Smart Design for Resources Allocation in IoT Application Service Based on Multi-agent System and DCSP

Mouadh Bali
LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, Algeria
Dept. Computer Science, Faculty of Exact Sciences, University of El Oued, Algeria
E-mail: bali-mouadh@univ-eloued.dz

Abdelkamel Tari
LIMED Laboratory, Faculty of Exact Sciences, University of Bejaia, Algeria
E-mail: tarikamel59@gmail.com

Abdallah Almutawakel and Okba Kazar
LINFI laboratory, Computer science department, University of Biskra, Algeria
E-mail: aboud.aboud2012@gmail.com, o.kazar@univ-biskra.dz

Keywords: IoT, IoT service, resource allocation, cloud computing, distributed constraints satisfaction problems, multi agent system

Received: September 21, 2019

In the present paper, we aim at solving two problems; the first problem occurring in the transformation of the IoT devices (sensors, actuators, ...) to cloud service. Therefore, we work on maintaining a smooth and efficient data transmission for the cloud and support customer applications like: data sharing, storage and processing. The second problem has two dimensions. In the first dimension, the problem is arisen in the submission of cloudlets (customer requested jobs) to Virtual Machines (VMs) in the hosts. To solve this problem, we propose scheduling algorithm for resource allocation according to the lowest cost and load. In the second dimension, the problem lies in the hosting of new VMs in the hosts. To overcome this problem, we need take into account the loads when housing new VMs in different datacenters. In this work, we suggest a resource allocation approach for services oriented IoT applications. The architecture of this approach is based on two technics: Multi Agent System (MAS) and Distributed Constraint Satisfaction Problems (DCSP). The MAS manages the physical resources, making decision and the communication between datacenters, while DCSP used to simplify the policy of the resources provisioning in Datacenters. Variables and constraints are distributed among multiple agents in different layers. The experimental results show that the efficiency of our approach is manifested in: Average System Load, Cost augmentation Rate and Available Mips.

Povzetek: Predlagan je način dodeljevanja virov za storitve v IoT aplikacijah na osnovi večagentnih sistemov (MAS) in zadovoljevanja porazdeljenih omejitev (DCSP).

1 Introduction

Internet of Things (IoT) and Cloud Computing are two paradigm technologies utilized for a wide range of application in our life. IoT is a smart system to connect physical objects with sensors to enable them to collect and share the data via the internet [18].

The cloud is type of parallel and distributed systems. It is described as a model for application execution and data storage [19],[2] Cloud infrastructure allows customers using a large number of resources such as: network, storage and applications [1]. The data centers have a large number of resources commonly known as RA [20]. In cloud computing, RA is an issue due several challenges such as complexity, heterogeneity of resource that resides in the datacenter, scheduling, virtualization, migration [2],[3].

The motivation for studying this problem comes from IoT limited properties including: limited storage capacity and complicated processes (data analysis and a lot of heterogeneity in the devices) [18]. As result, we work on satisfying users' needs by providing resources allocation with lower cost. This cost is computed on the basis of smart solutions in datacenters (best host) according the resource constraints [8]. We provide a distributed resource allocation approach based on two technics: multi agent system (MAS) [17] and distributed constraint satisfaction problems (DCSP) [5], [10], [11], [23]. Overall, our main goal is to provide high performance services and minimize the costs of resources operating.

In this paper, we study two problems related to IoT applications deployment in cloud computing. The first problem (Service Providing) occurring in the transformation of the IoT devices (sensors, actuators, …) to cloud service. Therefore, we work on a smooth and efficient data transmission for the cloud and support customer applications like data sharing, storage and processing. We suggest a number of functionalities for service providing: service creation, service publishing and service search. The second problem (Service Consumption) lies in the selection and execution of the service of resources allocation in the infrastructure of cloud computing. It occurs in two levels. In the first level, the problem is arisen in the scheduling of tasks (service cloudlet) to assign (submit) the cloudlets to the appropriate VMs taking into consideration the service's functional requirements and minimization of resources exploitation cost. In the second level, the problem lies in the hosting of new VMs in the hosts of the different datacenters according to their loads. The hosting of virtual machines has become a difficult issue in the resource allocation systems because each virtual machine is associated to a physical host according its available resources [6]. In order to solve the problem in both levels; we suggest smart solutions that depend on two techniques: The Multi Agent System (MAS) and CSP. The MAS manages the physical resources, making decision and the communication between datacenters. On the other hand, DCSP is used to simplify the policy of the resources provisioning in Datacenters.

We organize the rest of the article as follows: Section 2 presents research works as related to the subject of this paper. Section 3 offers background and basic concepts. The developed mechanism and system architecture are defined in 4 section. Section 5 presents the main scenarios of interactions in the proposed system. Section 6 provides an illustrative example to clarify our approach. The experimental results are shown in section 7, the last section concludes the paper and presents the future perspectives.

2 Related works

Because of the increasing demand of customers in the field of IoT in cloud infrastructure, many researchers have developed a number of methods to meet customers' demands by taking into account the efficiency of resources and operating expenses. Here, we mention some of the work done in this regard.

Ghanbari *et al.* [9] proposed an analytics study for resource allocation mechanisms for IoT. The Authors of this paper seek to provide a model in the IoT resource allocation which aims at reducing load balancing, minimizing operational cost and power consuming. By reviewed and discussed the advantages and disadvantages of this mechanisms, they compared several parameters in different articles such as: availability, performance, bandwidth, cost, energy, QoS, SLA, throughput, etc. Besides, there are more service quality parameters to be studied such as: self-allocation features, self-adapting, modeling and earning from studies past and current behaviour.

Ma *et al.* [13] suggest a model for task scheduling of the workflow in the IoT infrastructure as a service (IaaS) based on deadline constraints and cost-aware genetic optimization algorithm. To their approach is distributed at different levels according to the characteristics of cloud infrastructure due to the important features of the cloud (on-demand acquisition, heterogeneous dynamics and performance variation of VMs) so that no dependency exists between tasks at the same level. To demonstrate the feasibility of this approach, authors used the HEFT to generate individuals with the minimum completion time and cost.

Fayazi *et al*. [7] focus on two factors for resource allocation: the reliability and rapid implementation of the work. Therefore, they suggested cloud resource allocation based on auction mechanism. The increase and the decrease in the reliability are determined by the success or failure of the implementation. These solutions are checked by using imperialist competitive algorithm and cost function which is calculated by make span and reliability values. Beside of the diversity of the techniques used in this work, it needs more flexibility for the heterogeneous resources.

The work of Lu *et al.* [12] present a model to allocate the resources based on fairness evaluation framework by using two sub-models (Dynamic Demand Model (DDM) and Dynamic Node Model (DNM)) to describe the resource demand. The authors employ several typical algorithms in resource allocation like utility-based algorithm to prove their effectiveness. As strong point, this model supports the dynamic resources demands, but it does not take into account of the response time.

Mezache *et al.* [15] suggest a genetic algorithm for resource allocation with energy constraint in cloud computing. They focus on two levels of resource allocation: cloudlets to virtual machines and virtual machines to hosts. These levels allow adapting the resource allocation system and keeping the cloud resources updated by taking into account the current submitted cloudlets.

3 Background and basic concepts for IoT and cloud

In this section, we introduce some basic definitions and concepts as a background for our study.

3.1 Visions on integration internet of things and cloud computing

The hybridization (combination) between IoT and Cloud Computing generates synergy for both technologies and bring many benefits. Cloud infrastructure offers a clear advantage to IoT systems since its datacenters are able to calculate the users' needs of resources allocation efficiently. It ,thus, shortens the execution time, reduces cost and speeds big data processing [16]. This combination between IoT and Cloud Computing allows to provide a number of technical benefits to users (for example, storage, optimization of resource utilization and energy efficiency) [4], [22]. Figure 1 describes the combination between IoT and Cloud Computing.

