Future Generation Computer Systems 104 (2020) 92–104

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Cloud federation formation using genetic and evolutionary game theoretical models

Ahmad Hammoud [a], Azzam Mourad [a,*], Hadi Otrok [b], Omar Abdel Wahab [c], Haidar Harmanani [a]

[a] *Department of Computer science and Mathematics, Lebanese American University, Lebanon*
[b] *Center for Cyber–Physical Systems (C2PS), Department of EECS, Khalifa University, Abu Dhabi, United Arab Emirates*
[c] *Department of Computer Science and Engineering, Université du Québec en Outaouais, Gatineau, Canada*

ARTICLE INFO

Article history:
Received 14 March 2019
Received in revised form 9 July 2019
Accepted 15 October 2019
Available online 21 October 2019

Keywords:
Cloud federation
Evolutionary game theory
Genetic algorithm
Stability

ABSTRACT

This paper proposes 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). Cloud federation helps cloud providers to take advantage of the available unused virtual machines. It allows the providers to combine their resources in order to serve a larger pool of requests that could not have been served otherwise. We tackle the problem of forming federations while maximizing the total profit they yield using a Genetic Algorithm. However, the main problem may rise after the federation formation where many cloud providers, due to the dynamicity, may be tempted to reallocated 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 to improve the 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 evolutionary game model gave better results in terms of profit and QoS's due to its mechanism of reaching a stable state, in which no provider has incentive to reallocate his resources into different federations.

1. Introduction

Cloud computing is the practice of delivering computing resources through the internet. The demand on cloud computing resources has increased in the past few years due to the various advantages it provides [1] 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 Quality of Service (QoS). A cloud provider is responsible for providing services to the cloud consumers such as infrastructure, computing resources and storage [2]. They offer Infrastructure as a Service (IaaS); a model in which computing resources, such as servers and storages, are made available to the clients. This model, alongside with the Platform as a Service (PaaS) and Software as a Service (SaaS), represent the three main cloud computing service categories offered by cloud providers. One of the fundamentals of cloud computing is virtualization which allows more than one virtual machine (VM) to be hosted on the same physical machine. Virtualization allows the cloud providers to rent out VMs that provide clients with the functionality needed to execute multiple operating systems. These VMs can be substitutes of real machines. 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's [3,4].

The increasing demands and expansion of online businesses resulted with some requests which are big enough so that they cannot be served by one single cloud provider. For instance, a client might be requesting 50 VMs, while there are three cloud providers available in the market, and each of which has only 20 VMs available. In such a case, none of these three providers can fulfill the client's request and thus dropping that request, and losing a potential client. This problem has lead to a new business architecture, *cloud federation*, that consists of merging the resources of two or more providers together in order to increase

* Corresponding author.
E-mail address: azzam.mourad@lau.edu.lb (A. Mourad).

https://doi.org/10.1016/j.future.2019.10.008
0167-739X/© 2019 Elsevier B.V. All rights reserved.

their capacity of handling large requests [5]. Such an architecture is beneficial for both parties, i.e., the client who is in need for computing resources, and the cloud provider having additional resources. Furthermore, it improves the QoS of the clients' requests due the interoperability among providers, and allows the latter to rent out their unused resources. Thus, providers can increase their profits, and expand their geographical footprints without the need of new points of presence.

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 real challenge in cloud federations is finding an optimal way to form a stable federation, while maximizing profit. Several approaches [6–8] have been proposed, however, most of the reported work in the literature did not take into consideration the stability. Whereas the work claimed achieving stability, had restrictions on the participants that prevent some cloud providers from participating in the federations [9]. Other papers ignored the fact that providers might reassign their resources to federations formed based on another set of requests [10].

The objective of this work is to achieve optimality in profit and stability among federations, which will lead to a better reputation and to an increase in the profit. In this context, we first 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 had a good performance as will be shown in the simulation section, it did 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 [11–14]. 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 contribution of this paper is two-fold:

- Maximizing the profit achieved by the cloud federations using a Genetic Algorithm. 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 and prevent the loss of profit and reputation. By reaching evolutionary stable strategy, the population can survive any small mutant invasion.

The remainder of this paper is organized as follows. Section 2 presents the review of the current literature on the federation formation mechanisms. Section 3 presents and formulates the problem. Section 4 proposes a solution for the formation problem using a Genetic Algorithm. Section 5 explains the evolutionary game theoretical approach, and models the formation problem as a game. Section 6 provides a numerical investigation about the used models, while Section 7 states the experimental setup and analyzes the experimental results. Finally, Section 8 concludes the paper.

2. Related work

We focus in this section on presenting the work done on federated cloud computing. In [15], 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 [7], 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 [6], 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 [16], 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 [17], 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. Halabi and Bellaiche [18] 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 did not consider the impact of this proposed mechanism on the profit of the formed federations.

In [10], 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 did not 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. In our previous work [19], we proposed a minmax game to tackle the problem of having passive malicious cloud providers in the federations after the formation process had taken place. However, we did not consider stability as an issue, nor how it can affect the profit. Dhole et al. [9] 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 cannot meet that constraint, or cloud providers with a small number of VMs available.

In [20], the authors proposed a genetic approach for cloud brokering, called 'QBROKAGE'. 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.

