An Energy-aware Brokering Algorithm to Improve Sustainability in Community Cloud

Maurizio Giacobbe[1], Marco Scarpa[1], Riccardo Di Pietro[2] and Antonio Puliafito[1]
[1]*Department of Engineering, University of Messina, Contrada Di Dio, 98166 Messina, Italy*
[2]*C.I.A.M., University of Messina, S. Pugliatti 1, 98100 Messina, Italy*

Keywords: Cloud Computing, Community Cloud, Energy-aware Brokering, Green Cloud, Low Carbon, Resource Allocation, Sustainability, Virtualization.

Abstract: Cloud computing is a paradigm for large scale distributed infrastructures, platforms or software services which represents a hot topic in Information Technology (IT) recently in both industrial and academic areas. Its use is motivated by the possibility to promote a new economy of scale in different contexts. Along with the well-known public, private and hybrid Cloud models, the Community Cloud is an emerging concept based on a deployment model in which a Cloud infrastructure allows a specific community of consumers to share interests, goals and responsibilities. It can be owned and managed by the community, by a third party, or a combination of them. In such scenario, new low-carbon strategies at Cloud sites are necessary to allow those latter to reduce the consumption in presence of a massive exploitation of IT services. Therefore, balancing performances with both sustainability and cost saving concepts is a challenge. In this paper, we present a low carbon strategy designed to make the best choice in resources allocation, based on sustainability, availability and costs. The proposed energy-aware Brokering Algorithm (eBA) allows to push down carbon dioxide emissions through the Community Cloud ecosystem, by running instances at the most convenient sites.

1 INTRODUCTION

Nowadays, worldwide companies which make business in the *Information and Communication Technology (ICT)* field are increasingly sensitive to the environmental sustainability issue. Their products and services are empowering customers, both people and organizations, to satisfy their requests in several contexts, where improving efficiency and reducing pollution are two essential goals. For example, the *2015 Global 100 Most Sustainable Corporations in the World index* of the *Corporate Knights Magazine* reports *Accenture (Ireland)* is the first in *IT Services* (54th overall position). Meanwhile tha ranking reports *Nokia (Finland), Lenovo Group (China) and EMC (United States)* are the most sustainable companies in *Technology, Hardware, Storage & Peripherals*.

Community Cloud is an emerging topic in ICT. It is a deployment model in which a Cloud infrastructure is built and provisioned in order to be used by a specific community of consumers with shared concerns, goals, and interests (Murugesan and Bojanova, 2016). It can be owned and managed by the community itself, by a third party, or a combination of both. The deployment environment can be provided by a mesh of Cloud providers in order to satisfy the specific requirements and conditions of the community. Cloud providers can be interconnected based on open standards in order to provide a universal decentralized Cloud computing environment.

Our study addresses medium and small size Cloud providers towards solutions allowing them to compete with large Cloud providers in a more sustainable service marketplace. We watch to a dynamic scenario where Cloud providers share their IT resources among their respective Community Cloud sites (i.e., datacenters) in order to reduce costs and energy-efficiency gap if compared with the *top* Cloud computing service providers (e.g., Amazon, Google, Rackspace, etc.). An automated negotiation process facilitates the bilateral negotiation between the **Community Cloud broker** and multiple providers to achieve several objectives for the community members. However, for these purposes, balancing the above objectives with performances is a challenge. To this end, an approach based on Cloud brokering can simplify the procedures in making the best choice. A brokerage scenario is exemplified in Fig. 1: an *Au-*

Giacobbe, M., Scarpa, M., Pietro, R. and Puliafito, A.
An Energy-aware Brokering Algorithm to Improve Sustainability in Community Cloud.
DOI: 10.5220/0006300201660173
In *Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2017)*, pages 166-173
ISBN: 978-989-758-241-7