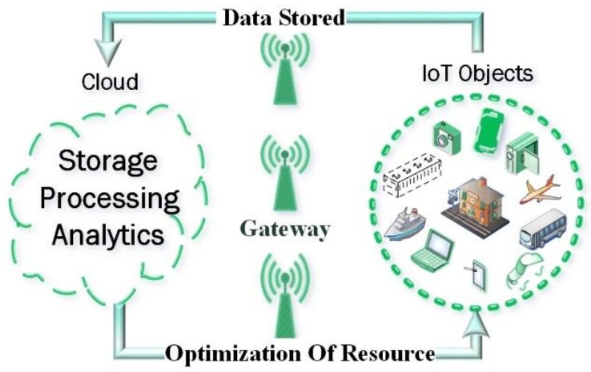


Figure 1: The combination between IoT and Cloud Computing.

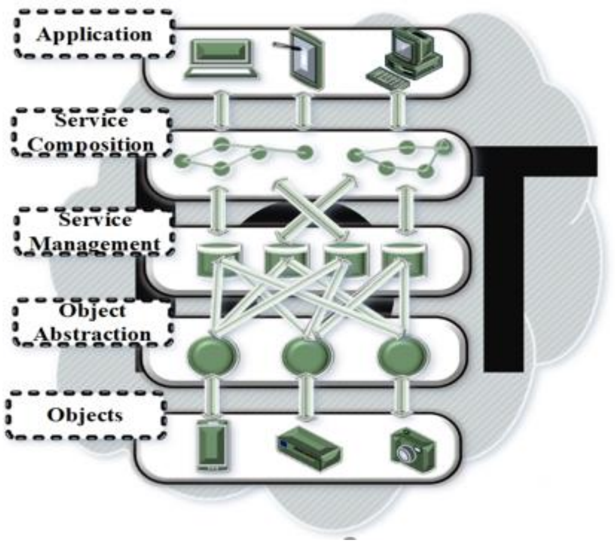


Figure 2: Architecture of IoT service-oriented.

3.2 IoT service-oriented architecture

The aim of service oriented architecture (Figure 2) is to take advantage of the infrastructure of things and the cloud resources for obtaining a better quality of service (reduce the computing costs and improve the overall performance) [24]. The IoT services and devices are usually heterogeneous, and its resources are limited (e.g., memory, processing, bandwidth and energy). To manage such constrained environments, we need to build up a flexible architecture that is capable of managing these resources.

3.3 Components of IoT system

In Figure 3, we present four fundamental components of IoT system (function and mechanism).

- **IoT devices and sensors:**

Sensor is one of IoT devices that has the capability to detect, measure and collect data from the physical environment such as: light, motion, heat, pressure or similar entities [9], [21].

- **IoT gateways:**

The IoT gateway is a bridge between sensor networks and cloud services. The role of gateway is processing the

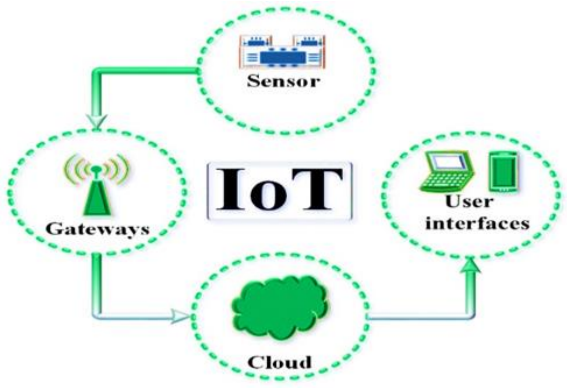


Figure 3: Components of IoT System.

collected data from sensors, then send it the cloud computing [21].

- **Cloud function:**

Cloud function facilitates the advanced analytics and the monitoring of IoT devices in order to shortening the execution time, reducing costs and reducing energy consumption.

- **User interfaces:**

User interfaces are the visible and tangible part of the IoT system. They enable users to contact and monitor their activities in services that they have already subscribed using IoT system.

3.4 IoT deployed applications

In 0 The deployment of IoT devices encounters number of challenges such as: heterogeneity, storage, bandwidth, implementation of management protocols. To overcome these challenges, researchers turn to the combination between IoT and Cloud Computing. This type of combination contributes in the deployment of high, smarter applications for smarter homes and offices, smarter transportation systems, smarter hospitals, smarter enterprises and factories [4], [25].

3.5 The internet of things and multi agent systems

Thanks its characteristics (intelligence, reactivity, autonomy, mobility and the ability to perform making decision). The MAS allows an efficient management for IoT applications in the physical cloud infrastructure such as: the heterogeneity, distribution and the data management In IoT applications. Briefly, MAS provides a decentralized smart solution to frame the new problems and their solutions in the resource allocation approach for services oriented IoT applications [22].

3.6 Cloud infrastructure and constraint satisfaction problem

The Constraint Satisfaction Problem technique is used to formulate and solve several artificial intelligence related problems such as: Scheduling and Optimization [14]. In the cloud Infrastructure, we use DCSP to simplify the policy of the resources provisioning in Datacenters. DCSP problem is formulated as a distributed Variables and constraints to multiple agents. In MAS, each agent makes its proposal plan (solution) by using the distributed negotiation and satisfying its constraints. The various variables and constraints are identified, and the scenario of computing is painted accordingly.

4 Developing a new approach for RA in IoT

At this stage, we proposed a new RA in IoT service. Then, we discuss its System Objectives, architecture, layers, DCSP modelling and system scenario.

4.1 System objectives

This paper is interested mainly in the field of cloud of things. Particularly, it shows the importance of resource allocation in data centers. The aim of our approach is to ensure optimal management of resource allocation for service-oriented IoT applications based on decentralized intelligence in distributed computing. To achieve the stated goals i.e. load balancing (minimizing power consumption), efficiently exploiting resources and minimizing the execution time, we suggest:

1. Designing a system to manage the cloud infrastructure based on a multi-agent system for the allocation of resources in the cloud of things.

2. Developing a system to manage these resources by using two techniques: Multi-Agent System (SMA), Distributed Constraint Satisfaction Problems (DCSP).

3. Implementing and simulating the proposed system through a scenario that demonstrates the effectiveness of the proposed approach for the management of resources in the cloud of things.

In this concern, we introduce a number of concepts and rules for IoT service delivery system specifications and the resource allocation process in cloud computing as shown below:

-Concepts:

1- Cloud service contains a set of parameters (called nonfunctional parameters) such as:

Latency	Cost	Data-format	Availability
Real number	Real number	Real number	Real number

2- To execute cloud service, it requires a set of cloudlet's resources (called functional parameters). The cloudlet is represented in term of (Ram, Storage, Cpu and Bandwidth).

RAM (MB)	Storage (GB)	CPU (mips)	Bandwidth (Gbit/s)
Real number	Real number	Real number	Real number

3- Submission of Cloudlet to VM: is the selection of the Virtual machines (VMs) that have enough available resources to run cloudlet according to its resource requirements.

4- The hosting of VMs in hosts: is the process of selecting the host that provides the least price, low load and the best resources available for this VM.

-Rules:

1- Every object can be linked to many services.
2- Each service has one cloudlet request.
3- Every Cloudlet should submitted to one VM.
4- Every VM can submit more than one cloudlet.
5- Every Host can host more than one VM.
6- Every Datacenter has two types of hosts: ON hosts and OFF hosts
7- Every host has special price.
8- The relationship between the price and the load of the host has a direct impact, where the augmentation in the load causes the increment of the price.

