A Game-Theoretic Approach to Distributed Coalition Formation in Energy-Aware Cloud Federations

Marco Guazzone[1], Cosimo Anglano[2], Matteo Sereno[1]
[1]Department of Computer Science, University of Torino, Italy
[2]Department of Science and Technological Innovation, University ofl Piemonte Orientale, Italy

ABSTRACT

Federations among sets of Cloud Providers (CPs), whereby a set of CPs agree to mutually use their own resources to run the VMs of other CPs, are considered a promising solution to the problem of reducing the energy cost. In this paper, we address the problem of federation formation for a set of CPs, whose solution is necessary to exploit the potential of cloud federations for the reduction of the energy bill. We devise a distributed algorithm, based on cooperative game theory, that allows a set of CPs to cooperatively set up their federations in such a way that their individual profit is increased with respect to the case in which they work in isolation, and we show that, by using our algorithm and the proposed CPs' utility function, they are able to self-organize into Nash-stable federations and, by means of iterated executions, to adapt themselves to environmental changes. Numerical results are presented to demonstrate the effectiveness of the proposed algorithm.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; K.6.4 [**Management of Computing and Information Systems**]: System Management

General Terms

Management,Performance

Keywords

Cloud Federation, Cooperative Game Theory, Coalition Formation

1. INTRODUCTION

Many modern Internet services are implemented as cloud applications consisting of a set of *Virtual Machines* (VMs) that are allocated and run on a physical computing infrastructure managed by a virtualization platform (e.g., Xen [11], VMware [4], etc.). These infrastructure are typically owned by a *Cloud Provider* (CP) (e.g., Amazon AWS, Rackspace, Windows Azure, etc.), and are located into a (set of possibly distributed) data center(s).

One of the key issues that must be faced by a CP is the reduction of its energy cost, that represents a large fraction of the total cost of ownership for physical computing infrastructures [33]. This cost is mainly due to the consumption of the physical resources that must be switched on to run the workload.

To reduce energy consumption, two techniques are therefore possible for a CP: (a) to minimize the number of hosts that are switched on by maximizing the number of VMs allocated on each physical resource (using suitable resource management techniques [20, 6]), and (b) to use resources that consume less energy.

Cloud federations [36], whereby a set of CPs agree to mutually use their own resources to run the VMs of other CPs, are considered to be a promising solution for the reduction of energy costs [14] as they ease the application of both techniques.

As a matter of fact, while each individual CP is bound to its specific energy provider and to the physical infrastructure it owns, a set of federated CPs may enable the usage of more flexible energy management strategies that, by suitably relocating the workload towards CPs that pay less for the energy, or that have more energy-efficient resources, may reduce the energy bill for each one of them.

In order to exploit the energy saving potential of cloud federations, it is however necessary to address the question concerning its formation. As a matter of fact, it is unreasonable to assume that a CP unconditionally joins a federation regardless of the benefits it receives, while it is reasonable to expect that it joins a federation only if this brings it a benefit.

In this paper, we address the problem of federation formation for a set of CPs, and we devise an algorithm that allows these CPs to decide whether to federate or not on the basis of the profit they receive for doing so. In our approach, each CP pays for the energy consumed by each VM, whether it belongs to its own workload or to the one of another CP, but receives a payoff (computed as discussed later) for doing so.

The algorithm we propose is based on cooperative game theory [32, 34]. In particular, we rely on *hedonic games* (see [12] for their definition) whereby each CP bases its decision on its own preferences. Depending on the specific operational conditions of each CP (i.e., the resource requirements of its workload, its cost of energy, the energy consumption of its physical machines, and the revenue it obtains when running each VM on these machines), different federations (each one consisting of a subset of the CPs), or even no federation at all, may be formed by the involved CPs. We call *federation set* the set of distinct federations formed by a set of CPs.

The algorithm we propose computes the federation set that results in the highest profit that can be achieved by a set of autonomous and selfish CPs. This derives from the fact that this algorithm ensures that all the federations formed by groups of CPs are *stable*, that is CPs have no incentive to leave the federation once they decide to participate.

Unlike similar proposals (e.g., [26]), that rely on a centralized architecture in which a trusted third party computes the federation set, we adopt a distributed approach in which each CP autonomously and selfishly makes its own decisions, and the best solution emerges from these decisions without the need of synchronizing them, or to resort to a trusted third party. In this way, we avoid two drawbacks that affect existing proposals, namely the difficulty of finding a third party that is trusted by all the CPs, and the need to suspend the operations of all the CPs when the federation set is being computed.

The rest of this paper is organized as follows. In Section 2, we describe the system under study, and provide some simple motivating example. In Section 3, we present a cooperative game-theoretic model of the system under study, and show stability conditions and profit allocation strategies that provide the theoretical foundation for the distributed coalition formation algorithm, that is also presented in this section. In Section 4, we present results from an experimental evaluation to show the effectiveness of the proposed approach. In Section 5, we provide a short overview of related works. Finally, conclusions and an outlook on future extensions are presented in Section 6.

2. PROBLEM FORMULATION

In this section, we first formally describe the problem addressed in the paper (see Section 2.1), and then we illustrate some issues that must be properly addressed in order to properly solve it (see Section 2.2).

2.1 System Description

We consider a set of n CPs denoted by $\mathcal{N} = \{1, 2, \dots, n\}$, where each CP i is endowed with a set $\mathcal{H}_i$ of physical hosts. We denote as $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \cdots \cup \mathcal{H}_n$ the set of all the hosts collectively belonging to the various CPs. These hosts are grouped into a set $\mathcal{G}$ of *host classes* according to their processor type and to the amount of physical memory they provide; all the hosts in the same class are homogeneous in terms of processor and memory size. For any $h \in \mathcal{H}$ we denote by $g(h)$ the function that gives the host class of h (i.e., a function $g : \mathcal{H} \to \mathcal{G}$).

As discussed in [35], we assume that a host h consumes $C^{\min}_{g(h)}$ Watts when its CPU is in the idle state, $C^{\max}_{g(h)}$ Watts when its CPU is fully utilized, and $\Big(C^{\min}_{g(h)} + f \cdot \big(C^{\max}_{g(h)} - C^{\min}_{g(h)}\big)\Big)$ when a fraction $f \in [0, 1]$ of its CPU capacity is used. This model, albeit simple, has been shown to provide accurate estimates of power consumption for different host types when running several benchmarks representative of real-world applications [35].

Physical hosts run cloud workloads, consisting in a set $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2 \cup \cdots \cup \mathcal{J}_n$ of VMs, where $\mathcal{J}_i$ denotes the set of VMs that compose the workload of the i-th CP (each VM contains the whole execution environment of one or more applications).

As typically done by CPs, VMs are grouped into a set $\mathcal{Q}$ of *VM classes* according to the computing capacity provided by their virtual processors, and to the amount of physical memory they are equipped with; all the VMs belonging to the same class provide the same amount of computing capacity and of physical memory. For instance, Amazon's EC2 [1] defines the *Elastic Compute Unit (ECU)* as an abstract computing resource able to deliver a capacity equivalent to that of a 1.2 GHz 2007 Xeon processor, and provides various *instance types* (that are equivalent to our VM classes) that differ among them in the number of ECUs and in the amount of RAM they are equipped with. More specifically, *small*, *medium*, and *large*, corresponding to VM class 1, 2, and 3, respectively, provide 1 ECU and 1.7 GB of RAM, 2 ECUs and 3.7 GB of RAM, and 4 ECUs and 7.5 GB of RAM, respectively.

For any VM $j \in \mathcal{J}$, we denote by $q(j)$ the function that gives its VM class (i.e., a function $q : \mathcal{J} \to \mathcal{Q}$). Using this notation, for any $j \in \mathcal{J}$ we denote by $CPU_{q(j)}$ and by $RAM_{q(j)}$ the amount of computing capacity and of physical memory of VM j, respectively. As an example, in the Amazon EC2 case we have that $CPU_1 = 1$ ECU and $RAM_1 = 1.7$ GB, while $CPU_3 = 4$ ECU and $RAM_3 = 7.5$ GB.