Evolutionary game theory has mostly been used in the fields of biology, economy, and sociology. Lately, it has also been used to address the problem of bandwidth allocation, in which mobile users compete over bandwidth from several desktop users to receive multimedia streaming [21], and to analyze the advanced persistent threats against cloud storage [22]. More recently, an evolutionary game has been formulated to model the IoT devices clustering problem, in which, these devices can autonomously

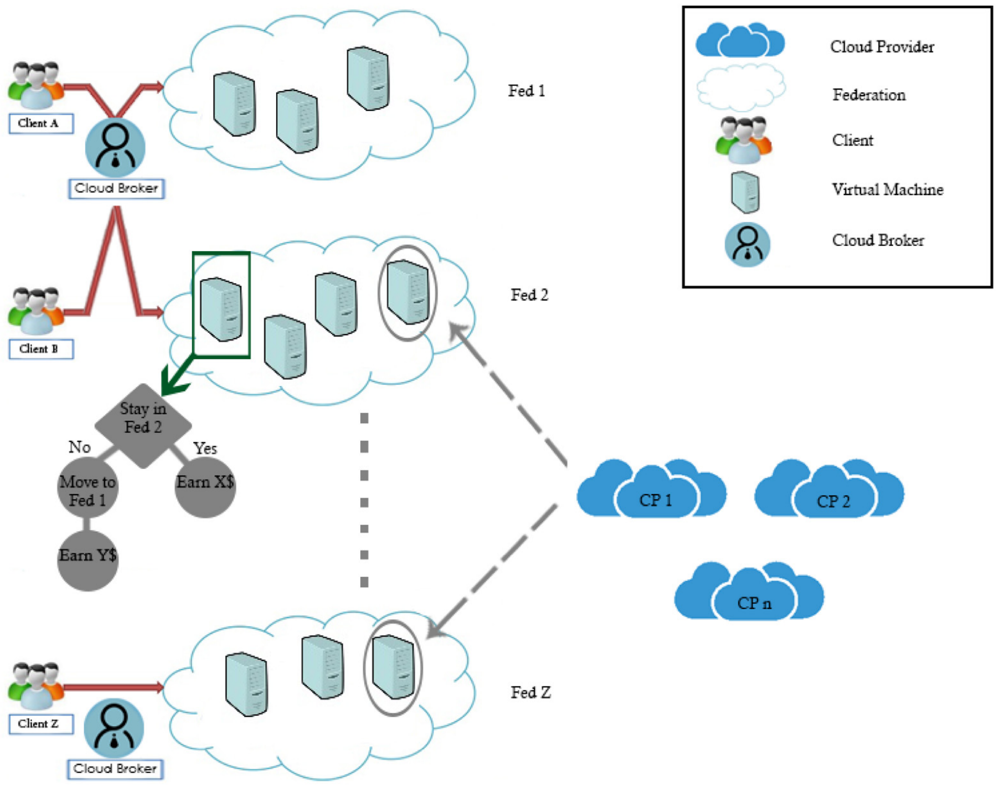


Fig. 1. Cloud federations: a VM's dilemma.

self-organize into clusters in a fully distributed manner [23]. To the best of our knowledge, none of the mentioned works tackled the problem of forming stable cloud federations using genetic and evolutionary game theory.

3. 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.

3.1. Approach overview

As shown in Fig. 1, 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 [2]. 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 will drop, which will affect its reputation. The objective of this paper 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.

3.2. System model

Let $CP = \{cp_1, cp_2, \ldots, cp_i\}$ denote the set of cloud providers, and let $F = \{f_1, f_2, \ldots, f_n\}$ represent the set of federations that serve a set of requests R. Each federation has a number of virtual machines such that $V_{f_n} = \{v_1(f_n), v_2(f_n), \ldots, v_m(f_n)\}$, 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.

3.3. Problem formulation

As stated earlier, the main purpose of our work is to find a federation structure that maximizes the providers' profit. This objective will not be met unless stability is reached since the reallocation of the cloud providers' resources to different federations will affect the QoS of the VMs, thus leading to a decrease in the users' satisfaction and hence the accrued payment. Formally, the payoff of the set F of federations at time t can be represented as follows:

$$U_t(F) = \sum_{f_i \in F} Rev_{f_i} \times Rep_t(f_i) - \sum_{v_k(f_i) \in V_{f_n}} OC_t(v_k(f_i)) + TC_t(v_k(f_i)) \quad (1)$$

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

$$Rep_t(f_i) = \frac{\theta_{f_i} \vee \gamma_{f_i}}{\gamma_{f_i}} \times \frac{\alpha_{f_i}}{100} \quad (2)$$

where γ_{f_i} represents the average response time of the virtual machines allocated to federation f_i (i.e. the time a federation takes to respond to a certain request), θ_{f_i} is 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 during which the federations were available to serve requests).

$OC_t(v_k(f_i))$ and $TC_t(v_k(f_i))$ represent the cost of running the virtual machine $v_k(f_i)$, belonging to federation f_i, on a certain host at time t, such that $OC_t(v_k(f_i))$ is the operational cost of the virtual machine, which includes CPU usage, memory and storage allocated, and energy consumption, as follows:

$$\begin{aligned} OC_t(v_k(f_i)) =& CPUCost_t(v_k(f_i)) + MemoryCost_t(v_k(f_i)) \\ &+ StorageCost_t(v_k(f_i)) + EnergyConsumption_t(v_k(f_i)) \end{aligned} \tag{3}$$

$TC_t(V_k(f_i))$ is the cost of the traffic incurred by running this VM at time t. 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:

$$Pay_t(f_i) = Rev_{f_i} \times \frac{1}{\eta_{f_i}} \tag{4}$$

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

Finally, the average payment is calculated using the following:

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

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:

$$maximize \;\; U_t(F) \tag{6}$$

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

$$minimize \;\; \sum_{f_i \in F} (Pay_t(f_i) - AvgPay_t)^2 \tag{7}$$

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. Federation formation using genetic algorithm

Genetic algorithm is a metaheuristic approach that was introduced by John Holland in 1960, and extended by Goldberg in 1989 [24]. Like any other metaheuristic technique, GA is a search technique used to efficiently solve complex optimization problems. Typically, GAs attempt to find a near optimal solution in a relatively short time. Genetic Algorithms have been applied in many fields such as computer gaming, investment strategies, digital circuits, and placement problems. What makes GA attractive is its simplicity and effectiveness [25]. The main components of GA are (1) initial population selection, (2) fitness evaluation, and (3) evolution process.

4.1. Initial population selection

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 and form 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. A federation formation/destruction shall be represented by a 0, where 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 zeros 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.

Algorithm 1 Initial Population Generator

```
 Input: pSize, pRatio, numberOfReq, prices[], vms[]
 Output: pArray
 1: $pArray = \emptyset$
 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 $pRatio \le r$ 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 $choice \le price + temp$ 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 $pArray$
```

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).

4.2. 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. 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 value or more of the chromosome 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 '12**53**64'.

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 being initialized from the whole search space. The population is generated using the metaheuristic discussed in Algorithm 1. At line 3, each candidate solution inside P is evaluated based on the predefined fitness function. Then, they get sorted from best to worst. The loop at line 5 will only break when the condition is satisfied, which might be either reaching a satisfying solution, or the federation being dissociated. The selection process at line 6 requires pairing candidates together (with proportion to their fitness), and apply genetic operators on them (i.e. crossover and mutation at lines 7 and 8). At lines 11, the new population will get evaluated by the same fitness function used at line 3. Then the population will get sorted again based on the fitness. At line 13 the worst $n/2$ candidates will get eliminated from the list, where n represents the total number of candidates inside. 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).

5. Federation formation using evolutionary game theory

Although GA performs well on improving the profit, as will be shown in Section 7, 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.

5.1. Preliminary

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 what is called payoff matrix [26]. A strategy represents the Nash Equilibrium if the first player does not get better outcome by changing his strategy, while the second player is holding to his current strategy. Therefore, there are no incentives for players to deviate from their current decisions. Some games might have more than one Nash Equilibrium, and some others might not have any.

Evolutionary game theory was introduced in [27]. This type of games shows that the analysis of game theory can be applied even if a player exhibits different forms of behavior, and does not 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 strategies being replicated by others [13].

Evolutionary stable strategy (ESS) is a strategy that once adopted by a population, it can survive even if it got invaded by any small group of invaders (i.e. having different strategies). Once that strategy is reached, it will not change over time. In case of an invasion, the invaders will die after a few generations due to the stability of that strategy. This means 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 that invasion, and force the mutants to switch their strategies to *A* due to the natural selection. For instance, let $O(A, B)$ represent the outcome of an individual choosing strategy *A* facing another with strategy *B*. *A* is stable if it represents 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 incentive to break from his current strategy, even if the population got invaded by a few mutants.

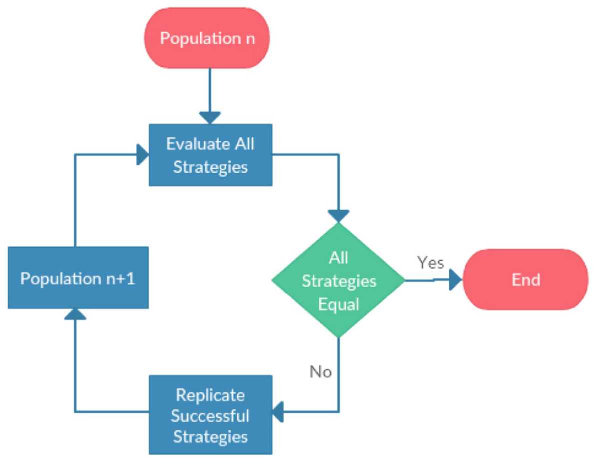


Fig. 2. Flow chart of the evolutionary game theory.

The idea behind applying evolutionary game theory is reaching the ESS where stability occurs. Fig. 2 shows the flow chart of the game, where it can initially starts from any generation that 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, 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 utilities of the strategies 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 the same payoff to their corresponding players; i.e. ESS is reached. Therefore, all players are not dissatisfied about their chosen strategies, which leads to a state where no player has incentive of switching to a different one.

5.2. Player strategy

As a rational decision maker, a cloud provider always seeks for gaining more profit. Some federations may provide more profit for some providers 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 with more than what he needs). Also, f_j might not yield the same profit (i.e. Y) used to have 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.