4.2 Smart design for resources allocation in IoT applications

In this section, we are mainly interested in introducing a System Architecture for IoT Resource Allocation, its functional aspect and various layers to provide a better understanding to: how it works, how it stores and how to access to the cloud. Figure 4 describes the proposal smart design.

Layer1 (customer): In this layer, the system focuses on customers and their requests.

The customer requests are presented in term of service name and characteristics.

Layer2 (IoT Service): This layer has a significant the role as mediator between Customer Layer and Broker Layer. It contains two agents:

1. Object agent (OA): is reactive agent that represents an IoT object (physical device). It enables to control exchange and collect data from this device in order to provide a set of services to customers.
2. Mediator agent (MA): is cognitive agent; its role is to manage the customers' requests and the provided services. The main components of this agent are given below:

• **Service registry:** aims to allow the OA agents to publish the information about their services in term of performance and functionalities.

• **Service selection**: searches for a set of selected services in the registry that meet the customer request.

• **Service transfer** MA creates a list of requested cloudlets from the performance characteristics of the selected services. Then, it sends this list of cloudlets to Broker Agent in the next layer. Broker Agent, in turn, arranges this list of cloudlets and send it to Resources layer

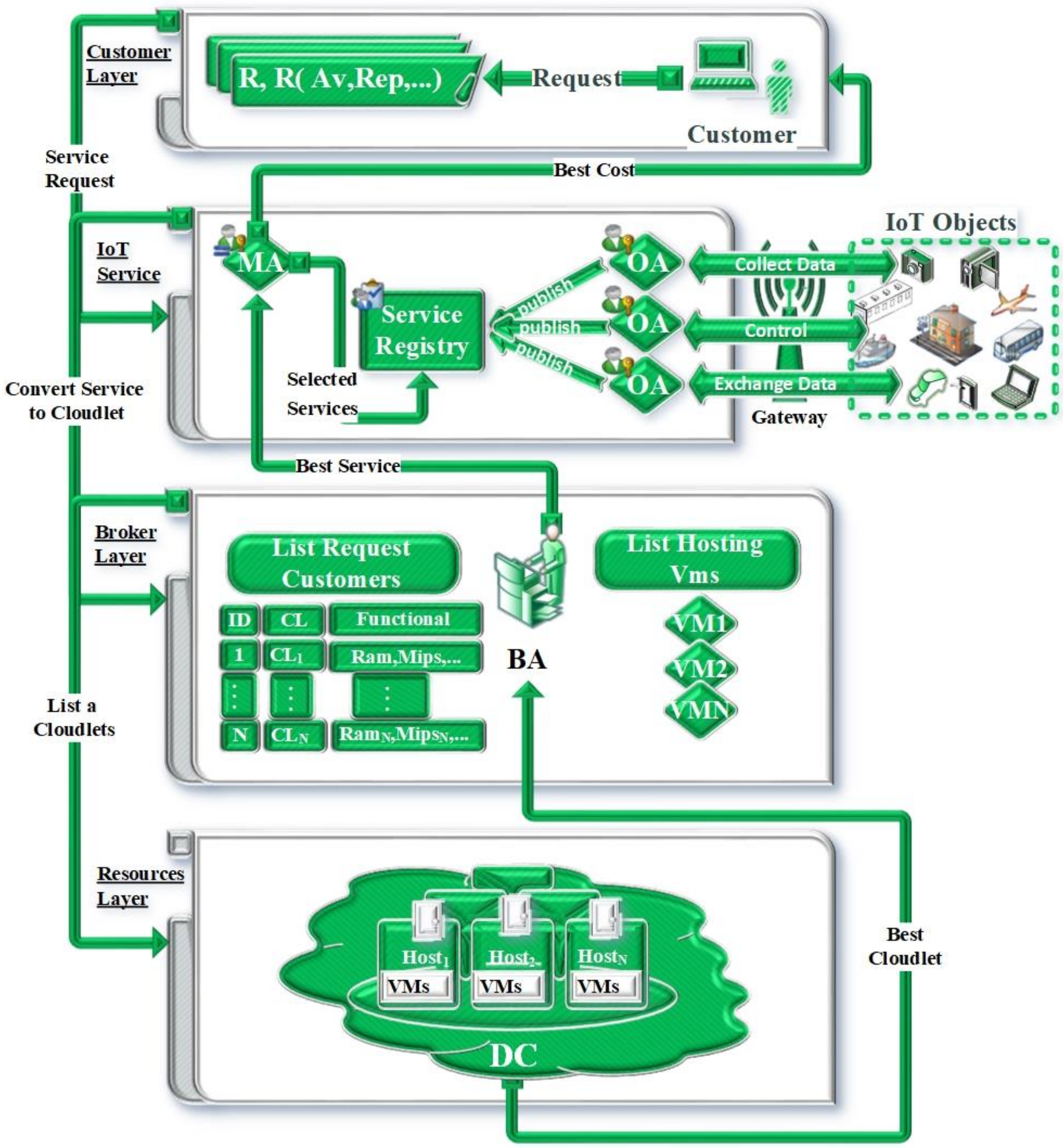


Figure 4: Smart Design for Resources Allocation in IoT Applications.

ID	CL	budget
1	CPU, RAM, Bandwidth, Storage	$
2	CPU, RAM, Bandwidth, Storage	$
…	……..	……

Vm id	Size	RAM	Bandwidth	Mips/pe	Number of Pe
ID	GB	MB	Gbit/s	Mips	Real Number
…..	……	……	……	……	……

Table 1: Example of Broker Agent components.

for the selection of the best cloudlet from this list by taking into account the resource allocation strategy in this layer.

•**Service Bind**: after the selection of the best cloudlet, the MA connects the customer with the provider of the service that is associated by selected cloudlet. It also allows the OA to execute this service through this cloudlet.

Layer3 (Broker Layer): The role of this layer is to manage the resources between IoT service and Resources layer. The broker agent (BA) manages the list of cloudlet requests, free VMs list, performance and delivery of cloud resources. The main role of this agent is to arrange a list of cloudlets, then send to Resources layer.

Layer4 (Resources layer): is the most important layer in the system due to its role in the managing, processing and selecting the best RA for the cloudlet in two levels: local level (between HA agents in the same datacenter) and the global level (between DCA agents of the cloud). This layer contains three types of agents. We introduce these agents and clarify the relationship between them by Figure 5.

Datacenter agent (DCA): communicates with BA and hosts agents in the same datacenter. It also negotiates with other **DCA**.

Host agent (HA): controls a host in state **ON**.

Host off agent HOffA: controls a host in state **OFF**.

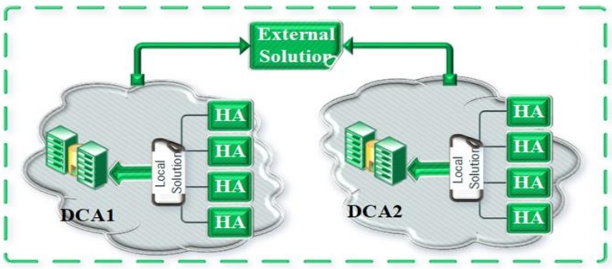


Figure 5: Relationships between DCA and HA agents.

4.3 Relationships between DCA and HA agents

DCSP problem is formulated as a distributed Variables and constraints to multiple agents. In MAS, each agent makes its proposal plan (solution) by using the distributed negotiation and satisfying its constraints. The various variables and constraints are identified, and the scenario of computing is painted accordingly.