When allocated on a physical host h, a VM j uses a certain fraction $A_{q(j),g(h)}$ of CPU capacity and a certain fraction $M_{q(j),g(h)}$ of physical memory. $A_{q(j),g(h)}$ can be determined by measuring, with a suitable benchmark (e.g., GeekBench [2]), the computing capacity Cap_v delivered by the virtual processor of VMs in $q(j)$ and the capacity Cap_p delivered by the physical processor of hosts in $g(h)$, and then by dividing these quantities, i.e.,

$A_{q(j),g(h)} = \frac{Cap_v}{Cap_p}$. For instance, if $Cap_v = 1,000$ and $Cap_p = 8,000$, then $A_{q(j),g(h)} = 0.125$. $M_{q(j),g(h)}$ can instead be computed as $M_{q(j),g(h)} = \left\lceil \frac{RAM_{q(j)}}{\text{RAM size of hosts in } g(h)} \right\rceil$.

Each CP i charges, for each VM j, a *revenue rate* (that depends on the class $q(j)$ of that VM) that specifies the amount of money that the VM owner must correspond per unit of time. For instance, Amazon charges 0.08 \$/hour, 0.16 \$/hour, and 0.32 \$/hour for *small*, *medium*, and *large* instances, respectively. Consequently, CP i earns a global revenue rate that is given by the sum of the revenue rates of individual VMs. To run its workload, CP i incurs an energy cost quantified by the *energy cost rate* (the amount of money that is paid per unit of time), which is the energy cost resulting from the allocation of (a subset of) the workload $\mathcal{J}$ on its host set $\mathcal{H}_i$ that must be paid per unit of time (see Section 3.3 for a discussion on the optimization technique we use to minimize it). We define the *net profit rate* of CP i as the difference between its global revenue rate (obtained for hosting a set of VMs) and its global energy cost rate (that it incurs to run such VMs).

Our goal is to allocate all the VMs in $\mathcal{J}$ on the hosts in $\mathcal{H}$ (independently from the corresponding CPs) in such a way to maximize the *net profit rate* of each CP i. This goal can be achieved by finding the smallest set of hosts that are sufficient to accommodate the resource shares of all the VMs in $\mathcal{J}$ such that the overall energy consumption is minimized, and by providing a suitable revenue for those CPs that host VMs belonging to other CPs.

Table 1: Operational scenarios of CP$_1$, CP$_2$, CP$_3$

(a) Characteristics of host classes

Host Class	CPU	RAM (GB)	$C^{\min}/C^{\max}$ (W)
1	2× Xeon 5130	16	86.7/274.9
2	Xeon X3220	32	143.0/518.4
3	2× Xeon 5160	64	490.1/1,117.8

(b) Characteristics of VM classes

VM Class	Processor	#CPUs	RAM (GB)
1	AMD Opteron 144	1	1
2	AMD Opteron 144	2	2
3	AMD Opteron 144	4	4

(c)Per-VM physical resource shares

Host Class	Class-1 VM	Class-2 VM	Class-3 VM
1	$(0.20, 0.062500)$	$(0.4, 0.12500)$	$(0.8, 0.2500)$
2	$(0.15, 0.031250)$	$(0.3, 0.06250)$	$(0.6, 0.1250)$
3	$(0.10, 0.015625)$	$(0.2, 0.03125)$	$(0.3, 0.0625)$

Table 2: Scenario 1 results

CP	Powered-on Hosts	Consumed Power (kW)	Energy Cost (\$/hour)
(a) no federation among CPs			
CP$_1$	10	2.37	0.95
CP$_2$	10	3.68	1.47
CP$_3$	4	3.84	1.54
Total	24	9.89	3.96
(b) federation among all the CPs			
CP$_1$	30	7.12	2.85
CP$_2$	0	0.00	0.00
CP$_3$	0	0.00	0.00
Total	30	7.12	2.85

2.2 Issues in coalition formation

The most straightforward way to form a coalition[1] is to include in it *all* the CPs (the *grand coalition*). This solution is certainly attractive because of its simplicity and ease of implementation, and can bring significant benefits to its participant.

To illustrate, let us consider three different CPs, named CP$_1$, CP$_2$, and CP$_3$, whose operational scenarios are characterized by the values shown in Table 1, where we report the characteristics of the host classes (Table 1(a)), of the VM classes (Table 1(b)), and the resource shares of each VM class (Table 1(c)). Assume, for the moment, that we have both a way to compute the set of hosts that must be switched on to minimize energy consumption (a suitable optimization problem is presented in Section 3.3), and a profit distribution rule that yields suitable revenues to CPs hosting external VMs, so that the minimization of the energy consumption within a federation of CPs corresponds to the maximization of their net profit rates (such a rule is discussed in Section 3.1).

[1]In this paper, the terms *federation* and *coalition* (which is widely adopted in the game-theoretic community) are used interchangeably.

Now, let us consider a simple scenario (that we name *Scenario* 1) in which each CP owns 30 hosts of a single class, and in particular that CP$_1$, CP$_2$, and CP$_3$ own only class-1, class-2, and class-3 hosts, respectively; furthermore, all the CPs have the same workload (in particular, each CP has to allocate 10 class-3 VMs) and energy cost (0.4 \$/kWh).

If each CP uses only its own resources and allocate its own workload (i.e., no federation is formed), it achieves the energy cost rate reported in Table 2(a), where also the total cost rate is reported. If, conversely, they form a grand coalition (i.e, they jointly perform a global workload allocation using the union of their respective host sets), their corresponding individual and overall energy cost rates are reported in Table 2(b).

As can be seen from these results, the grand coalition yields a smaller total cost rate of energy that, as discussed before in Section 2.1, corresponds to a larger net profit rates for the individual CPs. In particular, the overall energy cost rate is reduced by 28% (from 3.96 \$/hour to 2.85 \$/hour), thanks to the fact that in the federation case only the hosts belonging to CP$_1$ are used to run all the VMs.

Table 3: Scenario 2 results

CP	Powered-on Hosts	Consumed Power (kW)	Energy Cost ($/hour)
(a) no federation among CPs			
CP_1	22	10.47	4.19
CP_2	13	14.03	5.61
CP_3	13	14.03	5.61
Total	48	38.53	15.41
(b) federation among all CPs			
CP_1	41	19.71	7.89
CP_2	9	9.93	3.97
CP_3	4	4.47	1.79
Total	54	34.11	13.65
(c) CP_1 and CP_2 federated among them, CP_3 alone			
CP_1	42	20.20	8.08
CP_2	0	0.00	0.00
CP_3	13	14.03	5.61
Total	84	34.23	13.69

Given the benefits resulting from the grand coalition in this example, it is natural to speculate whether the problem of computing a federation set really arises in practice.

Unfortunately, as shown below, there are cases when the grand coalition does not represent the best solution for all the involved CPs. Indeed, consider another scenario (that we name *Scenario* 2) where the characteristics of the hosts and of the VMs are the same as before (see Table 3), but the number and type of hosts and VMs differ from those assumed in *Scenario* 1. In particular, now CP_1 owns 42 class-2 hosts and its workload consists in 65 class-2 VMs, while CP_2 and CP_3 own 41 class-3 hosts each and both have 61 class-2 VMs as workload.

The individual and overall energy cost rates corresponding to the optimal solution for the no federation and the grand coalition cases are reported in Table 3(a) and (b), respectively. Again, we observe a reduction of the energy cost in the grand coalition case (seeTable 3(a) and (b)), although this reduction is smaller than in the *Scenario* 1 case (it amounts to 11.4%).

However, by looking at the results in Table 3(c), we can observe that if CP_1 and CP_2 federate among them and exclude CP_3, their overall energy consumption rate *is smaller* than in the case of the grand coalition, although the overall cost involving all three CPs is higher. As a matter of fact, in the grand coalition the joint cost rate of CP_1 and CP_2 amounts to $7.89 + 3.97 = 11.86$ $/hour, while in the sub-coalition case it amounts to 8.08 $/hour. This means that CP_1 and CP_2 have an incentive to leave the grand coalition (in case it has been formed) and to form a federation including only them or, said in game-theoretic words, the grand coalition is *unstable* (we discuss this concept in Section 3).

This example shows that a more sophisticated solution than simply forming the grand coalition is necessary in order to ensure stability and that, as a general rule, the resulting federation set may include several federations.