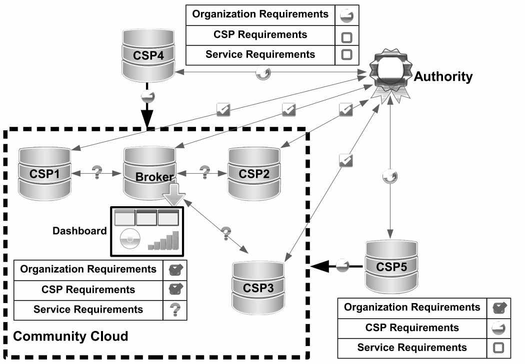

Figure 1: Brokerage in a Communtity Cloud environment.

thority in charge evaluates both the *Organization* and the *Cloud service provider (CSP)* requirements (e.g., SLA) to determine if they are satisfactory to become *Community members*. The same for a candidate broker which consequently is able to evaluate *service requirements* and to rank the offers among the presented by the Community members. For example, *service requirements* can be based on the use of electricity at a Cloud site. It can change in given moments of the day and in given periods of the year, furthermore differing for geographical area and for energy source. Moreover, a Cloud site can use a solar energy source and the efficiency of its Photo-Voltaic system (PV) can change in different moments of the day (i.e., morning, afternoon, evening, night) and in different periods of the year (i.e., spring, summer, autumn, winter). A really sustainable approach, as that presented in this paper, can help Cloud providers also in receiving funds to realize new *green* plants, thus to produce clean energy and to receive *Renewable Energy Certificates (RECs)*.

The reminder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the main sustainability metrics that have to be considered in our strategy. Section 4 presents our approach to make the best choice in resource allocation. In Section 5, we present a simulation environment for our analytic evaluations considering real parameters, thus proving the goodness of our approach. Section 6 concludes the paper.

2 RELATED WORK

Currently, most of the energy-aware management strategies are specifically focused on independent Cloud providers, and less are beginning to look at Community Cloud. Scientific literature presents several contributions on "green" Cloud, and intra-Datacenters resource scheduling in order to reduce energy consumption, but less attention has been devoted to "sustainable" community, cooperative or federated Clouds. We, instead, present a decision-making approach which is mainly focused on the allocation of instances where they can run in a more sustainable way. The Community Cloud model helps its implementation.

In (Kessaci et al., 2013) the authors present a multi-objective genetic algorithm, named *MO-GA*, to optimize energy consumption, carbon dioxide emissions and generated profit of a geographically distributed Cloud computing infrastructure. Differently from the *MO-GA* approach, we focus on possible advantages shared among a community of CSPs.

In (CHANDRASEKAR, 2014), the authors present a review of literature on Cloud Brokerage Services, but Community Cloud is not a considered.

In (Hamze et al., 2016) the authors present a framework which addresses resource allocation according to an end-to-end SLA. This is established between a Cloud service user and several CSPs in a Cloud networking environment. Compared to that study, our work mainly looks at sustainability taking into account significant parameters by type of service/application.

A survey of the major contributions dealing with energy sustainability and cost-saving strategies aimed at Cloud computing and Federation is presented in (Giacobbe et al., 2015). The survey helps researchers to identify the future trends of energy management in Cloud Federation.

In (Volk et al., 2013) the authors present two complementary energy-efficiency optimization approaches, each one of them covered in the scope of the two European *CoolEmAll* and *Eco2Clouds* projects. However, the brokerage role is not a considered.

In (Usha et al., 2012), the authors propose a work based on a multi-criteria optimization technique for better selection of a service provider, by using a Pareto-based approach to decide the Cloud service provider which satisfies the *Quality of Service (QoS)* requirements for the user. However, that work does not cover the dynamic composition of services based on the migration of data.

3 SUSTAINABILITY METRICS

In order to design the proposed algorithms, we need to preliminary consider several sustainability metrics, that are generally computed based on a real-time monitoring of the electrical loads consumptions at each *Data Center* (DC).

The **Power Usage Effectiveness (PUE)** is a sustainability metric recommended by the *Green Grid consortium* to characterize the DC infrastructure efficiency. PUE is generally defined as follows:

$$PUE = \frac{P_{DC}}{P_{IT}} \tag{1}$$

PUE indicates how much the internal power consumption P_{DC} of a DC exceeds the Information Technology power consumption P_{IT} at the same DC, mainly due to electrical equipments and cooling systems. It is one of the four sub-metrics useful to compute the *Data center Performance Per Energy*.