4.3.1 Defining of the variables

In this section, we show the most important variables and their definitions Table 2.

4.3.2 Constraints

The aim of this section is to select the best solution for any task in DCSP systems. We thus need to define a set of constraints by using the previous defined variables that correspond to system requirements.

Constraint 1 (Service Usability): verifies a service ***S*** that meets customer request ***R***; it should satisfy the nonfunctional characteristics of the customer request according to the following constraint:

$$\text{S meet R}: \begin{cases} R(Av) \le S(Av) \\ and \\ R(Rep) \le S(Rep) \end{cases} \quad (1)$$

Constraint 2 (Service Capacity): allows the service to handle new customer request. Before representing customer request in term of cloudlet, it should respect its capacity limitation:

$$S(hr) + 1 \le S(Cap) \quad (2)$$

Constraint 3 (Cloudlet Submission ability): virtual machine $\boldsymbol{VM_l}$ has already a set of **M** cloudlets. In order to submit more cloudlet ***m'***, this condition must be satisfied:

$$Cl_{m'}(length) + \sum_{m=1}^{M}\left(Cl_m(length) + Cl_m(outputsize)\right) \le VM_l(ram) \quad (3)$$

$$Cl_{m'}(file\ size) + \sum_{m=1}^{M} Cl_m(file\ size) \le VM_l(storage) \quad (4)$$

$$Cl_{m'}(bw) + \sum_{m=1}^{M} Cl_m(bw) \le VM_l(bw) \quad (5)$$

$$Cl_{m'}(mips) + \sum_{m=1}^{M} Cl_m(mips) \le VM_l(mips) \quad (6)$$

Where:

$$Cl_m(mips) = CL(length) * CL(nbr\_pe) \quad (7)$$

$$VM_l(mips) = \sum_{k=1}^{P} PE_{kl}(mips) \quad (8)$$

$$and\ \boldsymbol{m'}\ \neg\epsilon\ [1, \boldsymbol{M}] \quad (9)$$

Constraint 4 (VM Hosting ability): To allow a Host ***J*** hosting a new virtual machine ***l'*** (free or migrated **VM**), we must verify these conditions:

$$VM_{l'}(ram) + \sum_{l=1}^{V} VM_l(ram) \le Host_j(ram) \quad (10)$$

$$VM_{l'}(storage) + \sum_{l=1}^{V} VM_l(storage) \le Host_j(storage) \quad (11)$$

$$VM_{l'}(bw) + \sum_{l=1}^{V} VM_l(bw) \le Host_j(bw) \quad (12)$$

$$VM_{l'}(mips) + \sum_{l=1}^{V} VM_l(mips) \le Host_j(mips) \quad (13)$$

Where:

$$Host_j(mips) = \sum_{k=1}^{P} PE_{kj}(mips) \quad (14)$$

$$and\ \boldsymbol{l'} \notin [1, \boldsymbol{V}] \quad (15)$$

Constraint 5 (Ranking of Host Agents): The ranking Algorithm is based on **mipsPrice**. In case of finding two Hosts with the same price, then we must use **mipsLoad**:

$$Best\ Host = \begin{cases} \min\left(Host_j(mipsPrice), Host_{j'}(mipsPrice)\right) \\ and \\ \min\left(Host_j(mipsLoad), Host_{j'}(mipsLoad)\right), \\ in\ case:\ Host_j(mipsPrice) \equiv Host_{j'}(mipsPrice) \end{cases} \quad (16)$$

Where:

$$Host_j(mipsLoad) = \frac{Host_j(usedMips)}{Host_j(mips)} \quad (17)$$

$$Host_j(usedMips) = \sum_{l=1}^{N} VM_{j\,l}(mips) \quad (18)$$

Constraint 6 (Best VM Hosting Selection): The selection of the best host between different hosts ***j*** and ***j'*** for hosting VM, it is organized on the basis of Hosting Cost:

Variable	Description	Domain
R	The request of the customer	$\{R_1, \dots, R_v, \dots R_s\}$
O	The abstract object, each object is connected to physical device (gateway, sensor, actuator…).	$\{O_1, \dots, O_w, \dots O_t\}$
S	$\mathbf{S_{xw}}$: The offered service ***x*** by the object ***w***.	$\{S_{11}, \dots, S_{x\,w}, \dots S_{ut}\}$
CL	$\mathbf{Cl_{m\,x\,w}}$: Cloudlet of the service ***x*** from the object ***w***	$\{cl_{111}, \dots, cl_{m\,x\,w}, \dots cl_{c\,u\,t}\}$
R(Av)	The requested Availability.	Rate value (%)
R(Rep)	The requested Reputation.	Naturel number
S(Av)	The Availability of the service.	Rate value (%)
S(Cap)	The Capacity of the service: it is the number of requests can be handled per unit of time.	Naturel number/time
S(Rep)	The Reputation of the service.	Naturel number
S(hr)	The sum of current handled requests by the service.	Real number
Host	Physical host	$\{Host_1, \dots, Host_j, \dots Host_h\}$
VM	Virtual machine	$\{vm_1, \dots, vm_l, \dots vm_v\}$
Host(pe)	Processor in the host	$\{Pe_{11}, \dots, pe_{kj}, \dots pe_{ph}\}$
Host (ram)	Size of host's ram	Naturel Number
Host(bw)	Bandwidth of the host	Real number
Host(Storage)	Size of the host's storage	Naturel Number
Host(mips)	Sum of Capacities of Processors in the host	Real number
Host(used_mips)	Sum of Capacities of the Processors used by virtual machines hosted in the host	Real number
Host(mips_load)	The energy of the host, the Capacity of Processors used in accordance to the total capacity of Processors in the host.	Real number (%)
Host(mips_price)	Unit price of mips in the host	Real number, obtained from proposed model for every host
Pe(mips)	Capacity of the Processor	Real number
VM(size)	VM's hard disc Size	Naturel Number
VM(ram)	Size of the ram	Naturel Number
VM(bw)	Bandwidth of the VM	Real number
VM(mips)	Sum of Capacities of the Processors of the VM	Real number
VM(Cost$_j$)	The hosting Cost of the VM in the host *J*	Real number (DA)
CL(length)	Size of the of CL.	Real number
CL(file size)	Total size of files of CL	Real number
CL(output size)	Size of the result of the execution of CL	Real number
CL(nbr_pe)	Max number of Processors of CL	Naturel Number
CL(mips)	Capacity of the Processors of Cl	Real number
Cl(Cost$_{lj}$)	it is the cost of resource exploitation of the submitted Cl in the VM *l* which is hosted in the host *j*	Real number (DA)

Table 2: Defining of the Variables.

$$Best\ Vm\,Hosting = min\left(Vm\left(Cost_j\right), Vm\left(Cost_{j'}\right)\right) \quad (19)$$

Constraint 7(**Best Cloudlet Selection):** selection of the best Cloudlet (service) for customer request is based on resources exploitation Cost:

$$Best\ CL = min\left(Cl\left(Cost_{l\,j}\right), Cl\left(Cost_{l'j'}\right)\right) \quad (20)$$

Where:

$Cl\left(Cost_{l\,j}\right)$: is the cost of resource exploitation of the submitted *Cl* in the ***VM$_l$*** which is hosted in the host ***j.***

$Cl\left(Cost_{l'j'}\right)$: is the cost of resource exploitation of the submitted *Cl* in the ***VM$_{l'}$*** which is hosted in the host ***j'.***

5 Scenario of interactions in the proposed system

In this section, we present the main scenarios to provide and select RA for IoT service in the proposed system. Also, we illustrate the interactions between Agents by sequence diagrams where there are two object agents (OA1, OA2) and two datacenters agents (DCA1, DCA2). Every datacenter has two Host Agents (HA1, HA2).