3. THE COOPERATIVE CP GAME

As illustrated in the previous section, a CP must consider various factors before deciding whether to join or not a federation. Among them, the most important ones are:

- *Stability*: a federation is *stable* if none of its participants finds that it is more profitable to leave it (e.g., to stay alone or to join another federation) rather than cooperating with the other ones.

- *Fairness*: when joining a federation, a CP expects that the resulting profits are fairly divided among participants. As unfair division leads to instability, it is necessary to design an allocation method that ensures fairness.

In this section, we model the problem of coalition formation as a *coalition formation cooperative game with transferable utility* [32, 34], where each CP cooperates with the other ones in order to maximize its net profit rate, and we present a distributed algorithm for solving this problem.

3.1 Characterization

Given a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of CPs (henceforth also referred to as the *players*), a *coalition* $\mathcal{S} \subseteq \mathcal{N}$ represents an agreement among the CPs in $\mathcal{S}$ to act as a single entity (i.e., $\mathcal{S}$ is a federation of CPs). Specifically, in this paper, a coalition $\mathcal{S}$ implies that the CPs belonging to $\mathcal{S}$ perform a global allocation of their joint workload $\mathcal{J}_{\mathcal{S}}$ by using as host set the union $\mathcal{H}_{\mathcal{S}}$ of their host sets. In other words, the CPs of the coalition act as a single CP with a load that is the composition of the loads and with a host set that is a composition of the host sets of the coalition.

A coalition $\mathcal{S}$ is associated with a *revenue rate* $r(\mathcal{J}_{\mathcal{S}})$ and with an *energy cost rate* $e(\mathcal{J}_{\mathcal{S}}, \mathcal{H}_{\mathcal{S}})$. The revenue rate of $\mathcal{S}$ is simply the sum of revenue rates of individual VMs $j \in \mathcal{J}_{\mathcal{S}}$, while $e(\mathcal{J}_{\mathcal{S}}, \mathcal{H}_{\mathcal{S}})$ can be derived by minimizing the energy cost resulting from the allocation of the workload $\mathcal{J}_{\mathcal{S}}$ on the host set $\mathcal{H}_{\mathcal{S}}$ (we discuss this in Section 3.3).

We model the system under study by using the most common form of cooperative games, i.e., the *characteristic form* [32], where the value of a coalition $\mathcal{S}$ depends solely on the members of that coalition, with no dependence on how the players in $\mathcal{N} \setminus \mathcal{S}$ are structured (where $\mathcal{N} \setminus \mathcal{S}$ denotes the set difference).

Given the above definitions, for the system under study we define the *coalition value* $v(\cdot)$ as:

$$v(\mathcal{S}) = \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{J}_i} r_i\big(q(j)\big) - e\big(\mathcal{J}_\mathcal{S}, \mathcal{H}_\mathcal{S}\big) \tag{1}$$

where $q(j)$ is the class of VM j (see Section 2.1).

The value $v\big(\mathcal{S}\big)$ must be divided, according to a given rule, among the participants to $\mathcal{S}$. The share $x_i(\mathcal{S})$ of $v\big(\mathcal{S}\big)$ received by CP i is its *payoff*, and the vector $\boldsymbol{x}(\mathcal{S}) \in I\!R^{|\mathcal{N}|}$, with each component $x_i(\mathcal{S})$ being the payoff of CP $i \in \mathcal{S}$, is the *payoff allocation.*

In our cooperative game, CPs seek to form coalitions in order to increase their payoffs. As discussed at the beginning of this section, payoffs should be fairly allocated so that stable coalitions can form.

In order to ensure fairness in the division of payoffs, we use the *Shapley value* [42], a solution method that is based on the concept of *marginal contribution.* [2]

In particular, the Shapley value of player i can be defined as:

$$\phi_i(v) = \sum_{\mathcal{S}\subseteq\mathcal{N}\setminus\{i\}} \frac{|\mathcal{S}|!\big(n-|\mathcal{S}|-1\big)!}{n!}\Big(v\big(\mathcal{S}\cup\{i\}\big) - v\big(\mathcal{S}\big)\Big) \tag{2}$$

where the sum is over all subsets $\mathcal{S}$ not containing i.

It is important to note that since the Shapley value for a player is based on the concept of the player's *marginal contribution* to a coalition (i.e., the change in the worth of a coalition when the player joins to that coalition), the larger is the contribution provided by a player to the coalition, the higher is the payoff allocated to it. This means that, in a given CP federation, some "more-contributing" CP will be rewarded by other "less-contributing" CPs to be enrolled in the federation.

Let us now discuss the stability. In game theory, a typical way to guarantee stability is to ensure that the allocation of payoffs falls in the so called *core* [32], that can be intuitively defined as the set of payoff allocations that guarantees that no group of players has an incentive to leave the coalition $\mathcal{N}$ to form another coalition $\mathcal{S} \subset \mathcal{N}$. It can be shown that a game with a non-empty core contains allocations that can be voluntarily agreed by all players and are thereby stable, while in a game with an empty core, some players (or groups of players) are better off when acting alone than when cooperating all together (the grand coalition $\mathcal{N}$).

Unfortunately, it can be shown that the core of cooperative CP game defined above *can be empty* (see Section A for a formal proof of this statement).

To illustrate the effects of a game with an empty core, we return to the *Scenario* 2 example of Section 2.2, and we compute the coalition values and payoffs corresponding to all the federations that can be formed by the three involved CPs. From these values, that are tabulated in Table 4, we note that for the grand coalition we have that $\phi_1(v) + \phi_2(v) = 11.79$, while for the smaller coalition $\{1, 2\}$ it results that $v\big(\{1, 2\}\big) = 12.08$. That is, for CP$_1$ and CP$_2$, the coalition $\{1, 2\}$ is more convenient than the grand coalition, or, in other words, the grand coalition is unstable. More formally, we can prove that, for the *Scenario* 2, the core is empty by simply solving the optimization problem shown in Figure 1, which directly derives from the definition of the core (e.g., see [32]), to find one of the imputations (if any) inside the core. This optimization problem is infeasible (i.e., it does not exist any payoff vector (x_0, x_1, x_2) that satisfies the core conditions) and hence the core is empty. These results thus confirm the intuition provided in Section 2.2, i.e., that in some situations the grand coalition may lead to instability.

It is important to note that, in general, the emptiness of the core does not depend on the particular payoff allocation strategy, but it is instead a peculiarity of the game. In order to solve this problem, we have thus to resort to a specific type of cooperative game, the so called *coalition formation cooperative game* (see [7, 16, 34]) that, as shown later, achieves stability by forming independent and disjoint smaller coalitions when the grand coalition does not form (as in the *Scenario* 2 case

$$\begin{aligned}
&\text{maximize } z = x_0 + x_1 + x_2\\
&\text{subject to}\\
&\qquad x_0 \geq 6.21,\\
&\qquad x_1 \geq 4.15,\\
&\qquad x_2 \geq 4.15,\\
&\qquad x_0 + x_1 \geq 12.08,\\
&\qquad x_0 + x_2 \geq 12.08,\\
&\qquad x_1 + x_2 \geq 8.49,\\
&\qquad x_0 + x_1 + x_2 = 16.27,\\
&\qquad x_0 \geq 0,\\
&\qquad x_1 \geq 0,\\
&\qquad x_2 \geq 0.
\end{aligned}$$

Figure 1: The optimization model to test the emptiness of the core for the *Scenario* 2

Table 4: Scenario 2: Coalition values and payoffs

Coalition	$v(\cdot)$	$\phi_i(v)$
$\{1\}$	6.21	$\{6.21\}$
$\{2\}$	4.15	$\{4.15\}$
$\{3\}$	4.15	$\{4.15\}$
$\{1,2\}$	12.08	$\{7.07, 5.01\}$
$\{1,3\}$	12.08	$\{7.07, 5.01\}$
$\{2,3\}$	8.49	$\{4.25, 4.25\}$
$\{1,2,3\}$	16.27	$\{7.31, 4.48, 4.48\}$

[2]More specifically, we use the *Aumann-Dréze* value [8], which is an extension of the Shapley value for games with coalition structures.

discussed above).

More specifically, we consider a class of coalition formation games known as *hedonic games* [12, 16]. Hedonic games can be seen as special cases of cooperative games where a player's preferences over coalitions depend only on the composition of his coalition. That is, players prefer being in one coalition rather than in another one purely based on who else is in the coalitions they belong.