5.3. 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. Evolutionary games are often applied in scenarios that are characterized by large populations (finite and infinite), where in a small population, stability can be easily breached and affected when a small number of players change their strategies after the ESS has been reached [13,23,28]. In fact, in our approach we have large number of VMs forming the federations, hence we are dealing with a large population satisfying the requirement for evolutionary game based solution. It is worth mentioning that with the wide adoption of the cloud computing technology, the market is booming and the number of providers and virtual machines is becoming huge, as shown in recent reports released by MarketWatch and many others [1,4,29].
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, Eq. (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 analyze 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

$$\sum_{i=1}^{n} x_i = 1 \tag{8}$$

The general form of the replicator equation is as follows

$$\dot{x}_i = x_i[f_i(x) - \o(x)] \tag{9}$$

where $f_i(x)$ is the fitness function of selecting a strategy i (i.e. the i^{th} federation), which is Eq. (4), and $\o(x)$ 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 Eq. (10).

$$\o(x) = \sum_{j=1}^{n} x_j f_j(x) \tag{10}$$

Eq. (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 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 the average utility $ø(x)$ 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 $ø(x)$ 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 for all $x_i \in x$, $\dot{x}_i$ is 0.

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 $ø(x)$
8:     for all $vm_j \in VMs$ do
9:         if $ø(x) > f_{strat(vm_j)}(x)$ 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:            $ESS\_Reahced = false$
17:        end if
18:    end for
19:    if $ESS\_Reached == true$ then
20:        break
21:    end if
22:    $t = t + 1$
23: end while
```

Our evolutionary game is considered dynamic by nature since it is time-aware, which enables the players to learn and adjust their strategies from one time period to another in such a way to maximize their profit.

The choice of an evolutionary non-cooperative game over a classical non-cooperative or a coalitional game stems from the fact that it does not impose any strict assumption regarding the players' rationality. In fact, a coalitional (or cooperative) game model is a game model in which players cooperate in order to create an added value and then share the resulting profit. Unlike cooperative games, in non-cooperative games (including evolutionary ones), players participate individually in order to maximize their own profits in response to other players' strategies. Both classical cooperative and non-cooperative games models assume that players always make rational choices. However, in practical scenarios, players might be tempted to act in an irrational manner due to some external or unexpected factors, which might be the case in our problem. In this regard, evolutionary game theory is the best candidate solution since it accounts for this fact by achieving the equilibrium through the evaluation process that replicates the successful strategies and repulse the non-successful ones over the time [30].

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

Proof. The algorithm solves the replicator dynamics' $\dot{x}_i = 0$ 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 would be zero.

The evolutionary stable strategy will remain effective even if a small portion of the players decide to change their federations [23]. The motivations that might push some players to change their federations after reaching the stability might be related to some technical problems in the virtual machines, new resources added to some federations, or the formation of a new set of federations. However, even in such cases, our algorithm will still be able to restore equilibrium in a short time and converge again to a stable state through constantly replicating the successful strategies.

5.4. Case study: Two cloud federations

An evolutionary equilibrium is a fixed point of the replicator dynamics [31]. We show that there exists an equilibrium in the following scenario:

The evolutionary equilibrium will occur when $\dot{x}_i = 0$. For simplicity and without loss of generality, suppose now that we have two federations f_1, f_2, and a set of providers having η virtual machines in total that need to be deployed. In order to reach the equilibrium, we need to solve the following equality (which means that the providers are indifferent between joining f_1 or f_2):

$$f_1(x) = f_2(x) \tag{11}$$

The equality can be converted into the following expression:

$$Rev_{f_1} \times \frac{1}{\eta_{f_1}} = Rev_{f_2} \times \frac{1}{\eta_{f_2}} \tag{12}$$

By substituting the η_{f_i} with $x_i \times \eta$, and since we only have 2 strategies, we can express x_2 in terms of x_1, such that:

$$x_2 = 1 - x_1 \tag{13}$$

The equation then becomes:

$$Rev_{f_1} \times \frac{1}{x_1 \times \eta} = Rev_{f_2} \times \frac{1}{(1 - x_1)\eta} \tag{14}$$

x_1 can then be computed as per Eq. (15) as follows:

$$x_1 = \frac{Rev_{f_1}(1 - x_1)}{Rev_{f_2}} \tag{15}$$

The stability can be then analyzed through evaluating the Jacobian matrix of the replicator dynamics' as per the following:

$$J = \begin{bmatrix} \frac{\partial \sigma x_1^a (f_1^a(x) - ø^a(x))}{\partial x_1^a} & \frac{\partial \sigma x_1^a (f_1^a(x) - ø^a(x))}{\partial x_1^b} \\ \frac{\partial \sigma x_1^b (f_1^b(x) - ø^b(x))}{\partial x_1^a} & \frac{\partial \sigma x_1^b (f_1^b(x) - ø^b(x))}{\partial x_1^b} \end{bmatrix} = \begin{bmatrix} J_{1,1} & J_{1,2} \\ J_{2,1} & J_{2,2} \end{bmatrix} \tag{16}$$

Table 1
Requests.

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

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

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

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

The fixed point is considered stable if all of the eigenvalues have a negative real part [31]:

$$\lambda(J) = \frac{(J_{1,1} + J_{2,2}) \pm \sqrt{4J_{1,2}J_{2,1} + (J_{1,1} - J_{2,2})^2}}{2} \tag{17}$$

6. 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 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 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 $5{*}2.4{+}2{*}4{+}5{*}4 = 40$.

6.1. Solution using genetic algorithm

We try to stabilize the federations by finding the optimal formation using Algorithm 2. After the first generation, GA seeks to come up with a better formation to be a solution for the second generation by finding a way to increase the payoff (Section 3, Eq. (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, i.e. reaching 4. The second federation got an 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.

6.2. 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 allocated for the first request, 4 VMs for the second, and 5 for 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.

7. Experimental evaluation

In this section, we investigate the performance of our two proposed models compared to the Sky federation model proposed in [10].

7.1. 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 80 cloud providers having different numbers of available virtual machines varying from 10 to 30. At any time, a number of clients can request a specific amount of CPU cores. Requests have been categorized as small, medium, and large [10]. 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 received requests. We compared our two

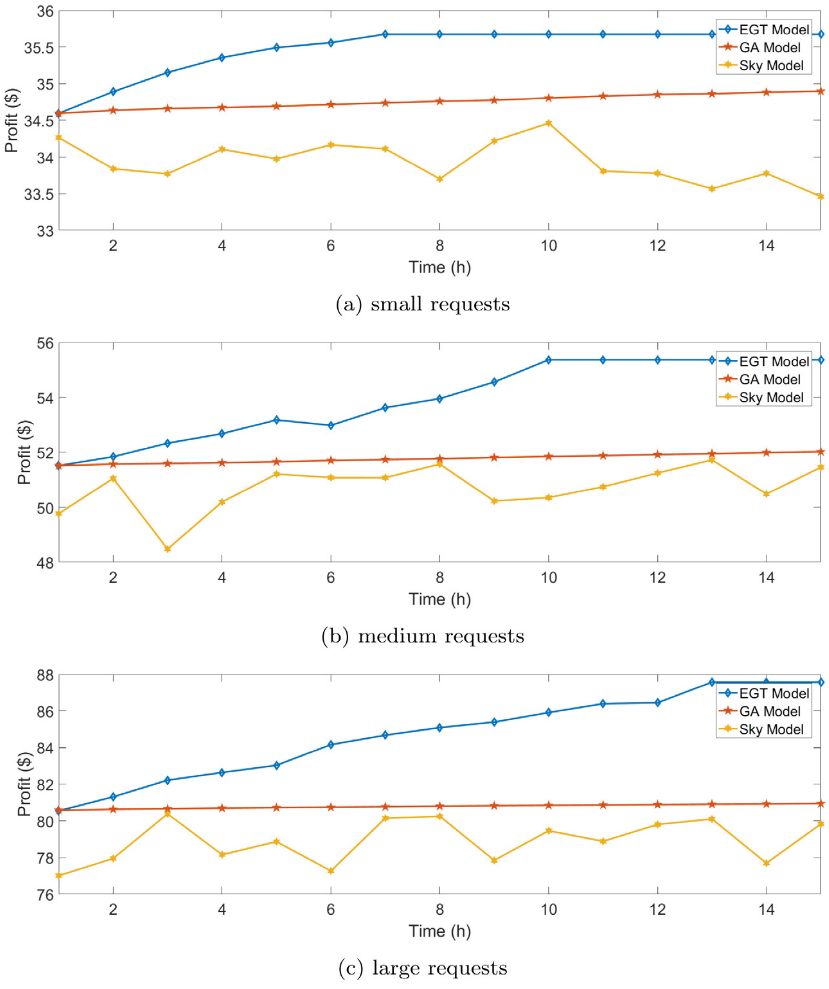


Fig. 3. Total profit.

Table 5
Requests types.

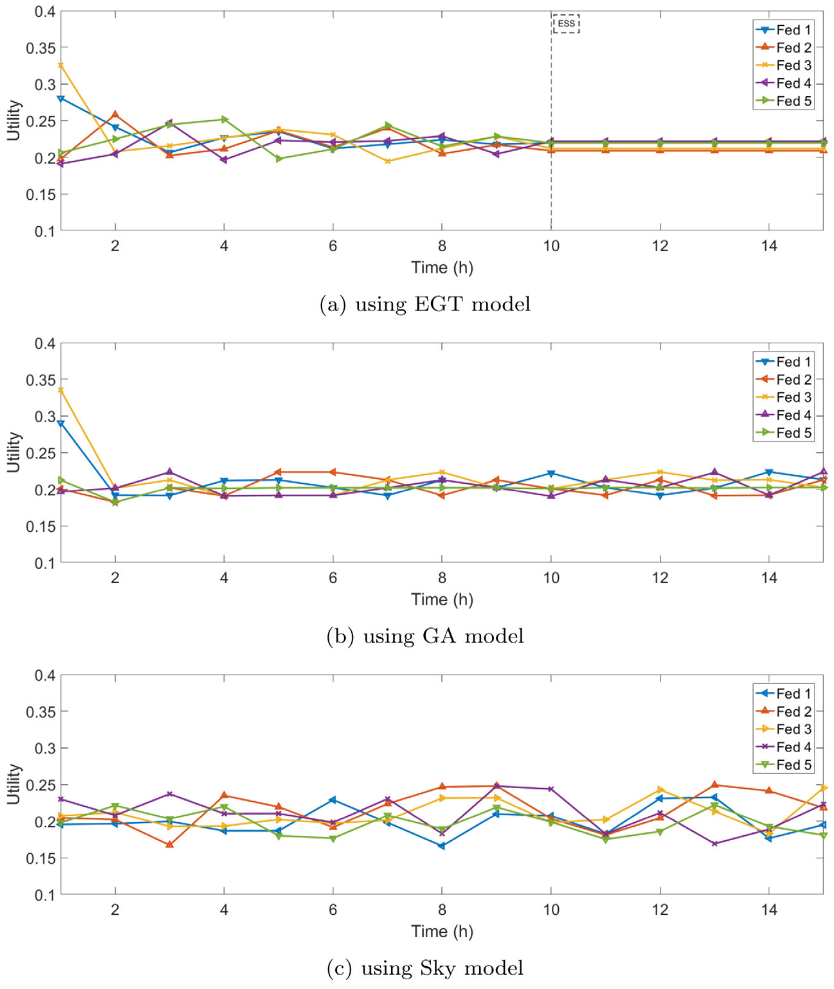
Type	Number of federations	Number of VMs
Small	3	240
Medium	5	400
Large	7	560

models with the sky model in [10] in terms of total profit, utility, response time, availability, saved energy and saved resources. 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, availability, resources and energy saved) 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 dataset[1] 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.

7.2. Results and discussion

The first set of experiments aims to study the total profit obtained by the set of federations (Fig. 3). In Fig. 3(a), 3(b), and 3(c), 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 h (i.e. 7 generations). It started as 51.5$ for medium requests and reached 55.5$ in 10 h. For large requests, the profit increased from 80.5$ to 87.5$ in 13 h. 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 could not 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