5.1 Global interaction

In this section, we explain the global interactions in the proposed system on three main levels: IoT Service request, Cloudlets Submission and Hosting Virtual Machines. The Search Algorithm and diagram in Figure 6 present the detailed descriptions for these interactions.

5.1.1 Search algorithm

```
Search Algorithm
Input
   Request: customer request contains the nonfunctional characteristics (Av and Rep).
   SR: Service Registry contains the services list and their Characteristics.
Output
  SL: List of found Services
  SL = Ø
  for all S in SR do
     if (Request, S) verify C1) then
       if (S verifies C2) then
          add S to SL
       end if
     end if
  end for
  Return SL
end.
```

Algorithm 1: Search Algorithm.

5.2 Cloudlets submission

The process of cloudlets submission in datacenter and their hosts is illustrated in Figure 7. In addition, the Planning Algorithm (Algorithm1) illustrates the process of cloudlets submission inside the Hots.

5.2.1 Planning algorithm

```
Algorithm Planning
Input
   R: List of requested Cloudlets
Output
  BCL: Best Cloudlet
  BCL = 0
  For all VM in this host do
    While ∃ CL ∈ R and (VM, CL) verify C3 do
       remove CL from R
       if (BCL=0 or (CL(cost), BCL(cost)) verify C7) then
          BCL = CL //the new CL is the best cloudlet
       end if
    end while
  end for
```

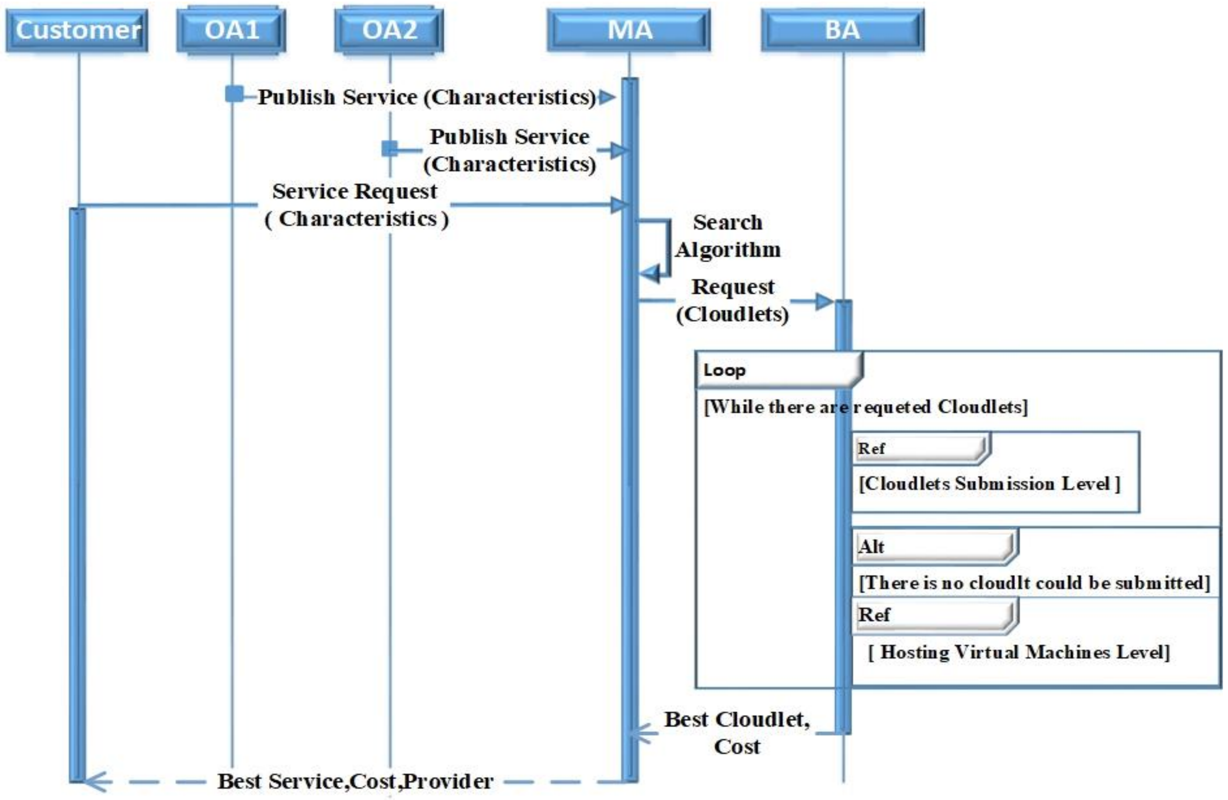

Customer
OA1
OA2
MA
BA
Publish Service (Characteristics)
Publish Service (Characteristics)
Service Request (Characteristics)
Search Algorithm
Request (Cloudlets)
Loop
[While there are requeted Cloudlets]
Ref
[Cloudlets Submission Level]
Alt
[There is no cloudlt could be submitted]
Ref
[Hosting Virtual Machines Level]
Best Cloudlet, Cost
Best Service,Cost,Provider

Figure 6: The global interactions in the system.

```
  Return BCL
end.

```

Algorithm 2: Planning Algorithm.

5.3 Hosting virtual machines

In a case where there is no VM resource available, we launch the Hosting Virtual Machines to submit the requested Cloudlets. The BA starts the process of hosting free virtual machines as illustrated in Figure.8.

6 Illustrative example

To illustrate our approach, we consider an example and discuss a case study of an IoT Application for smart transport system. We discuss this case study from two dimensions:

1. IoT service deployment

First dimension: we focus on the aspect of the defining, publishing and searching services in addition to the different characteristics of these services and the customers' requests. We show a scenario of using this dimension by the following steps:

Step 1: A company has IoT application for smart taxi. It provides the service of reservation of autonomous cars and tracking (monitoring program to be executed in the cloud) the car during the trip.

Step 2: Each autonomous car (physical IoT) is connected to an agent (object agent) in the cloud (IoT layer). This agent publishes information about his service in MA services registry. The Table 3 illustrates some characteristics of the service in term of functional and nonfunctional.