The **Data center Performance Per Energy (DPPE)** is a sustainability metric introduced by the *Japan's Green IT Promotion Council* in order to improve on the PUE. DPPE is defined by the following Formula (2):

$$DPPE = ITEU * ITEE * 1/PUE * 1/(1 - GEC) \tag{2}$$

and it is essentially based on four sub-metrics:

- the *Information Technology Equipment Utilization* (*ITEU*);
- the Information Technology Equipment Efficiency (*ITEE*);
- the Power Usage Effectiveness metric (*PUE*);
- the Green Energy Coefficient (*GEC*).

The *DPPE* is defined in such a way that a greater value in DPPE indicates a greater energy efficiency (GPC, 2012).

In the following, we will assume that each Cloud service provider dynamically computes the *DPPE* at each site measuring in real time the relative sub-metrics, without exposing functionalities deemed to sensitive or risky for its own business. We denote its value for *n*-th site at time *t* with $DPPE_n(t)$.

The **Carbon Dioxide Intensity Of Electricity** (*CDIE*) is a measure of the quantity of carbon dioxide emitted by an IT infrastructure with respect to the used energy, and it is measured in $kgCO_2/kWh$. It depends on the region where the Cloud site is located and it is based on the government's published data for that region of operation for that year. In particular, we refer to the *Intergovernmental Panel on Climate Change* (*IPCC*) database. IPCC is the leading international body for the assessment of climate change (IPR, 2014). Since *CDIE* changes year to year, it is a time dependent quantity thus we denote the admitted quantity into the generic cloud site *n* at time *t* with $CDIE_n(t)$.

Our strategy involves this factor taking into account the impact of operational carbon usage.

Based on the definitions above, we introduce the **sustainability impact factor** of site *n* at time *t* as the ratio of the above-mentioned *CDIE* and *DPPE* metrics:

$$k_n(t) = \frac{CDIE_n(t)}{DPPE_n(t)} \tag{3}$$

It is expressed in *KgCO2/kWh*. If it is multiplied for the energy consumption (kWh) resulting from running a service at the related *n*-th site, it represents the "weight" in terms of carbon dioxide emission (kgCO2), i.e., the *workload footprint*, which is correlated with that energy consumption. The higher it is, the greater the pollution due to run that service. We remark that the values of *CDIE* are published yearly and that the *DPPE* for a given site changes only when structural modifications are done. Due to this reasons, both of them could be considered constant in time depending on the time period analyzed. In such case the value of sustainability impact factor is written as:

$$k_i = \frac{CDIE_i}{DPPE_i} \tag{4}$$

assuming it constant over the time.

4 ENERGY-AWARE RESOURCE ALLOCATION APPROACH

In this Section, we introduce a new approach to make the best choice in *resource allocation* to push down environmental pollution. We mainly refer to reducing carbon dioxide emissions through the Community Cloud ecosystem, running a certain instance workload at the most convenient DCs, thus contemporary taking into account sustainability, availability and monetary cost criteria.

More specifically, we start from considering resource allocation in terms of *instance* i at a node n, and the related workload $w_{n,i}$. In this context, an instance is a temporary virtual server that needs to be allocated in order to run services. The instance is distinguished from classical static virtual server due to its dynamism: an instance that is allocated on a specific node can be easily moved to other nodes thus to be better managed according to real needs. Workload is expressed in terms of power consumption (kW) needed to run a particular instance.

4.1 Availability and Service Price Criteria

Since we want to develop a method to quantify how the workload submitted to a Community Cloud impacts on its sustainability, we consider the availability of DCs to take into account operating periods of time during which systems can produce pollution.

Availability is the degree at which a system, product or component is operational and accessible when required for use. The product quality model defined in *ISO/IEC 25010* comprises availability as a quality characteristic. Moreover it is an important *Key Performance Indicator (KPI)* generally computed as a function of the total service time, the Mean Time Between Failure (MTBF), and the Mean Time to Repair (MTTR); it is well known that the availability at stable conditions is given by the following equations:

$$av = (MTBF/(MTBF + MTTR)) * 100 \quad (5)$$

when expressed as percentage quota. Physical interpretation of availability is the percentage of time during which a system correctly operates. We will assume that during some not operational periods of time a computational node does not produce any useful work but it could waste power to perform other kind of maintenance activities.