1 http://cloudharmony.com/.



(a) using EGT model

(b) using GA model

(c) using Sky model

Fig. 4. VM utility. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

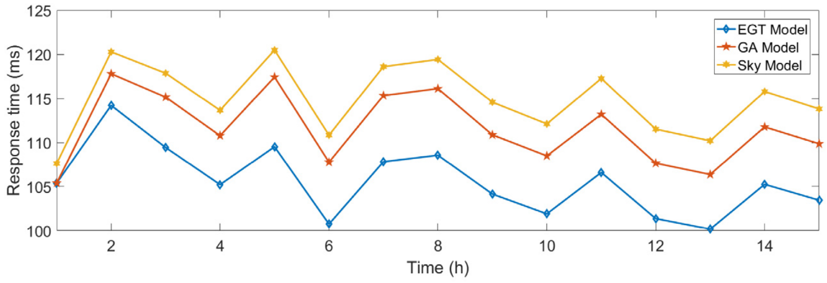
three models because their algorithm consists of only forming the federations from the first hour, and neglecting the fact that the virtual machines will change federation with time, which makes it easier for a cloud provider to leave and join other federation in a very random way, hence leading to an 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. 4). 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. 4(a), 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. 4(b) and 4(c)) 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.

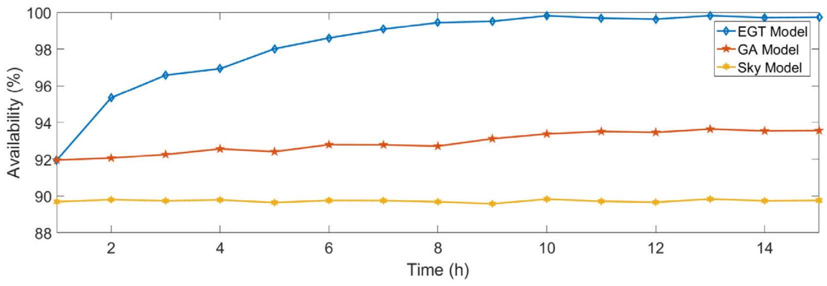
Moreover, we measured the QoS of the formed federations (Fig. 5). In Fig. 5(b) and 5(a), we evaluate the availability and response time of the federations respectively using the three models. The availability represents the percentage of times the VMs were able to execute their assigned tasks normally 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 could not 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.

In the last set of experiments, we measured the saved energy and resources of the formed federations. Fig. 6(a) shows the impact of the different models compared based on the energy consumed by the virtual machines. The energy was calculated according to the study in [32], in which they studied the power consumption of VMs while performing networking tasks. The figure reveals that both the GA and EGT entail less energy consumption through saving up to 1,000,000 and 3,200,000 Joules per hour respectively at time t = 10. On the other hand, the Sky approach entails high energy consumption and could not improve with time due to neglecting the fact that the virtual machines will change federation with time.

Also, in order to observe the effect of the three models on the formed federations in term of saved resources (i.e., CPU, RAM and storage), we measured how these models enhanced the percentage of saved resources (Fig. 6(b)). We notice that the

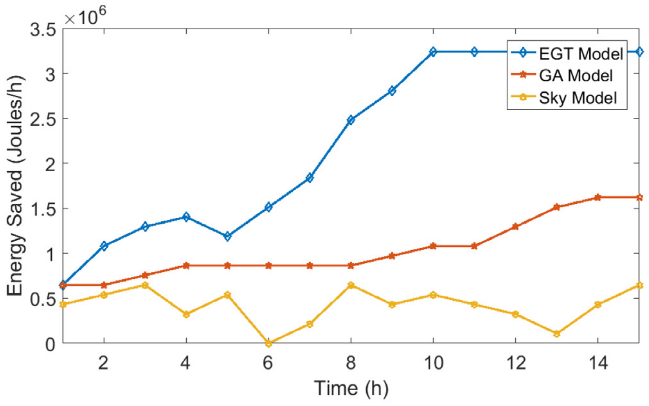


(a) response time

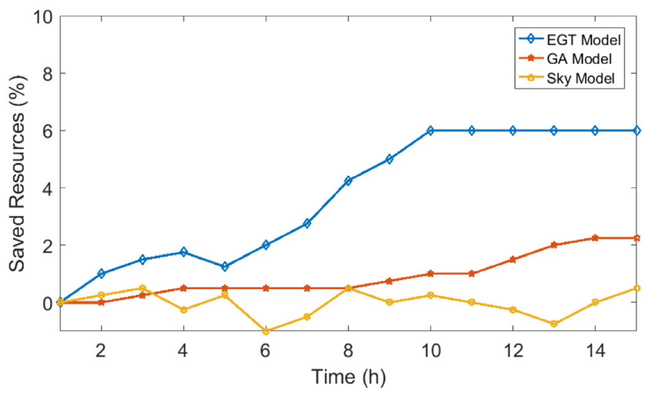


(b) availability

Fig. 5. Quality of service.



(a) Energy Saved



(b) Resources Saved

Fig. 6. Energy and resources saved.

evolutionary game model saved up to 6.0% of resources at time t = 10, i.e., 48 6-cores, 768 GB of RAM, and 7200 GB of storage on average. The genetic model kept on saving resources until reaching 1.0% at time t = 10, and up to 2.25% at time t = 15, which is equivalent to 18 6-cores, 288 GB of RAM, and 2700 GB of storage on average. On the other hand, the performance of the Sky model shows a considerable instability compared to the two other models.

8. Conclusion

Cloud federation is an architecture that allows cloud providers to make use of their unallocated virtual machines, by merging their resources together to serve a pool of clients whose requests cannot be handled by any of these providers alone. In this paper, 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 does not 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 [10]. 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. Moreover both models can help in saving energy and resources. 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%.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been supported by the Lebanese American University and the Center for Cyber–Physical Systems (C2PS), Khalifa University, Abu Dhabi, UAE.

References

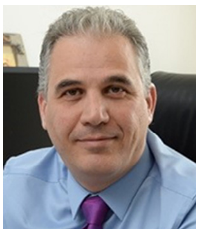
[1] J. Novet, Amazon cloud revenue jumps 45 percent in fourth quarter. CNBC, 2018, Retrieved from https://www.cnbc.com/2018/02/01/aws-earnings-q4-2017html.
[2] Michael Hogan, Fang Liu, Annie Sokol, Jin Tong, Nist cloud computing standards roadmap, NIST Special Publication 35 (2011) 6–11.
[3] Sky-high market growth driving demand for cloud infrastructure specialists, Networkers, 2018, Retrieved from https://www.networkerstechnology.com/growth-cloud-dem{and}-infrastructure-specialists.