Let us reformulate our cooperative CP game as a hedonic game. Given the set $\mathcal{N} = \{1, 2, \ldots, n\}$ of players (i.e., CPs), we define a *coalition partition* as the set $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_l\}$ that partitions the CPs' set $\mathcal{N}$. That is, for $k = 1, \ldots, l$, each $\mathcal{S}_k \subseteq \mathcal{N}$ is a disjoint coalition such that $\bigcup_{k=1}^{l} \mathcal{S}_k = \mathcal{N}$ and $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset$ for $j \neq k$. Given a coalition partition Π, for any CP $i \in \mathcal{N}$, we denote by $\mathcal{S}_\Pi(i)$ the coalition $\mathcal{S}_k \in \Pi$ such that $i \in \mathcal{S}_k$.

To set up the coalition formation process, we need to define a *preference relation* so that each CP can order and compare all the possible coalitions it belongs and hence it can build preferences over them. Formally, for any CP $i \in \mathcal{N}$, a *preference relation* $\succeq_i$ is defined as a complete, reflexive, and transitive binary relation over the set of all coalitions that CP i can form (see [12]). Specifically, for any CP $i \in \mathcal{N}$ and given $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{N}$, the notation $\mathcal{S}_1 \succeq_i \mathcal{S}_2$ means that CP i prefers being a member of $\mathcal{S}_1$ over $\mathcal{S}_2$ or at least i prefers both coalitions equally. The strict counterpart of $\succeq_i$ is denoted by $\succ_i$ and implies that i strictly prefers being a member of $\mathcal{S}_1$ over $\mathcal{S}_2$. Note that the definition of a preference relation is one of the peculiarities of the coalition formation process. In general, this relation can be a function of several parameters, such as the payoffs that the players receive from each coalition, the approval of the coalition members, and the players' history, just to name a few.

In our coalition formation CP game, for any CP $i \in \mathcal{N}$, we use the following preference relation:

$$\mathcal{S}_1 \succeq_i \mathcal{S}_2 \iff f_i(\mathcal{S}_1) \geq f_i(\mathcal{S}_2) \tag{4}$$

where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{N}$ are two coalitions containing CP i, and $f_i(\cdot)$ is a preference function, defined for any CP $i \in \mathcal{N}$ and any coalition $\mathcal{S}$ containing i, such that:

$$f_i(\mathcal{S}) = \begin{cases} x_i(\mathcal{S}), & \text{if } \mathcal{S} \notin h(i), \\ -\infty, & \text{otherwise.} \end{cases} \tag{5}$$

where $x_i(\mathcal{S})$ is the payoff received by CP i in $\mathcal{S}$, and $h(i)$ is a history set where CP i stores the identity of the coalition that it visited and left in the past. The rationale behind the use of $h(\cdot)$ is to avoid that a CP visit the same set of coalitions twice (a similar idea has also been used in previously published work, such as in [39, 40]). Thus, according to Eq. (5), each CP prefers to join to the coalition that provides the larger payoff, unless it has already been visited and left in the past. The strictly counterpart $\succ_i$ of $\succeq_i$ is defined by replacing $\geq$ with $>$ in Eq. (4).

Step 0: Initialization.
At time $t = 0$, the CPs are partitioned as:

$$\Pi_0 = \Big\{\{1\}, \{2\}, \ldots, \{n\}\Big\},$$
$$h(i) = \emptyset, \quad \forall i \in \mathcal{N}.$$

Step 1: Coalition Formation Stage I.
Given the current coalition partition Π_c, each CP i investigates possible hedonic shift operations, in order to look for a coalition $\mathcal{S}_k \in \Pi_c \cup \emptyset$ (if any) such that:

$$\mathcal{S}_k \cup \{i\} \succ_i \mathcal{S}_{\Pi_c}(i).$$

Step 2: Coalition Formation Stage II.
If such coalition $\mathcal{S}_k$ is found, CP i decides to perform the hedonic shift rule to move to $\mathcal{S}_k$:

1. CP i updates its history $h(i)$ by adding $\mathcal{S}_{\Pi_c}(i)$.
2. CP i leaves its current coalition $\mathcal{S}_{\Pi_c}(i)$ and joins the new coalition $\mathcal{S}_k$.
3. Π_c is updated:

$$\Pi_{c+1} = \Big(\Pi_c \setminus \{\mathcal{S}_{\Pi_c}(i), \mathcal{S}_k\}\Big) \cup \Big\{\mathcal{S}_{\Pi_c}(i) \setminus \{i\}, \mathcal{S}_k \cup \{i\}\Big\}.$$

Otherwise, CP i remains in the same coalition so that:

$$\Pi_{c+1} = \Pi_c$$

Step 3: Coalition Formation Stage III.
Repeat Step 1 and Step 2 until all CPs converge to a final partition Π_f.

Figure 2: The Distributed Coalition Formation Algorithm

3.2 A Distributed Coalition Formation Algorithm

We are now ready to define our distributed algorithm for coalition formation that allows each player to decide in a *selfish* way to which coalitions to join at any point in time.

This algorithm is based on the following *hedonic shift rule* (see [38]): given a coalition partition $\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$ on the set $\mathcal{N}$ and a preference relation $\succ_i$, any CP $i \in \mathcal{N}$ decides to leave its current coalition $\mathcal{S}_\Pi(i)$ and join another coalition $\mathcal{S}_k \in \Pi \cup \emptyset$ (with $\mathcal{S}_k \neq \mathcal{S}_\Pi(i)$) if and only if $\mathcal{S}_k \cup \{i\} \succ_i \mathcal{S}_\Pi(i)$. The shift rule can be seen as a selfish decision made by a CP to move from its current coalition to a new one, *regardless of the effects of this move on the other CPs.*

Whenever a CP i applies this rule, it updates its history set $h(i)$ to store the coalition $\mathcal{S}_\Pi(i)$ it is leaving. After the rule is applied, the partition Π changes into a new partition Π', such that:

$$\Pi' = \Big(\Pi \setminus \{\mathcal{S}_\Pi(i), \mathcal{S}_k\}\Big) \cup \Big\{\mathcal{S}_\Pi(i) \setminus \{i\}, \mathcal{S}_k \cup \{i\}\Big\} \tag{6}$$

Using the hedonic shift rule and the preference relation defined in Eq. (4) and Eq. (5), we construct a distributed coalition formation algorithm, shown in Figure 2.

To implement the proposed algorithm in a real environment, a suitable approach must be taken. For a centralized approach, CPs can rely to a central coordinator, to which CPs communicate their decisions and from which CPs obtain information to update their local state. For what regards a distributed approach, suitable techniques for neighbor discovering, communication and synchronization must be used. To this end, well-known algorithms exist in distributed and multi-agent systems literature (e.g., see [15, 46]).

It is worth noting that the presented algorithm can be executed at specific instants of time or when new VM requests arrive to CPs, thus making our coalition formation mechanism able to adapt to environmental changes.

We now prove that our algorithm always converges to a stable partition.

Proposition 1. *Starting from any initial coalition structure* Π_0*, the proposed algorithm always converges to a final partition* Π_f*.*

Proof. The coalition formation phase can be mapped to a sequence of shift operations. That is, according to the hedonic shift rule, every shift operation transforms the current partition Π_c into another partition Π_{c+1}. Thus, starting from the initial step, the algorithms yields the following transformations:

$$\Pi_0 \rightarrow \Pi_1 \rightarrow \cdots \rightarrow \Pi_c \rightarrow \Pi_{c+1} \qquad (7)$$

where the symbol $\rightarrow$ denotes the application of a shift operation. Every application of the shift rule generates two possible cases: (a) $\mathcal{S}_k \neq \emptyset$, so it leads to a new coalition partition, or (b) $\mathcal{S}_k = \emptyset$, so it yields a previously visited coalition partition with a non-cooperatively CP (i.e., with a coalition of size 1). In the first case, the number of transformations performed by the shift rule is finite (at most, it is equal to the number of partitions, that is the Bell number; see [37]), and hence the sequence in Eq. (7) will always terminate and converge to a final partition Π_f. In the second case, starting from the previously visited partition, at certain point in time, the non-cooperative CP must either join a new coalition and yield a new partition, or decide to remain non-cooperative. From this, it follows that the number of re-visited partitions will be limited, and thus, in all the cases the coalition formation stage of the algorithm will converge to a final partition Π_f. □

We address the stability of the final partition Π_f by using the concept of *Nash-stability* (see [12]). Intuitively, a partition Π is considered Nash-stable if no CP has incentive to move from its current coalition $\mathcal{S}_\Pi(i)$ to join a different coalition of Π, or to act alone. More formally, a partition $\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$ is *Nash-stable* if $\forall i \in \mathcal{N}$, $\mathcal{S}_\Pi(i) \succeq_i \mathcal{S}_k \cup \{i\}$ for all $\mathcal{S}_k \in \Pi \cup \emptyset$.