Service price is a quantifiable criterion that addresses customers and organizations in their business. Generally, it is expressed in $\$/h$ (i.e., dollars-per-hour) or $\$/GB$ (i.e., dollars-per-GigaByte). Starting by identifying several profiles of service requests, for example in terms of required running time Δ_r of an instance i at a node n, it is possible to determine the total cost for that service as follows:

$$cost_{n,i} = \int_{t_{start}}^{t_{start}+\Delta_r} service\_price_{n,i}(t)\,dt \quad (6)$$

Usually providers offer instance placement services with a fixed price in the maintenance time. Therefore, eq. (6) becomes:

$$cost_{n,i} = service\_price_{n,i} \cdot \Delta_r \quad (7)$$

4.2 Analytic Aspects in a Cloud-to-Cloud Comparison for an Eco-Sustainable Community Cloud Environment

In this work, we assume that an instance workload has to be moved from a Cloud source node a to a Cloud destination node b in the Community based on sustainability and cost saving goals.

An instance uses electricity to run at any node, and this power consumption generally changes from a node to another due to different technological choices. Moreover, each node can change the power consumption distribution in time. As a consequence, the carbon footprint differs at each node. Therefore, we mainly distinguish two different phases during which an instance can be managed, i.e., the **running** at a specific node and the **migration** from the source to a possible destination.

1. **Running Phase**. To evaluate the carbon footprint an execution instance i has at a generic Cloud node n at time t for a δ long period, we introduce the following function:

$$F_r(n,i,t,\delta) = k_n * \int_t^{t+\delta} (av_N * w_{n,i}(\tau) + (1 - av_n) * p_n)\,d\tau \quad (8)$$

where k_n is the sustainability impact factor at Cloud node n, $w_{n,i}(t)$ is the power consumption to run the i-th instance workload at Cloud node v. The availability av_n at Cloud node n is used to take into account the real usage of the infrastructure when the instance i runs at that node. The 'idle' condition at the same node, instead, is taken into account through the p_n basic power consumption factor. Based on eq. (8), the carbon footprint of a given load l when it runs on Cloud source node a at time t_a for Δ_r time instants is:

$$co2_{a,l} = F_r(a,l,t_a,\Delta_r) \quad (9)$$

Generally speaking, when two different Cloud nodes a and b are considered, their footprints over the same time interval are different ($co2_{a,l} \neq co2_{b,l}$) because both their sustainability impact factor and availability are different; on the contrary, if the two Cloud nodes a and b are in the same bladecenter (or datacenter) they are characterized by similar sustainability impact factors and availability ($k_a \approx k_b$ and $av_a = av_b$), resulting in the same footprint ($co2_{a,l} = co2_{b,l}$).

2. **Migration phase**. To characterize the carbon footprint to move an instance i from a Cloud node c_1 to a Cloud node c_2 within a time δ, we introduce the following function:

$$F_m(c_1,c_2,i,t,\delta) = k_{c_1} * \int_t^{t+\delta} w_{c_1,i}(\tau)d\tau + k_{c_2} * \int_t^{t+\delta} w_{c_2,i}(\tau)d\tau \quad (10)$$

$F_m()$ takes into account the fact that during the migration phase two copies of the instances exist in the source and in the destination node thus the footprint is affected by the power consumption of both of them.

The carbon footprint to move load l from a Cloud node a to a Cloud node b within a time Δ_m starting at t_m, is thus computed as:

$$co2_{a \to b,l} = F_m(a,b,l,t_m,\Delta_m) \quad (11)$$

3. **The Algorithm**. The decision making *energy-aware Brokering Algorithm* (*eBA*) is detailed through the algorithms 1 and 2 using pseudo-code where we used the symbol h instead Δ_r to simplify the notation.