[4] B. Evans, Top cloud vendors will crush $100 billion in 2018 revenue; Microsoft, Amazon, IBM hit $75 billion? Forbes, 2018, Retrieved from https://www.forbes.com/sites/bobevans1/2018/05/21/top-cloud-vendors-will-crush-100-billion-in-2018-revenue-microsoft-amazon-ibm-hit-75-billion/#3aaed51c7548.
[5] R. Buyya, J. Broberg, A.M. Goscinski (Eds.), Cloud Computing: Principles and Paradigms (Vol. 87), John Wiley & Sons, 2010.
[6] S. Rebai, M. Hadji, D. Zeghlache, Improving profit through cloud federation, in: Consumer Communications and Networking Conference, CCNC, 2015 12th Annual IEEE, IEEE, 2015, pp. 732–739.
[7] Inigo Goiri, Jordi Guitart, Jordi Torres, Characterizing cloud federation for enhancing providers' profit, in: Cloud Computing, CLOUD, 2010 IEEE 3rd International Conference on, IEEE, 2010, pp. 123–130.
[8] M. Guazzone, C. Anglano, M. Sereno, A game-theoretic approach to distributed coalition formation in energy-aware cloud federations (extended version), 2013, arXiv preprint arXiv:13092444.
[9] A. Dhole, M.V. Thomas, K. Chandrasekaran, An efficient trust-based Game-Theoretic approach for cloud federation formation, in: Advanced Computing and Communication Systems, ICACCS, 2016 3rd International Conference on, Vol. 1, IEEE, 2016, pp. 1–6.
[10] Lena Mashayekhy, Mahyar Movahed Nejad, Daniel Grosu, Cloud federations in the sky: Formation game and mechanism, IEEE Trans. Cloud Comput. 3 (1) (2015) 14–27.
[11] Ahmed A Alabdel Abass, Liang Xiao, Narayan B. Mandayam, Zoran Gajic, Evolutionary game theoretic analysis of advanced persistent threats against cloud storage, IEEE Access 5 (2017) 8482–8491.
[12] Chonho Lee, Junichi Suzuki, Athanasios Vasilakos, Yuji Yamamoto, Katsuya Oba, An evolutionary game theoretic approach to adaptive and stable application deployment in clouds, in: In the 2nd Workshop on Bio-Inspired Algorithms for Distributed Systems, ACM, 2010, pp. 29–38.
[13] David Easley, Jon Kleinberg, Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Cambridge University Press, 2010.
[14] J. Newton, Evolutionary game theory: a renaissance, Games 9 (2) (2018) 31.
[15] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I.M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, et al., The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 535–545.
[16] S.H. Bhuiyan, M.M. Hasan, Revenue maximization in cloud federation based on multi-choice multidimensional knapsack problem, in: 2018 21st International Conference of Computer and Information Technology, ICCIT, IEEE, 2018, pp. 1–6.
[17] A.N. Toosi, R.N. Calheiros, R.K. Thulasiram, R. Buyya, Resource provisioning policies to increase iaas provider's profit in a federated cloud environment, in: High Performance Computing and Communications, HPCC, 2011 IEEE 13th International Conference on, IEEE, 2011, pp. 279–287.
[18] T. Halabi, M. Bellaiche, Towards security-based formation of cloud federations: A game theoretical approach, IEEE Trans. Cloud Comput. (2018).
[19] A. Hammoud, H. Otrok, A. Mourad, O.A. Wahab, J. Bentahar, On the detection of passive malicious providers in cloud federations, IEEE Commun. Lett. 23 (1) (2019) 64–67.
[20] G.F. Anastasi, E. Carlini, M. Coppola, P. Dazzi, QoS-aware genetic cloud brokering, Future Gener. Comput. Syst. 75 (2017) 1–13.
[21] G. Nan, Z. Mao, M. Yu, M. Li, H. Wang, Y. Zhang, Stackelberg game for bandwidth allocation in cloud-based wireless live-streaming social networks, IEEE Syst. J. 8 (1) (2014) 256–267.
[22] A.A.A. Abass, L. Xiao, N.B. Mandayam, Z. Gajic, Evolutionary game theoretic analysis of advanced persistent threats against cloud storage, IEEE Access 5 (2017) 8482–8491.
[23] N. Sawyer, M.N. Soorki, W. Saad, D.B. Smith, N. Ding, Evolutionary games for correlation-aware clustering in massive machine-to-machine networks, IEEE Trans. Commun. (2019).
[24] J. Sadeghi, S. Sadeghi, S.T.A. Niaki, Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: an improved particle swarm optimization algorithm, Inform. Sci. 272 (2014) 126–144.
[25] S. Rajasekaran, G.V. Pai, Neural Networks, Fuzzy Logic and Genetic Algorithm: Synthesis and Applications (with Cd), PHI Learning Pvt. Ltd, 2003.