Let us show that the partition to which our algorithm converges is Nash-stable.

Proposition 2. *Any final partition* Π_f *resulting from the algorithm presented in Figure 2 is Nash-stable.*

Proof. To show this, we use the proof by contradiction technique. Assume that the final partition Π_f is not Nash-stable. Consequently, there exists a CP $i \in \mathcal{N}$ and a coalition $\mathcal{S}_k \in \Pi_f \cup \emptyset$ such that $\mathcal{S}_k \cup \{i\} \succ_i \mathcal{S}_{\Pi_f}(i)$. Then, CP i will perform a hedonic shift operation and hence $\Pi_f \rightarrow \Pi'_f$. This contradicts the assumption that Π_f is the final outcome of our algorithm. □

The Nash-stability also implies the so called *individual-stability* (see [12]). A partition $\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$ is *individually-stable* if do not exist a player $i \in \mathcal{N}$ and a coalition $\mathcal{S}_k \in \Pi \cup \emptyset$ such that $\mathcal{S}_k \cup \{i\} \succ_i \mathcal{S}_\Pi(i)$ and $\mathcal{S}_k \cup \{i\} \succeq_j \mathcal{S}_k$ for all $j \in \mathcal{S}_k$.

It is worth noting that Nash-stability only captures the notion of stability with respect to movements of single CPs (i.e., no CP has an incentive to unilaterally deviate). However, it does not guarantee the stability with respect to other aspects that are instead captured by other stability concepts. For instance, the stability with respect to movements of groups of CPs is captured by the *core*-stability (see [12]), whereby no group of CPs can collectively defect and form a new coalition where each of them is better off. Unfortunately, the two stability concepts are not related each other, in general (i.e., one stability concept does not necessarily imply the other one). Moreover, there exist other stronger stability concepts but unfortunately there is not warranty that a satisfying partition does exist (e.g., see [9]) and the check for the existence is computationally hard (e.g., see [10]). Finally, Nash-stability does not guarantee the maximization of the overall net profit (e.g., the *social optimum* in the game-theoretic jargon) [22]. Despite all of that, Nash-stability is generally considered a reasonable trade-off.

Thus, we can conclude that our algorithm always converges to a partition Π_f which is both Nash-stable and individually stable.

3.3 Computation of the Coalition Value

To use the game-theoretic model discussed in the previous section, we need a way to find (for a given coalition) the optimal workload allocation (i.e., the allocation that minimizes the energy cost), that allows us to compute the coalition value.

To this end, we define a *Mixed Integer Linear Program* (MILP) modeling the problem of allocating a set $\mathcal{J}_\mathcal{S}$ of VMs onto a set $\mathcal{H}_\mathcal{S}$ of hosts so that the hourly energy cost is minimized.

We base our MILP on the model described in [13], that has been first revised to improve its computational performance and then extended in order to incorporate the heterogeneity of physical resources and the energy cost.

The resulting optimization model is shown in Figure 3, where we use the same notation introduced in Section 2.1,[3] and we denote with $o(i)$ the function that is 1 if host i is powered on and 0 otherwise (i.e., a function $o : \mathcal{H} \to \{0, 1\}$), with L_k and S_k the power (in W) consumed during the switch-on and switch-off operations of a host of class k, respectively, with $G_{i_1,i_2,v}$ the hourly cost (in \$/hour) to migrate a VM of class v from CP i_1 to CP i_2, with E_i the hourly cost (in \$/Wh) of the energy consumed by a host belonging to CP i, with $c(i)$ the function that gives the CP that owns host i (i.e, a function $c : \mathcal{H} \to \mathcal{N}$), and with $h(j)$ the function that gives the host where VM j is allocated (i.e., a function $h : \mathcal{J} \to \mathcal{H}$).

In the optimization model we define, for any VM $j \in \mathcal{J}$ and host $i \in \mathcal{H}$, the following decision variables: b_{ji} is a binary variable that is equal to 1 if VM j is allocated to host i; α_i is a real variable representing the overall fraction of CPU assigned to all VMs allocated on host i; p_i is a binary variable that is equal to 1 if host i is powered on. The objective function $e(\mathcal{J}, \mathcal{H})$ (hereafter, e for short) represents, for a specific assignment of decision variables, the hourly energy cost (in \$/hour) due to the power consumption induced by the federation of CPs to host the given VMs.

The resulting VM allocation is bound to the following constraints:

- Eq. (8b) imposes that each VM is hosted by exactly one host;
- Eq. (8c) states that only hosts that are switched on can have VMs allocated to them; the purpose of these constraints is to avoid that a VM is allocated to a host that will be powered off;
- Eq. (8d) ensures that (1) the CPU resource of a powered-on host is not exceeded, and (2) that no CPU resource is consumed on a host that will be powered off;
- Eq. (8e) assures that (1) the RAM resource of a powered-on host is not exceeded, (2) that VMs hosted on that host receive their required amount of RAM, and (3) that no RAM resource is consumed on a host that will be powered off;
- Eq. (8f) states that all VMs must exactly obtain the amount of CPU resource they require;

[3]To ease readability, we simplify it by denoting with $\mathcal{J}$ and $\mathcal{H}$ the cloud workload and the host set, respectively (i.e., we omit the dependence by $\mathcal{S}$).

$$\begin{aligned}
\text{minimize } e = \sum_{i \in \mathcal{H}} \Big[& p_i C^{\min}_{g(i)} + \alpha_i (C^{\max}_{g(i)} - C^{\min}_{g(i)}) \\
& + p_i(1 - o(i)) L_{g(i)} + (1 - p_i) o(i) S_{g(i)} \Big] E_{c(i)} \\
& + \sum_{j \in \mathcal{J}} b_{ji} G_{c(h(j)),c(i),q(j)} \qquad \text{(8a)}
\end{aligned}$$

subject to

$$\begin{aligned}
& \sum_{i \in \mathcal{H}} b_{ji} = 1, && j \in \mathcal{J}, && \text{(8b)}\\
& \sum_{j \in \mathcal{J}} b_{ji} \le |\mathcal{J}| p_i, && i \in \mathcal{H}, && \text{(8c)}\\
& \alpha_i \le p_i, && i \in \mathcal{H}, && \text{(8d)}\\
& \sum_{j \in \mathcal{J}} b_{ji} M_{q(j)g(i)} \le p_i, && i \in \mathcal{H}, && \text{(8e)}\\
& \sum_{j \in \mathcal{J}} b_{ji} A_{q(j)g(i)} = \alpha_i, && i \in \mathcal{H}, && \text{(8f)}\\
& b_{ji} \in \{0, 1\}, && j \in \mathcal{J}, i \in \mathcal{H}, && \text{(8g)}\\
& \alpha_i \in [0, 1], && i \in \mathcal{H}, && \text{(8h)}\\
& p_i \in \{0, 1\}, && i \in \mathcal{H}. && \text{(8i)}
\end{aligned}$$

Figure 3: The VM allocation optimization model

- Eq. (8g), Eq. (8h), and Eq. (8i) define the domain of decision variables b_{ji}, α_i, and p_i, respectively.

As in [13], in order to keep into considerations QoS requirements related to each class of VMs, we assumed that each VM j exactly obtains the amount of CPU $CPU_{q(j)}$ and RAM $RAM_{q(j)}$ as defined by its class $q(j)$ (see Section 2.1).

4. EXPERIMENTAL EVALUATION

To illustrate the effectiveness of our algorithm for coalition formation, we perform a set of experiments in which we compute the federation set for various scenarios including a population of distinct CPs.