In the proposed Community Cloud ecosystem, the above formulas are used to characterize resources of two providers through their footprints with respect the instance i under examination, in order to determine the best sustainability. Footprints of both source and destination nodes are computed and they are exploited to choose whether it is convenient to run the instance i on the infrastructure of the original service provider, the source, or at destination. The carbon footprint due to the running and migration phases are:

$$co2_{source,i} = co2_{a,i} + co2_{a \to b,i} \tag{12}$$

$$co2_{dest,i} = co2_{a \to b,i} + co2_{b,i} \tag{13}$$

When $\Delta_m \ll \Delta_r$, eq. (12) and (13) simplify into the following:

$$co2_{source,i} \cong co2_{a,i} \tag{14}$$

$$co2_{dest,i} \cong co2_{b,i} \tag{15}$$

Therefore, we can compare the $co2_{source,i}$ with all the possible $co2_{dest,i}$ in order to determine what is the best carbon footprint choice.

5 EXPERIMENTS

In order to evaluate the *eBA* algorithm behavior, we set up simulated scenarios by using the **J2CBROKER** tool (Giacobbe et al., 2016) developed at the *University of Messina*. If compared with well-known simulators (e.g., CloudSim) it differs because its specificity in to simulate brokerage scenarios.

5.1 Datasets

We present a modeling of both services and Cloud sites, thus to provide *input* data for the proposed *eBA* Algorithm. Each offer is modeled by a *json* document which includes two main collections (TABLE 1): the first one refers to a *Service Dataset*, to specify workload and performance parameters, and the second one to a *Sustainability Dataset* to calculate the *carbonfootprint* (Algorithm 2, line 25), the *cost* (Algorithm 2, line 26) and the *opt* evaluation index (Algorithm 2, line 31).

Algorithm 1 : The energy-aware Brokering Algorithm (eBA).

```
nosql_db = newNoSQL()
use nosql_db
define nosql_db.REQs_collection
define nosql_db.OFFs_collection
define nosql_db.reqsTags
define nosql_db.resulting
define reqsTags = nosql_db.REQs_collection.tags( )
define hMapReq = nosql_db.REQs_collection.gather(reqsTags.h)
define hMapOff = nosql_db.OFFs_collection.gather(reqsTags.h)
while true do
    define reqs_status = hMapReq.trigger( )
    define offs_status = hMapOff.trigger( )
    if (reqs_status is true) then
        hMapReq = hMapReq.update( )
    else
        if (offs_status is true) then
            hMapOff = hMapOff.update( )
        end if
    end if
    nosql_db.resulting = hMapOff.calc()
    nosql_db.resulting.find( )
    nosql_db.resulting.sort(co2_footprint, cost, opt, N)
end while
```

Table 1: Service and Sustainability Datasets.

Service Dataset	
Parameters	**Values**
Workload (watts)	200-300
Power basic (watts)	100
Running Time (hours)	10,24,360,750
Number of Instances in each Offer	12,14,16,18,20
Number of Instances in each Request	1,10,20,50
Availability (%)	99.90-99.99
Service Price ($/h)	0.007-0.112
Sustainability Dataset	
Parameters	**Values**
ITEU	0.3-0.6
ITEE	0.1-3.9
PUE	1.4-2.3
GEC	0.0-0.003
CDIE (kgCO2/kWh)	(*)

(*) source:https://www.ipcc.ch

The Service Dataset is obtained from a survey on several “top” providers of IT technologies (e.g., Dell), Cloud services and solutions (e.g., Amazon Web Services (AWS)).

The Sustainability Dataset results from the METI project (MJP, 2012) on characteristics and energy efficiency of several monitored Asian DCs. The simulator select a random value between the range set for each metric and each offer is characterized by its sustainability, cost and availability values.