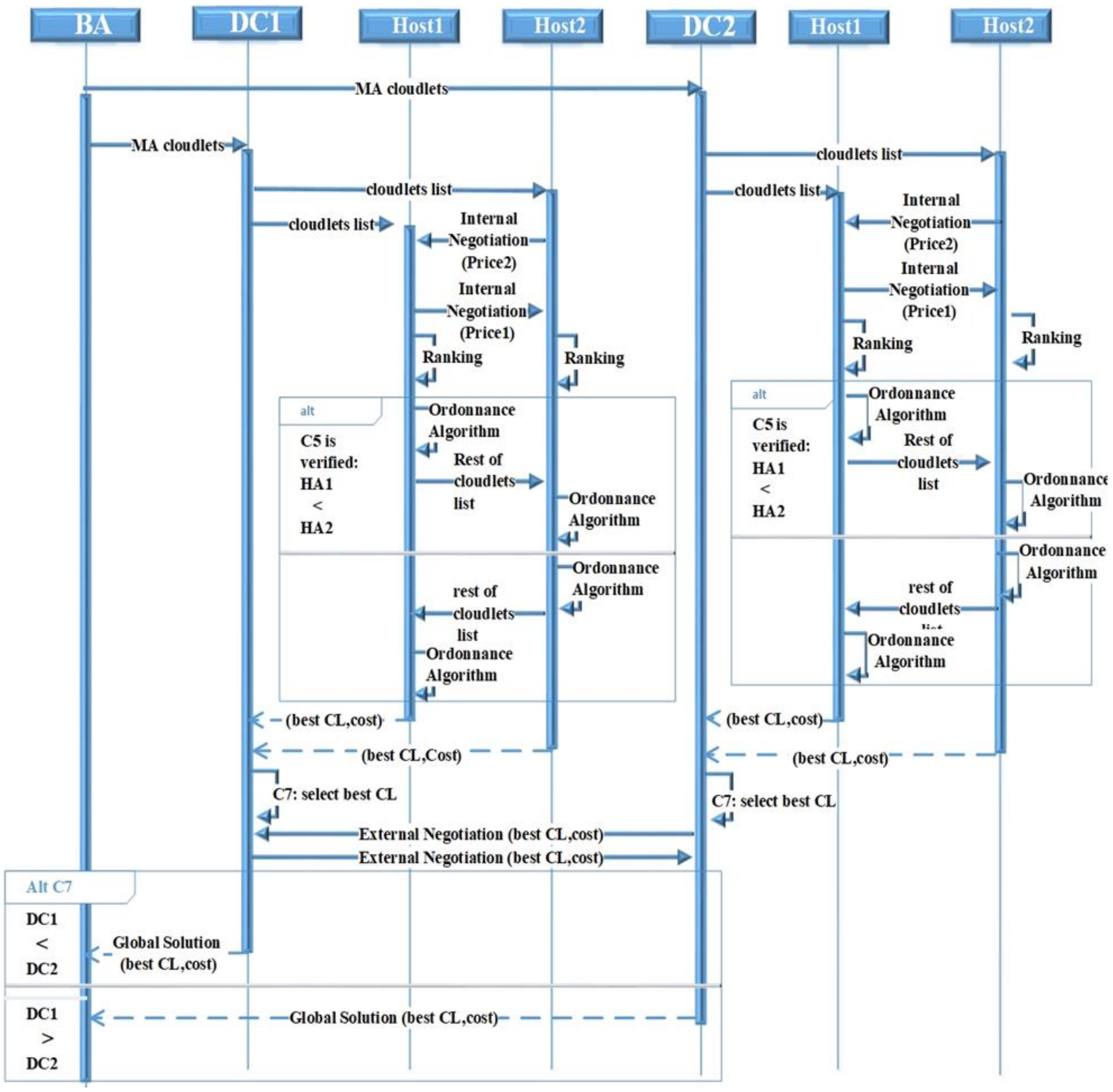


Figure 7: Cloudlets submission between datacenter and their hosts.

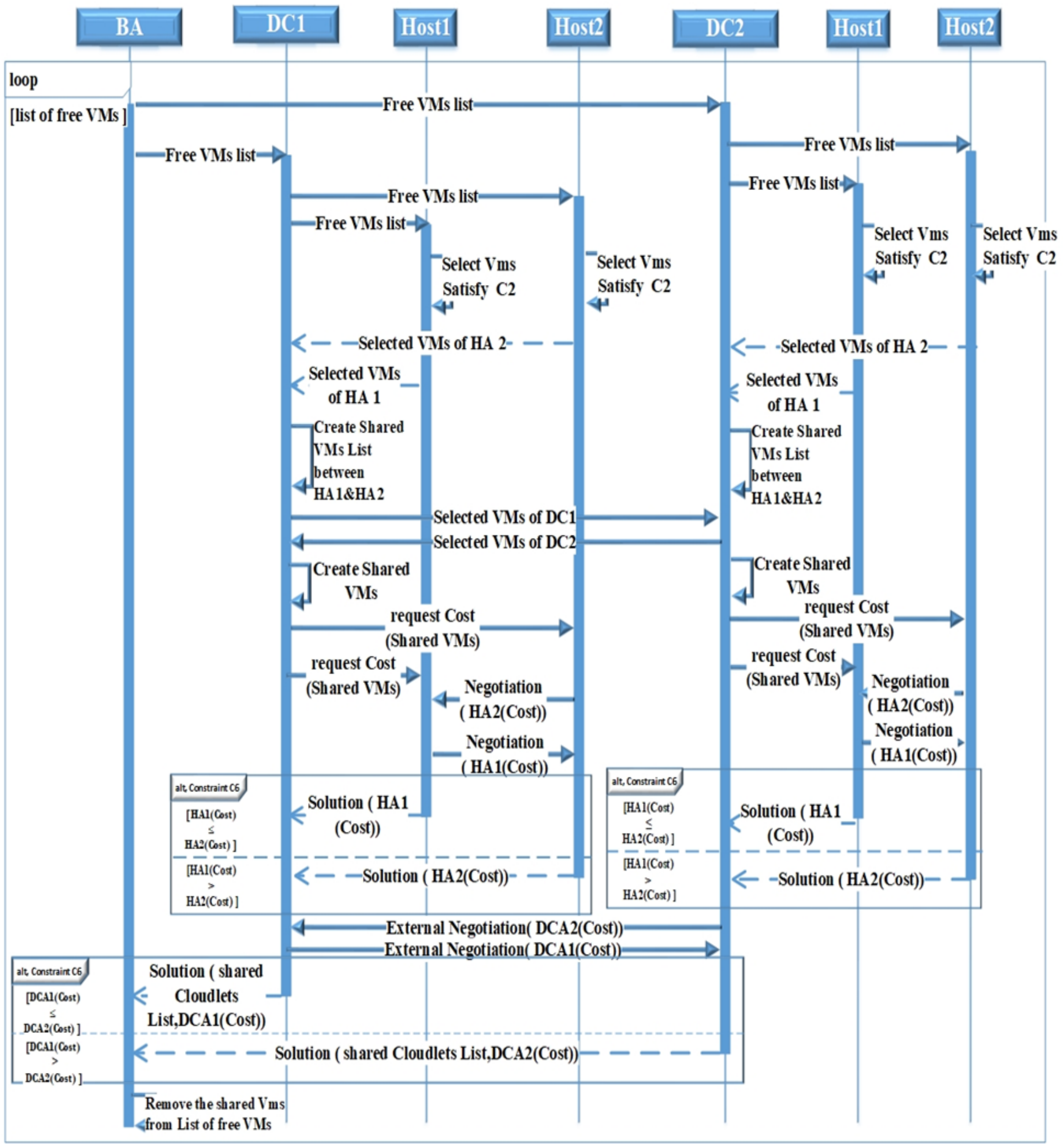


Figure 8: Hosting free virtual machines.

ID	Nonfunctional			Functional		
Agent id	Availability	Reputation		RAM(mb)	Storage (mb)	Cpu
OA1	80%	*****		300	500	2
OA2	65%	**		500	1024	3
….	…	…		….	….	…

Table 3: Services characteristics.

Step 3: The customer requests a car (service) via introducing the nonfunctional characteristics: availability, reputation and the type of desired trip.

Step 4: First, the MA searches in the registry the available services that meet the customer request. In order to select the best service from the found services, the MA converts these services into cloudlets by using resources requirements (from functional characteristics), and sends them to BA in the next layer.

2. Service selection in cloud computing (Planning procedure)

After obtaining the output (convert services to cloudlets) of the first dimension. We discuss how to the execution of the planning procedure in the second dimension in the cloud system functionality. We propose the cloud infrastructure that has two imaginary datacenters: where datacenter 1 has four hosts and three hosts for datacenter 2. In addition, there are eleven (11) Virtual machines (VMs) hosted in these different hosts. These VMs has already hosted thirty (30) Cloudlets, and BA needs to host seven (07) other requested cloudlets (CL31 … CL37) in these Vms. In this case, the system looks forward to check the best resource allocation process for these cloudlets according to the cost and energy consumption as shown in the following steps.