[26] D. Fudenberg, J. Tirole, Perfect bayesian equilibrium and sequential equilibrium, J. Econom. Theory 53 (2) (1991) 236–260.
[27] J.M. Smith, G.R. Price, The logic of animal conflict, Nature 246 (5427) (1973) 15.
[28] M.A. Nowak, A. Sasaki, C. Taylor, D. Fudenberg, Emergence of cooperation and evolutionary stability in finite populations, Nature 428 (6983) (2004) 646.
[29] Cloud Computing Server Market Is Booming Worldwide | Intel, IBM, Amazon, Google Cloud Platform, Salesforce. marketwatch. (2019, April). Retrieved from https://www.marketwatch.com/press-release/cloud-computing-server-market-is-booming-worldwide-intel-ibm-amazon-google-cloud-platform-salesforce-2019-04-23.
[30] J.M. Smith, Evolution and the Theory of Games, Cambridge university press, 1982.
[31] Y.A. Kuznetsov, Elements of Applied Bifurcation Theory (Vol. 112), Springer Science & Business Media, 2013.
[32] R. Shea, H. Wang, J. Liu, Power consumption of virtual machines with network transactions: Measurement and improvements, in: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 1051–1059.

Ahmad Hammoud received his Bachelor degree in business computing from the Lebanese University and Master's degree in computer science from the Lebanese American University (LAU), Beirut, Lebanon. His current research interests include cloud and fog federation, game theory, and security.

Azzam Mourad received the Ph.D. degree in electrical and computer engineering from Concordia University, Montreal, Canada. He is an associate professor of computer science at the Lebanese American University and Affiliate Associate Professor in Software Engineering and IT department at the Ecole de Technologie Superieure (ETS), Montreal, Canada. He served/serves as Associate Editor for IET Quantum Communication and IEEE Communications Letters, General Chair of IWCMC 2020, General Co-Chair of WiMob2016, and Track Chair, TPC member and reviewer of several prestigious conferences and journals. He is an IEEE senior member.

Hadi Otrok holds an associate professor position in the department of ECE at Khalifa University of Science and Technology, an affiliate associate professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada, and an affiliate associate professor in the electrical department at Ecole de Technologie Superieure (ETS), Montreal, Canada. He received his Ph.D. in ECE from Concordia University. He is a senior member at IEEE, and associate editor at: Ad-hoc networks (Elsevier) and IEEE communications letters. He co-chaired several committees at various IEEE conferences. His research interests include the domain of computer and network security, crowd sensing and sourcing, ad hoc networks, and cloud security.

Omar Abdel Wahab is an assistant professor at the Department of Computer Science and Engineering, Université du Québec en Outaouais, Canada. He holds a Ph.D. in Information and Systems Engineering from Concordia University, Montreal, Canada. He received his Ms.c. in computer science in 2013 from the Lebanese American University (LAU), Lebanon. From 2017 to 2018, he was a postdoctoral fellow at the École de Technologie Supérieure (ÉTS), Canada, where he worked on an industrial research project in collaboration with Rogers and Ericsson. The main topics of his current research activities are in the areas of artificial intelligence, cybersecurity, cloud computing, and big data analytics. He is recipient of many prestigious awards including Quebec Merit Scholarship (FRQNT Québec). Moreover, he is a TPC member of several prestigious conferences and reviewer of several highly ranked journals.



Haidar Harmanani received his BS, M.S, and Ph.D. in Computer Engineering from the Department of Electrical Engineering and Computer Science at Case Western Reserve University, Cleveland, Ohio, in 1989, 1991, and 1994 respectively. He is currently a professor of computer science at the Lebanese American University, Lebanon. He is a senior member of IEEE and a senior member of ACM. He serves on the steering committee of the IEEE NEWCAS conference and the IEEE ICECS conference. He has also served on the program committee of various international conferences. His research interests include electronic design automation, high-level synthesis, design for testability, and parallel programming.