Algorithm 2: The $calc()$ method of the energy-aware Brokering Algorithm (eBA).

```
1:  define h_num = hMapOff.count(reqsTags.h)
2:  define worst_cf[h_num]
3:  define worst_cost[h_num]
4:  for j = 1 to h_num do
5:      off_num = hMapOff.count(reqsTags.h.j)
6:      define co2_footprint[h_num][off_num]
7:      define cost[h_num][off_num]
8:      define opt[h_num][off_num]
9:      for i = 1 to off_num do
10:         define av, price, iteu, itee, pue, gec, cdie
11:         define w, p_basic, t0, h, dppe, k, a, N
12:         av = hMapOff.j.i.availability
13:         price = hMapOff.j.i.price
14:         iteu = hMapOff.j.i.iteu
15:         itee = hMapOff.j.i.itee
16:         pue = hMapOff.j.i.pue
17:         gec = hMapOff.j.i.gec
18:         cdie = hMapOff.j.i.cdie
19:         w = hMapOff.j.i.workload
20:         p_basic = hMapOff.j.i.p_basic
21:         t0 = hMapOff.j.i.tstart
22:         h = hMapOff.j.i.h
23:         dppe = dppe_func(iteu, itee, pue, gec)
24:         k = k_func(cdie, dppe)
25:         co2_footprint[j][i] =
            integral(t0, h, w, k, av, p_basic)
26:         cost[j][i] = cost_func(price, h)
27:     end for
28:     worst_cf[j] = max(co2_footprint, j)
29:     worst_cost[j] = max(cost, j)
30:     for i = 1 to off_num do
31:         opt[j][i] = a * (co2_footprint_{j,i} / worst_cf_j) +
            (a - 1) * (cost_{j,i} / worst_cost_j)
32:         hMapOff.j.i.update(co2_footprint, cost, opt)
33:     end for
34: end for
```

5.2 Simulation Environment

In the J2CBROKER simulator, both the client and server sides use their own mandatory *client json_conf* and *server json_conf* configuration files to dynamically set features and behavior during the simulation steps. The first one contains information about the server application and several fields which are used for the dataset simulation phase. The second one contains information about the server-side elaboration phase. The *Random Simulation Mode* (we use in our simulation at client-side) allows the random creation of the datasets to send at the server-side broker algorithm for computation. The *Guided Simulation Mode*, instead, allows the user to specify the list of the dataset files for the server-side broker algorithm. The communication between client and server is made through the HTTP POST requests exchange. The output of the simulation is a json file which contains the results of the elaborations done by the server-side *eBA* Algorithm.

5.3 Experimental Results

This paragraph reports, in a graphical form, the results produced by the simulations, based on a number of 1000 samples. Figures 2 and 3 show distinct results for the different running time h (as reported in service dataset) and based on the established parameters (i.e., weight a, confidence in terms of percentage and number N of instances to allocate). In particular, the weight a is a value in the [0,1] range and it is part of the *'opt'* Formula at the line 31 of the Algorithm 2. It is used to assign a weight for each offer in terms of sustainability and cost (the sum of the attributed weight equals one).

Figure 2 shows four examples among the simulated scenarios by using two typologies of graphs, i.e., kgCO2/DPPE vs h and *cost* vs h. They refer to four different number N of instances in each request (see Service Dataset in TABLE 1 and line 11 in the Algorithm 2). Examples report a weight parameter a equals 0.5 (that means to assign the same weight for sustainability and cost) and a 95% in confidence intervals for the selected $kgCO2/DPPE$ (i.e. the carbon dioxide emission compared with the DPPE expressed by the Formula (2)) and *cost* indexes. The purpose of these graphs is to give a clear indication on the amount of carbon dioxide emission-per-DPPE and the cost (i.e., money) varying the running time h at each Cloud site. If compared with the others through the y-axis reading, the fourth graph on the left shows that the carbon dioxide emission-per-DPPE confidence interval is more restricted. This means that the proposed algorithm encourages the broker in to select the 'best' offers in the presence of a high number of instances to allocate for each request. The same by reading the related *cost* graph (on the right). Furthermore, even if both carbon footprint and cost grow with N and h, their relative $kgCO2/DPPE$ and *cost* confidence intervals are below the 67% in the most expensive of all (N=50, h=750), that is a 23% less in wasteful among the Community Clouds.

