, and Djamal Zeglache. Improving profit through cloud federation. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 732–739. IEEE, 2015.
- [24] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

- [25] Javad Sadeghi, Saeid Sadeghi, and Seyed Taghi Akhavan Niaki. Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: an improved particle swarm optimization algorithm. *Information Sciences*, 272:126–144, 2014.
- [26] J Maynard Smith and George R Price. The logic of animal conflict. *Nature*, 246(5427):15, 1973.
- [27] Adel Nadjaran Toosi, Rodrigo N Calheiros, Ruppa K Thulasiram, and Rajkumar Buyya. Resource provisioning policies to increase iaas provider’s profit in a federated cloud environment. In *2011 IEEE International Conference on High Performance Computing and Communications*, pages 279–287. IEEE, 2011.
- [28] Luis S Vargas and Guillermo A Jiménez-Estévez. Genetic algorithms for the capacitor placement problem in distribution networks. In *2011 16th International Conference on Intelligent System Applications to Power Systems*, pages 1–5. IEEE, 2011.
- [29] CK Vijayakumari, Dileep Lukose, P Mythili, and Rekha K James. An improved design of combinational digital circuits with multiplexers using genetic algorithm. In *2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy*, pages 1–5. IEEE, 2013.
- [30] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrouk, and Azzam Mourad. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Transactions on Services Computing*, 11(1):184–201, 2018.