In all scenarios, we consider 4 CPs, whose infrastructures are characterized as reported in Table 5, and we use the same host classes, VM classes and VM shares as the ones defined in Table 1. We also assume that all CPs use the same revenue rate policy, that is they earn 0.08 \$/hour for class-1 VMs, 0.16 \$/hour for class-2 VMs, and 0.32 \$/hour for class-3 VMs. Furthermore,

Table 5: Experimental evaluation – Configuration of CPs

CP	# Hosts		
	Class-1	Class-2	Class-3
CP_1	40	0	0
CP_2	0	40	0
CP_3	0	0	40
CP_4	15	15	10

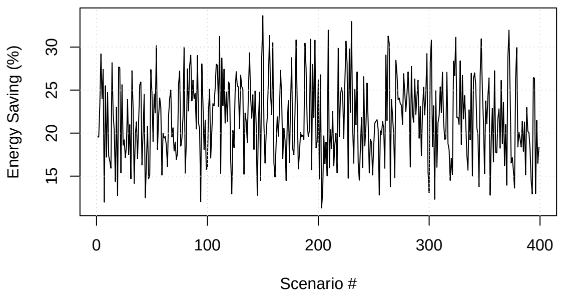
without loss of generality, we also assume that the electricity price is the same for all CPs and it is equal to 0.4 \$/kWh.

Starting from this configuration, we set up 400 scenarios that differ from each other in the workload of the various CPs, in the power state of each host and in the VM migration costs. Specifically, in each scenario the workload of each CP is set by randomly generating the number of VMs of each class as an integer number uniformly distributed in the $[0, 20]$ interval. In addition, to provide values to function $o(\cdot)$, we randomly generate the power state (i.e., ON or OFF) of each host according to a Bernoulli distribution with parameter 0.5. Furthermore, the values for L_k and S_k, for each host class k, are computed as the product of the electricity price, the maximum power consumption and the time taken to complete the switch-on or switch-off operation. This switch-on/-off time is randomly generated for each host class according to a Normal distribution with mean of 300 μsec and standard deviation (S.D.) of 50 μsec (e.g., see [27]). Finally, the VM migration costs $G_{c_1,c_2,k}$ from CP c_1 to CP c_2 for each VM class k are computed as the product between the data transfer cost rate, the data size to transfer and the time to migrate a VM of class k from CP c_1 to CP c_2, and assuming that our algorithm activates every 12 hours. The data transfer cost rate is taken from the Amazon EC2 data transfer pricing [1] and set to 0.001 \$/GB. Furthermore, we suppose that data are persistently transferred during the migration time at a fixed data rate of 100 Mbit/sec for all CPs. For what concerns the migration time, we assume that it is randomly generated according to a Normal distribution with mean of 277 sec and S.D. of 182 sec for VMs of class 1, with mean of 554 sec and S.D of 364 sec for VMs of class 2, and with mean of 1108 sec and S.D. of 728 sec for VMs of class 3 (e.g., see [5]). The migration cost between hosts of the same CP is assumed to be negligible.

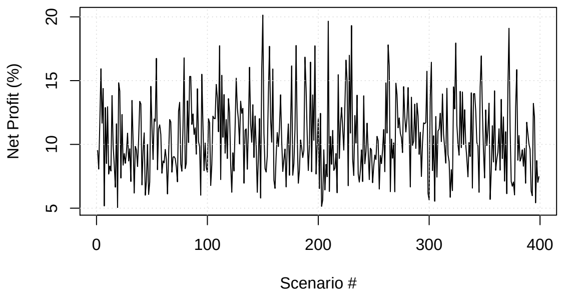
For each one the above scenario, we compute the federation set of the involved CPs by using an ad-hoc simulator written in `C++` and interfaced with CPLEX [3] to solve the various instances of the optimization model of Section 3.3.

In the rest of this section, we first present a summary of the performance obtained by our algorithm over all scenarios, and then, we illustrate its behavior by showing its run trace for one of these scenarios.

In Figure 4, we compare the performance of each scenario in terms of energy saving and net profit obtained with our algorithm with respect to the *no-federation* case (i.e., when CPs work in isolation). Specifically, the figures show, for each scenario, the percentage of the reduction of energy consumption (see Figure 4a) and of the increment of net profit (see Figure 4b) that all CPs obtain when they federate according to our algorithm



(a) Reduction of energy consumption (%)



(b) Increase of net profit (%)

Figure 4: Performance of our algorithm with respect to the no-federation case

with respect to the case of working individually. As can be seen from the figures, our algorithm, with respect to always work non-cooperatively, allows the CPs to reduce the overall consumed energy from 11.3% to 33.6% (with an average of 21.6%), and to increment the overall net profit from 5.1% to 20.1% (with an average of 10.5%).

We can also analyze the benefits provided by our algorithm from the point of view of each CP. Results from our experiments show that, from the CP perspective, the formation of federations yielded by our algorithm is always non-detrimental. Specifically, it results that, on average, the net profit earned by CP_1, CP_2, CP_3 and CP_4 increases by nearly 18.0%, 8.5%, 22.8% and 4.3% with respect to the no-federation case, respectively.

Finally, to illustrate how our algorithm works, we present the run trace for a single scenario, whose characteristics are reported in Table 6. [4] We select this scenario to illustrate the behavior of the algorithm when there are multiple Nash-stable partitions. [5]

For this investigation, we show in Table 7 all possible partitions together with the value function $v(\cdot)$ of every coalition inside each partition, and the corresponding Shapley values. [6] From the table, we can see that there are two Nash-stable coalitions, namely $\{1, 2, 3, 4\}$ and $\{\{1, 3\}, \{2, 4\}\}$. To arrive to one of these partitions,

[4]Due to lack of space, we only report the number of VMs.
[5]Note, our algorithm's output is always a single partition.
[6]Note, our algorithm does not necessarily enumerate all of such partitions.

Table 6: Experimental results – Workload of CPs in the case study

CP	# VMs		
	Class-1	Class-2	Class-3
CP_1	0	12	13
CP_2	18	5	11
CP_3	17	18	11
CP_4	3	2	0

our algorithm works as follows. Starting from partition $\{\{1\},\{2\},\{3\},\{4\}\}$ (i.e., every CP works individually), there are two different sequences of hedonic shift rules:

- Sequence #1: $\{\{1\},\{2\},\{3\},\{4\}\} \xrightarrow{3} \{\{1,3\},\{2\},\{4\}\} \xrightarrow{2} \{\{1,2,3\},\{4\}\} \xrightarrow{4} \{\{1,2,3,4\}\}$
- Sequence #2: $\{\{1\},\{2\},\{3\},\{4\}\} \xrightarrow{3} \{\{1,3\},\{2\},\{4\}\} \xrightarrow{2} \{\{1,3\},\{2,4\}\}$

where the index on top of each arrow denotes the CP that performs the corresponding hedonic shift rule.

Regardless what partition is finally selected, from the third column of Table 7 we can also observe that, for this scenario, the partition value improvement for both Nash-stable partitions with respect to the non-cooperative behavior (i.e., partition $\{\{1\},\{2\},\{3\},\{4\}\}$) is about 10% for partition $\{\{1,3\},\{2,4\}\}$ and nearly 17% for the grand-coalition.

5. RELATED WORKS

Recently, the concept of cloud federations [36, 29] has been proposed as a way to provide individual CPs with more flexibility when allocating on-demand workloads. Existing work on cloud federations has been mainly focused on the development of architectural models for federations [18], and of mechanisms providing specific functionalities (e.g., workload management [31, 25], accounting and billing [17], and pricing [23, 24, 28, 43]).

To the best of our knowledge, very little work has been carried out to jointly tackle the problem of dynamically forming stable cloud federations for energy-aware resource provisioning. Indeed, much of the existing work only focuses on a single aspect of the problem. In [19], the design and implementation of a VM scheduler for a federation of CPs is presented. The scheduler, in addition to manage resources that are local to each CP, is able to decide when to rent resources from other CPs, when to lease own idle resources to other CPs, and when to turn on or off local physical resources. Unlike our work, this work does not consider the problem of forming stable CP federations. In [30], a cooperative game-theoretic model for federation formation and VM management is proposed. In this work, the federation formation among CPs is analyzed using the concept of network games, but the energy minimization problem is not considered.