Figure 3 shows the confidence interval of the *opt* index for one instance allocation. The values reported in Figure 3 are the result of a post-processing phase, by getting as input all the best *opt* values calculated at each run step. If we consider, in fact, that for each run in our simulation, the worst case results in an *opt* index closer to one, the *eBA* Algorithm at Broker is able to select sets of offers with an *opt* index lower than 0.12, that is very low if compared with the worst

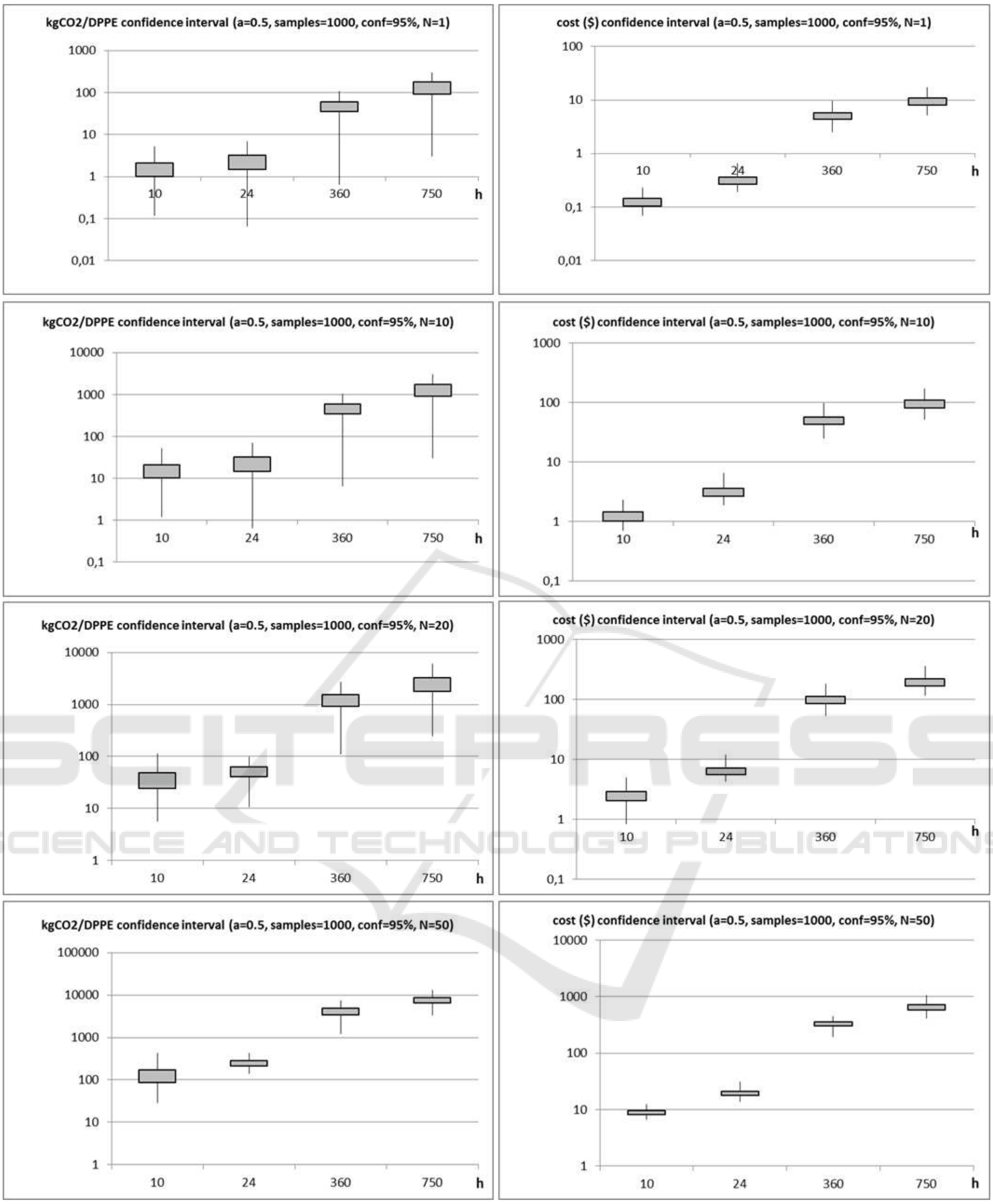

Figure 2: Confidence interval of *kgCO2/DPPE* and *cost* for different number of instances to allocate.