Step 1 (requests):
BA distributes the received list of cloudlets to all DCA. As result, every DCA informs his HA agents who are in ON state to start the ranking process.

Cloudlet id	Length	File size	Outputs size	Number of Pe
31	10MB	2 MB	1MB	2
32	13MB	1 MB	1MB	1
33	5MB	3 MB	2MB	1
34	10MB	1 MB	1MB	1
35	5MB	1 MB	1MB	2
36	2MB	3 MB	3MB	1
37	4MB	2MB	1MB	2

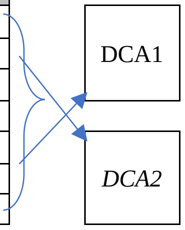


Table 4: Cloudlets List Distribute.

Step 2 (Interne Negotiation 1 "Ranking process"):
After, the **HA** agents (in **ON** state) share their prices and rank themselves into ascending order by the price. As illustrated in Table 5 the ranking in Datacenter **I** is: H2, H1, H4 where **H2** has the price (1.5 $) which is the lowest price. And for Datacenter **II:** H1, H2 where **H1** has the lowest price (1.4 $).

At the end of the ranking, every first **HA** informs his **DCA** by the result of the ranking and asks him to send back the list of cloudlets.

	rank	Host id	price
DCA1	1	H2	1.5 $
	2	H1	4 $
	3	H4	7.8 $
DCA2	1	H1	1.4 $
	2	H2	5.5 $

Table 5: Hosts ranking

Step 3 (Interne Negotiation 2 "Planning"):
After the ranking process, the first **HA** in the each **DCA** gets the list of cloudlets from his **DCA**, and starts the planning procedure by checking available resources in the hosted VMs of his Host and verifies the constraint **C1**. If there are Cloudlets and VMs that verify **C1**, then the first **HA** selects the best cloudlet that satisfies the constraint **C7**. The first **HA** sends the selected cloudlet to the **DCA** in the term of (cloudlet, VM, host, cost) as reply. At the end of his procedure, it sends the rest of cloudlets (they do not satisfy **C1**) to the next **HA** in the ranking list to consider them in his planning procedure. This process is repeated continuously until the last **HA** in the ranking list or there is no rest cloudlet.

Otherwise, in case of there is no Cloudlet that satisfies **C1** in any **HA**, this **HA** retransmits the whole of the list of cloudlets to the next **HA** in the ranking list to consider his planning procedure.

Step 4 (Local solution building):
After the planning, every **DCA** receives the solution from **HA** agents and selects the best solution, which satisfies C7, and consider it as his local solution. Table 6 illustrates the local solutions in **DC1** and **DC2** for **CL31** and **CL37.**

Step 5 (External Negotiation) :
The **DCA** agents share their solutions and negotiate to select the best solution using the best price (to satisfy **C7)**. The **DCA** that is the owner of the best solution sends his solution to **BA** to build the global solution as illustrated in Table7.

Cloudlet id	price	DCA id	Host id	Vm Id
33	37$	DCA2	H1	Vm8

Table 7: Global solution for the Broker Agent

Step 6 (Show solutions and confirmation):
After building the global solution, BA agent sends the cloudlet to MA. As result, MA sends the associated service of the cloudlet as response for customer request, enables (confirms) OA to launch the tracking device of the car and allows the customer to use the car with the lowest cost.

7 Simulation experiments

To evaluate the performance of our approach, we used CloudSim [15] which is a Java based and extensible simulation framework for resource allocation algorithms. In this section, we discuss the experimental configuration and the results obtained by using our approach.

7.1 Experimental configuration

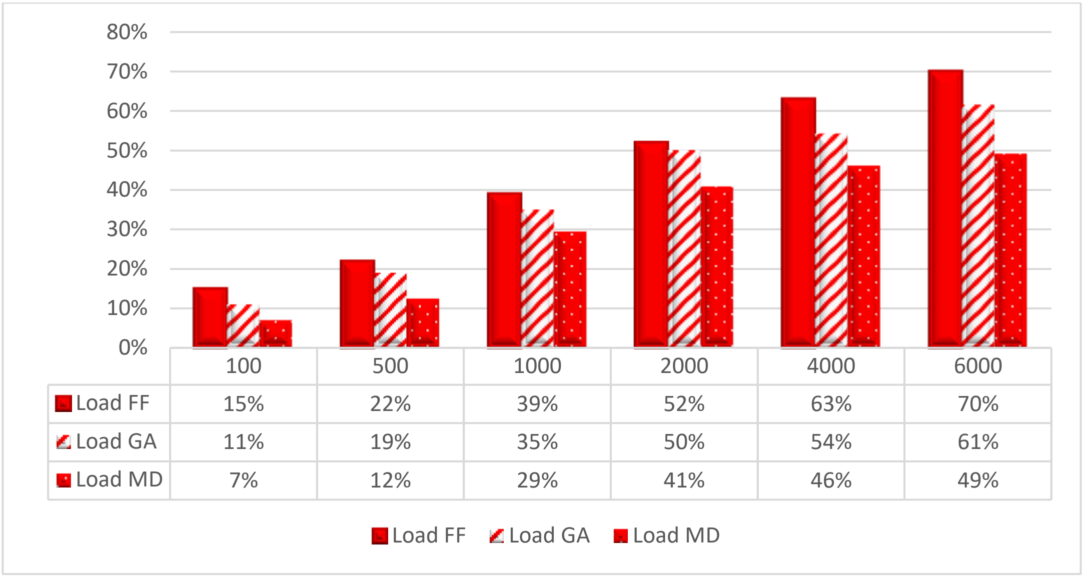
We define the different parameters in our experiments as follows: datacenters, hosts, virtual machines, Processors and cloudlet as shown in Table 8.

7.2 Simulation results

In this section, we present the experimental results and show the efficiency of our proposed approach by making a comparison between three solutions (First Fit algorithm (FF), the proposed Genetic Algorithm (GA) of Mezache et al. [15] and our algorithm (MD)). MD is built on MAS

	Local solution of DCA1				Local solution of DCA2		
Cl id	**cost**	**Host ID**	**Vm ID**	**Cl id**	**cost**	**Host ID**	**Vm ID**
31	39$	H2	**Vm6**	**31**	80$		Vm10
32	43$		Vm3	**32**	72$	**H1**	Vm10
33	41$		Vm1	**33**	**37$**		**Vm8**
34	78$		Vm2	**34**	44$		Vm7
35	78$	H1	Vm11	**35**	54$	H2	Vm7
36	46$		Vm3	**36**	50$		Vm8
37	40$		Vm2	**37**	93$		Vm7

Table 6: Local solution for every Datacenter



	100	500	1000	2000	4000	6000
Load FF	15%	22%	39%	52%	63%	70%
Load GA	11%	19%	35%	50%	54%	61%
Load MD	7%	12%	29%	41%	46%	49%

Figure 9: Average Load by number of requested cloudlets.