In [26], a profit-maximizing game-based mechanism to enable dynamic cloud federation formation is proposed. The dynamic federation formation problem is modeled as a hedonic game (like our approach), and the federations are computed by means of a merge-split algorithm. There are several important differences between this and our works: (1) we focus on the stability of individuals rather than of groups, (2) we propose a decentralized algorithm, (3) we demonstrate the stability of the obtained federations, and (4) we use the Shapley value instead of the normalized Banzhaf value (as in [26]), since the latter does not satisfy some important properties [44].

In [41], the problem of sharing unused capacity in a federation of CPs for VM spot market is formulated as a non-cooperative repeated game. Specifically, by using a Markov model to predict future non-spot workload, the authors introduce a set of capacity sharing strategies that maximize the federation's long-term revenue and propose a dynamic programming algorithm to find the allocation rules needed to achieve it. Our work can complement this approach by providing a solution to the formation of CP federations for non-spot VM instances.

6. CONCLUSIONS AND FUTURE WORKS

This paper investigates a novel dynamic federation scheme among a set of CPs. To this end, we propose a cooperative game-theoretic framework to study the federation formation problem, and a mathematical optimization model to allocate CP workload in an energy-aware fashion, in order to reduce CP energy costs.

In the proposed scheme, we model the cooperation among the CPs as a coalition game with transferable utility and we devise a distributed hedonic shift algorithm for coalition formation. With the proposed algorithm, each CP individually decides whether to leave the current coalition to join a different one according to his preference, meanwhile improving the perceived net profit. Furthermore, we prove that the proposed algorithm converges to a Nash-stable partition which determines the resulting coalition structure. Numerical results show the effectiveness of our approach.

The future developments of this research is following several directions. First of all, we would like to enhance the coalition value function in order to account for possible request losses due to lack of physical resources. Furthermore, we want to improve the game-theoretic and optimization models in order to include costs in terms of loss of revenues as well as other aspects like the ones related to trustworthiness among CPs.

As a second direction, we plan to integrate the long-term resource provisioning solution proposed in this paper with other short-term and medium-term resource management strategies (e.g., [6, 21]) to improve resource utilization and meet application-level performance re-

Table 7: Experimental results – Coalition values and Shapley values for all the 15 different partitions of the case study

$\Pi = \{\mathcal{S}_1, \ldots, \mathcal{S}_l\}$	$\Big\{v(\mathcal{S}_1), \ldots, v(\mathcal{S}_l)\Big\}$	$\sum_{\mathcal{S}_i \in \Pi} v(\mathcal{S}_i)$	$\{\phi_{\mathcal{S}_1}, \ldots, \phi_{\mathcal{S}_l}\}$
$\Big\{\{1\},\{2\},\{3\},\{4\}\Big\}$	$\{4.28, 3.45, 3.84, 0.38\}$	11.95	$\Big\{\{4.28\},\{3.45\},\{3.84\},\{0.38\}\Big\}$
$\Big\{\{1,2\},\{3\},\{4\}\Big\}$	$\{8.22, 3.84, 0.38\}$	12.44	$\Big\{\{4.52, 3.70\},\{3.84\},\{0.38\}\Big\}$
$\Big\{\{1,3\},\{2\},\{4\}\Big\}$	$\{9.59, 3.45, 0.38\}$	13.42	$\Big\{\{5.01, 4.57\},\{3.45\},\{0.38\}\Big\}$
$\Big\{\{1\},\{2,3\},\{4\}\Big\}$	$\{4.28, 7.82, 0.38\}$	12.48	$\Big\{\{4.28\},\{3.72, 4.10\},\{0.38\}\Big\}$
$\Big\{\{1,4\},\{2\},\{3\}\Big\}$	$\{4.69, 3.45, 3.84\}$	11.98	$\Big\{\{4.29, 0.40\},\{3.45\},\{3.84\}\Big\}$
$\Big\{\{1\},\{2,4\},\{3\}\Big\}$	$\{4.28, 4.33, 3.84\}$	12.45	$\Big\{\{4.28\},\{3.70, 0.63\},\{3.84\}\Big\}$
$\Big\{\{1\},\{2\},\{3,4\}\Big\}$	$\{4.28, 3.45, 5.43\}$	13.16	$\Big\{\{4.28\},\{3.45\},\{4.44, 0.99\}\Big\}$
$\Big\{\{1,2,3\},\{4\}\Big\}$	$\{13.27, 0.38\}$	13.65	$\Big\{\{5.00, 3.70, 4.57\},\{0.38\}\Big\}$
$\Big\{\{1,2,4\},\{3\}\Big\}$	$\{8.69, 3.84\}$	12.53	$\Big\{\{4.39, 3.80, 0.50\},\{3.84\}\Big\}$
$\Big\{\{1,2\},\{3,4\}\Big\}$	$\{8.22, 5.43\}$	13.65	$\Big\{\{4.52, 3.70\},\{4.44, 0.99\}\Big\}$
$\Big\{\{1,3,4\},\{2\}\Big\}$	$\{10.01, 3.45\}$	13.46	$\Big\{\{4.63, 4.78, 0.60\},\{3.45\}\Big\}$
$\Big\{\{1,3\},\{2,4\}\Big\}$	$\{9.59, 4.33\}$	13.92	$\Big\{\{5.01, 4.57\},\{3.70, 0.63\}\Big\}$
$\Big\{\{1,4\},\{2,3\}\Big\}$	$\{4.69, 7.82\}$	12.51	$\Big\{\{4.29, 0.40\},\{3.72, 4.10\}\Big\}$
$\Big\{\{1\},\{2,3,4\}\Big\}$	$\{4.28, 8.88\}$	13.16	$\Big\{\{4.28\},\{3.62, 4.36, 0.90\}\Big\}$
$\Big\{1,2,3,4\Big\}$	$\{14.01\}$	14.01	$\Big\{4.78, 3.78, 4.76, 0.68\Big\}$

quirements, and with techniques for incremental VM migration (e.g., [45]).

Finally, we want to implement and validate the proposed algorithm in a real testbed.

7. REFERENCES

[1] Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2`.

[2] GeekBench: Next-generation processor benchmark. `http://www.primatelabs.com/geekbench`.

[3] IBM ILOG CPLEX Optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer`.

[4] VMware Inc. `http://www.vmware.com`.

[5] S. Akoush, R. Sohan, A. Rice, A. Moore, and A. Hopper. Predicting the performance of virtual machine migration. In *Proc. of the MASCOTS*, 2010.

[6] L. Albano, C. Anglano, M. Canonico, and M. Guazzone. Fuzzy-Q&E: achieving QoS guarantees and energy savings for cloud applications with fuzzy control. In *Proc. of the* 3rd *CGC*, 2013.

[7] K. Apt and A. Witzel. A Generic Approach to Coalition Formation. *Int Game Theor Rev*, 11(03):347–367, 2009.

[8] R. Aumann and J. Dréze. Cooperative games with coalition structures. *Int J Game Theor*, 3(4):217–237, 1974.

[9] H. Aziz and F. Brandl. Existence of stability in hedonic coalition formation games. In *Proc. of the* 11th *AAMAS*, pages 763–770, 2012.

[10] H. Aziz, F. Brandt, and H. Seedig. Computing desirable partitions in additively separable hedonic games. *Artif Intell*, 195:316–334, 2013.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. of the* 19th *SOSP*, 2003.

[12] A. Bogomolnaia and M. Jackson. The Stability of Hedonic Coalition Structures. *Game Econ Behav*, 38:201–230, 2002.

[13] D. Borgetto, H. Casanova, G. D. Costa, and J.-M. Pierson. Energy-aware service allocation. *Future Generat Comput Syst*, 8(25):769–779, 2012.

[14] A. Celesti, A. Puliafito, F. Tusa, and M. Villari. Towards energy sustainability in federated and interoperable clouds. In H. Mouftah and B. Kantarci, editors, *Communication Infrastructures for Cloud Computing*. 2013.

[15] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. *Distributed Systems: Concepts and Design*. Addison Wesley, 5th edition, 2011.

[16] J. Drèze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[17] E. Elmroth, F. Marquez, D. Henriksson, and D. Ferrera. Accounting and Billing for Federated Cloud Management. In *Proc. of the* 8th *GCC*, 2009.