case. It means that the algorithm is able to select sets of offers with an *opt* index closer to zero (the least possible), taking into account not only sustainability but also the service price criterion.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented and discussed an energy-aware Brokering Algorithm to improve sustainability in Community Cloud ecosystems taking care of some metrics we consider particularly useful to im-

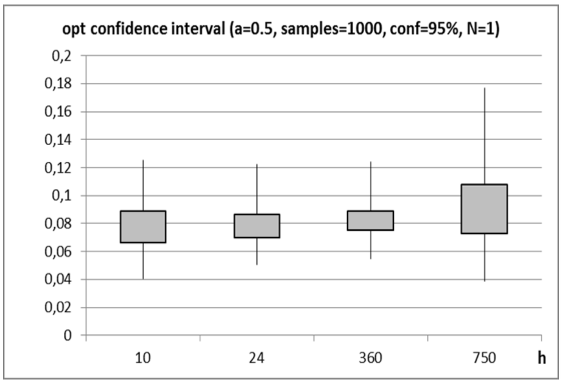

Figure 3: Confidence interval of the *opt* index for one instance allocation.

prove sustainability. The proposed approach is able to discover at a Cloud Broker the most convenient offers delivered by the Community Cloud service providers through a balance between sustainability and cost-saving requirements. The proposed approach allows to characterize offers on the basis of the geographic area where the offered Cloud resources are available, the energy-efficiency of the Cloud site, and service parameters.

In future works, we plan to investigate a strategy to smartly balance sustainability with several others performance metrics. Thanks to an optimum balance between sustainability, cost, and service parameters, a Community Cloud ecosystem can reduce the gap in competition with larger providers, towards an encouraging "green" resource sharing among Community Clouds.

REFERENCES

(2012). The Ministry of Economy Trade and Industry (METI) Japan Project - Enhancing the Energy Efficiency and Use of Green Energy in Data Centers. http://home.jeita.or.jp/greenit-pc/sd/pdf/ds2.pdf.

(2012). New data center energy efficiency evaluation index. dppe (datacenter performance per energy) measurement guidelines. (ver. 205). Technical report, Green IT Promotion Council.

(2014). The Carbon Dioxide Intensity Of Electricity, Intergovernmental Panel on Climate Change (IPCC) Report. http://www.ipcc.ch/.

CHANDRASEKAR, S. (2014). A review of literature on cloud brokerage services. *International Journal of Computer Science and Business Informatics*, 10(1).

Giacobbe, M., Celesti, A., Fazio, M., Villari, M., and Puliafito, A. (2015). Towards energy management in cloud federation: A survey in the perspective of future sustainable and cost-saving strategies. *Computer Networks*, 91:438 – 452.

Giacobbe, M., DiPietro, R., Puliafito, C., and Scarpa, M. (2016). J2CBROKER: A service broker simulation tool for cooperative clouds. In *10th EAI International Conference on Performance Evaluation Methodologies and Tools (Valuetools 2016)*.

Hamze, M., Mbarek, N., and Togni, O. (2016). Broker and federation based cloud networking architecture for iaas and naas qos guarantee. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 705–710.

Kessaci, Y., Melab, N., and Talbi, E.-G. (2013). A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation. *Cluster Computing*, 16(3):451–468.

Murugesan, S. and Bojanova, I. (2016). *Community Clouds*, pages 744–. Wiley-IEEE Press.

Usha, M., Akilandeswari, J., and Fiaz, A. (2012). An efficient qos framework for cloud brokerage services. In *International Symposium on Cloud and Services Computing (ISCOS)*, pages 76–79.

Volk, E., Tenschert, A., Gienger, M., Oleksiak, A., Sis, L., and Salom, J. (2013). Improving energy efficiency in data centers and federated cloud environments. In *CGC*, pages 443–450.