[18] A. Ferrer, F. Hernández, J. Tordsson, E. Elmroth,

A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan. OPTIMIS: A Holistic Approach to Cloud Service Provisioning. *Future Generat Comput Syst*, 28(1):66–77, 2012.

[19] Í. Goiri, J. Guitart, and J. Torres. Economic model of a cloud provider operating in a federated cloud. *Inform Syst Front*, 14(4):1–17, 2012.

[20] M. Guazzone, C. Anglano, and M. Canonico. Energy-Efficient Resource Management for Cloud Computing Infrastructures. In *Proc. of the* 3rd *CloudCom*, 2011.

[21] M. Guazzone, C. Anglano, and M. Canonico. Exploiting VM migration for the automated power and performance management of green cloud computing systems. In *Proc. of the* 1st *E2DC*, 2012.

[22] C. Hasan, E. Altman, and J.-M. Gorce. On the nash stability in the hedonic coalition formation games. *IEEE Trans Automat Contr.* Submitted.

[23] M. Hassan, M. S. Hossain, A. J. Sarkar, and E.-N. Huh. Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform. *Inform Syst Front*, pages 1–20, 2012.

[24] J. Künsemöller and H. Karl. A game-theoretical approach to the benefits of cloud computing. In *Proc. of the* 8th *GECON*, 2012.

[25] L. Larsson, D. Henriksson, and E. Elmroth. Scheduling and Monitoring of Internally Structured Services in Cloud Federations. In *Proc. of the* 16th *ISCC*, 2011.

[26] L. Mashayekhy and D. Grosu. A coalitional game-based mechanism for forming cloud federations. In *Proc. of* 5th *UCC*, 2012.

[27] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating server idle power. In *Proc. of the* 14th *ASPLOS*, 2009.

[28] M. Mihailescu and Y.-M. Teo. Dynamic Resource Pricing on Federated Clouds. In *Proc. of the* 10th *CCGrid*, 2010.

[29] R. Moreno-Vozmediano, R. Montero, and I. Llorente. IaaS Cloud Architecture: From Virtualized Data Centers to Federated Cloud Infrastructure. *Computer*, 45(12):65–72, 2012.

[30] D. Niyato, Z. Kun, and P. Wang. Cooperative virtual machine management for multi-organization cloud computing environment. In *Proc. of the* 5th *VALUETOOLS*, 2011.

[31] A. Nordal, A. Kvalnes, J. Hurley, and D. Johansen. Balava: Federating Private and Public Clouds. In *Proc. of the* 7th *SERVICES*, 2011.

[32] B. Peleg and P. Sudhölter. *Introduction to the Theory of Cooperative Games*. Springer Berlin Heidelberg, 2nd edition, 2007.

[33] T. E. S. Program. Report to Congress on server and Data Center energy efficiency. Technical report, U.S. EPA, Aug 2007.

[34] D. Ray. *A Game-Theoretic Perspective on Coalition Formation*. The Lipsey Lectures. Oxford University Press, 2007.

[35] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *Proc. of the HotPower*, 2008.

[36] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Mu noz, and G. Tofetti. RESERVOIR – When one cloud is not enough. *Computer*, 44(2), 2011.

[37] G. Rota. The number of partitions of a set. *Am Math Mon*, 71(5):498–504, 1964.

[38] W. Saad, Z. Han, T. Başar, M. Debbah, and A. Hjørungnes. A selfish approach to coalition formation among unmanned air vehicles in wireless networks. In *Proc. of the* 1st *GameNets*, 2009.

[39] W. Saad, Z. Han, T. Başar, M. Debbah, and A. Hjørungnes. Hedonic coalition formation for distributed task allocation among wireless agents. *IEEE Trans Mobile Comput*, 10(9):1327–1344, 2011.

[40] W. Saad, Z. Han, A. Hjørungnes, D. Niyato, and E. Hossain. Coalition formation games for distributed cooperation among roadside units in vehicular networks. *IEEE J Sel Area Comm*, 29(1):48–60, 2011.

[41] N. Samaan. A novel economic sharing model in a federation of selfish cloud providers. *IEEE Trans Parallel Distr Syst*, To appear.

[42] L. S. Shapley. A Value for n-person Games. In H. Kuhn and A. Tucker, editors, *Contributions to the Theory of Games*, pages 307–317. 1953.

[43] A. Toosi, R. Calheiros, R. Thulasiram, and R. Buyya. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *Proc. of the* 13th *HPCC*, 2011.

[44] R. van den Brink and G. van der Laan. Axiomatizations of the normalized Banzhaf value and the Shapley value. *Soc Choice Welfare*, 15(4):567–582, 1998.

[45] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. In *Proc. of the* 9th *Middleware*, 2008.

[46] G. Weiss, editor. *Multiagent Systems*. MIT Press, 2nd edition, 2013.

Table 9: Values of $v(\cdot)$ for CPs coalitions

Coalitions $\mathcal{S}$	$v(\mathcal{S})$ (\$/hour)
$\{1\}$	0.345
$\{2\}$	0.095
$\{3\}$	0.095
$\{1,2\}$	0.513
$\{1,3\}$	0.513
$\{2,3\}$	0.225
$\{1,2,3\}$	0.623

APPENDIX

A. THE CORE OF THE COOPERATIVE CP GAME CAN BE EMPTY

In this section, we present a more formal proof of the possible emptiness of the core of the cooperative CP game defined in Section 3.1. To do so, we use the proof by construction technique, by providing an instance of the game for which the core is empty.

In cooperative game theory, the *Bondareva-Shapley theorem* provides the necessary and sufficient conditions for the non-emptiness of the core solution concept [32].

Theorem A.1 (Bondareva-Shapley theorem). *Given a cooperative game $\langle \mathcal{N}, v\rangle$, the core of $\langle \mathcal{N}, v\rangle$ is non-empty if and only if for every function $\alpha : 2^{\mathcal{N}} \setminus \{\emptyset\} \to [0,1]$ where*

$$\forall i \in \mathcal{N} : \sum_{\mathcal{S} \in 2^{\mathcal{N}} :\, i \in \mathcal{S}} \alpha(\mathcal{S}) = 1$$

the following condition holds:

$$\sum_{\mathcal{S} \in 2^{\mathcal{N}} \setminus \{\emptyset\}} \alpha(\mathcal{S}) v(\mathcal{S}) \le v(\mathcal{N}). \tag{9}$$

We now show that for the cooperative CP game there exists at least one counterexample that violates the conditions Eq. (9) of the Bondareva-Shapley theorem.

Let us consider a simple scenario consisting of the same host and VM classes as defined in Section 2.2, and of three CPs, whose characteristics are reported in Table 8.

We build the cooperative CP game $\langle \mathcal{N}, v\rangle$ where $\mathcal{N} = \{1,2,3\}$ is the set of CPs and $v(\cdot)$ is the same characteristic function defined in Eq. (1). In Table 9, we show the enumeration of all possible CP coalitions for this game along with their values. To compute the coalition value we use the same revenue rate described in Section 4, that is 0.08 \$/hour for class-1 VMs, 0.16 \$/hour for class-2 VMs, and 0.32 \$/hour for class-3 VMs.

Let us choose as function $\alpha(\cdot)$ in A.1 the following function:

$$\alpha(\mathcal{S}) = \begin{cases} \frac{1}{2}, & \mathcal{S} \in \Big\{\{1,2\},\{1,3\},\{2,3\}\Big\}, \\ 0, & \text{otherwise.} \end{cases}$$

If the core is non-empty, Eq. (9) would hold. However, it results that:

$$v(\{1,2,3\}) < \frac{1}{2} \cdot \Big(v(\{1,2\}) + v(\{1,3\}) + v(\{2,3\}) \Big)$$

That is:

$$\begin{aligned} 0.623 &< \frac{1}{2} \cdot (0.513 + 0.513 + 0.225), \\ 0.623 &< 0.625. \end{aligned}$$

which clearly violates the conditions Eq. (9) of the Bondareva-Shapley theorem and hence the core for this game is empty. □

Table 8: Configuration of CPs

CP		# Hosts			# VMs		Energy Cost
	Class-1	Class-2	Class-3	Class-1	Class-2	Class-3	(\$/kWh)
CP_1	0	2	0	0	4	0	0.4
CP_2	1	0	0	0	1	0	0.4
CP_3	1	0	0	0	1	0	0.4