Parameters	Values
Max Length of cloudlet	50
Total number of cloudlets	500 –3000
Total number of VMs	530
VM memory (RAM)	100 –1000
Number of PEs requirements	500 –1500
Number of datacenters	3
Number of hosts	47

Table 8: Values of experiments Parameters.

with DCSP. In addition, we have defined performance metrics for the evaluation of the three proposed solutions. These solutions have common characteristics (Average System Load (ASL), Cost augmentation Rate (CR) and Available Mips (AM)). In the experiments, the customer request (Service Request) will be submitted to the IoT system for processing this request. In this case, the proposed system converts this request to a list of cloudlets (network bandwidth, Storage, CUP and load consumed) in order to fulfill this request with lowest cost by using our algorithm (MD). The main goal of our algorithm (MD) is to balance between the cost and energy of Datacenters hosts. The obtained results show that this goal is achieved through the common characteristics (metrics) that are shown as follow:

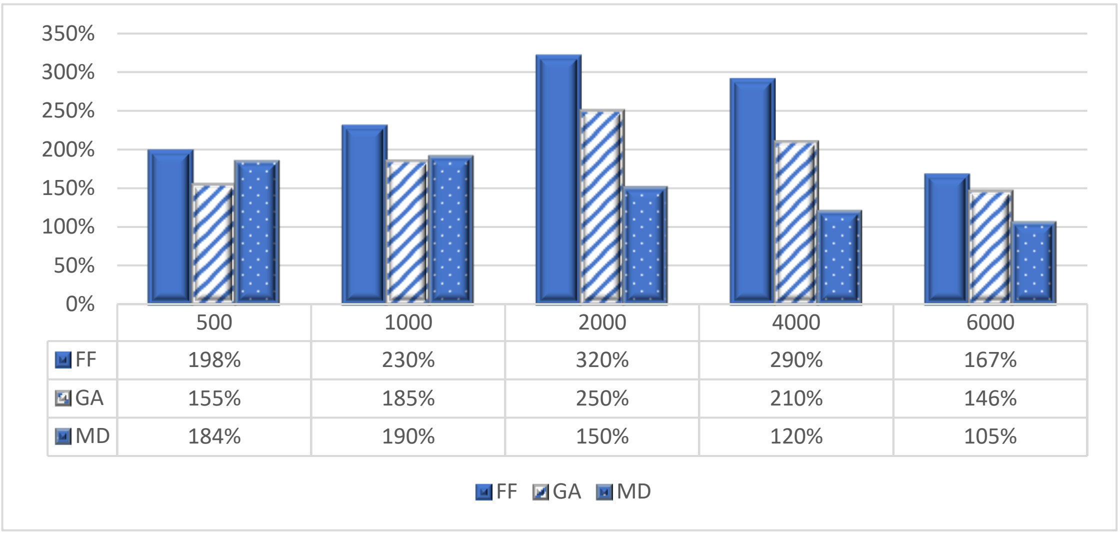
a) Average System Load (**ASL**) This metric represents the energy consumption. The importance of this metric lies in specifying the datacenters status and reducing energy consumption in their hosts. Usually, the ideal system average load gives us a balance between the different hosts inside their datacenters. Figure 9 presents Average Load by the number of requested cloudlets (FF, GA, MD). The obtained results show the efficiency of our algorithms (MD) in getting a lower values of Average System Load (ASL) compared to FF and GA algorithms. The obtained (ASL) values after using our algorithm (MD) improves over the in terms of Average System Load, so that it does not exceed 50%.

b) Cost augmentation Rate (CR) This metric represents the Cost augmentation rate by cloudlets number. The importance of this metric is manifested in reduce the costs of resources exploitation. The values of (CR) in Figure 10 demonstrate the positive contribution of our algorithm (MD) on reducing cost with almost of all groups. Our algorithm (MD) maintains the augmentation rate (CR) between (105% - 190%) except for the first groups (500 and 1000) where the GA has lower values in (CR). This due to the efficiency of our algorithm (MD) with groups which have an important number of cloudlets (more than 1000).

c) Available Mips (AM) This metric represents the Available Mips by Cloudlets. The importance of this metric lies in measuring the computing performance and increasing Available Mips in datacenters. The more MIPS available for the datacenter, the lower cost of the resources exploitation. In Figure 11, we observe that the values of AM obtained by GA are bigger than the values of other algorithms in groups that have less than 1000 cloudlets. While, our algorithm (MD) has better values of AM when the number of cloudlets increases over 1000.

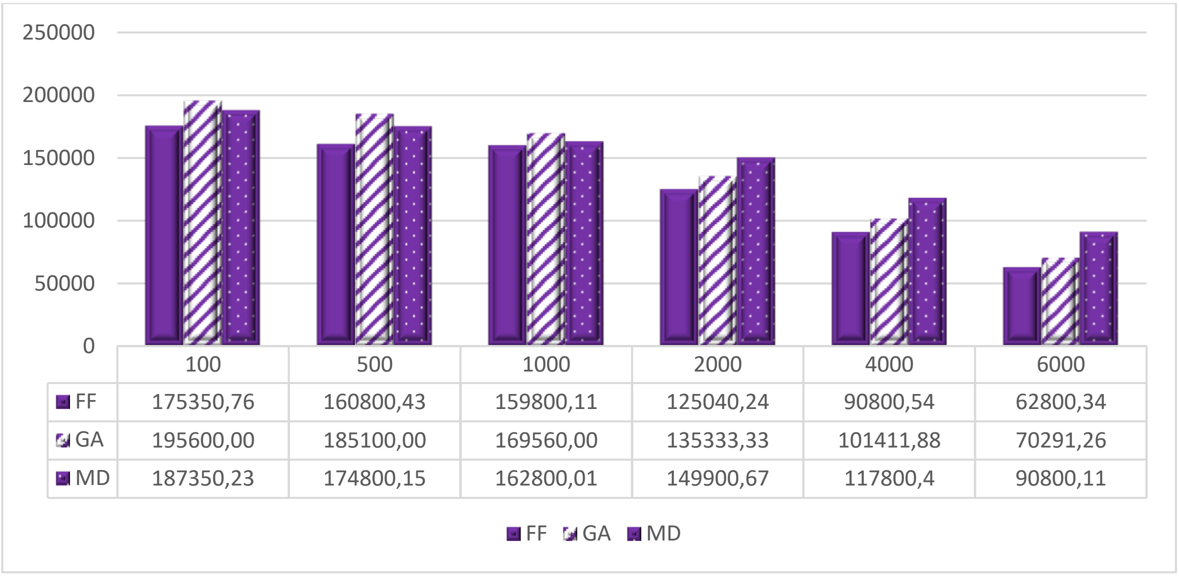
8 Conclusions and future work

In this paper, we addressed a new approach for Resource Allocation (RA) in Internet of Things. Our approach is based to decentralized intelligence into distributed computing by using two technics: MAS and DCSP. In this hybridization, variables are used to present the resources. While the rules and policies are presented by constraints. They are distributed among multiple agents in the different layers of the system. The experiments show that the use of DCSP beside MAS pave the way for new efficient paradigms in solving problems related not only to Resource Allocation but also to provide smart solutions which are helpful to synchronize the IoT application services with computing devices. The obtained results show that the efficiency of our approach is manifested in: (1) reducing energy consumption in datacenters by about



	500	1000	2000	4000	6000
FF	198%	230%	320%	290%	167%
GA	155%	185%	250%	210%	146%
MD	184%	190%	150%	120%	105%

Figure 11: Cost augmentation rate by cloudlets number.



	100	500	1000	2000	4000	6000
FF	175350,76	160800,43	159800,11	125040,24	90800,54	62800,34
GA	195600,00	185100,00	169560,00	135333,33	101411,88	70291,26
MD	187350,23	174800,15	162800,01	149900,67	117800,4	90800,11

Figure 10: Available Mips by Cloudlet.

50 %, (2) reducing cost augmentation Rate between (105% - 190%) and (3) increasing Available Mips in datacenters.

Despite the provided advantages of our approach, we highlight the need of extending in its architecture to support other specific cases for IoT applications. Big data are generated day-to-day from the system, causing many challenges such as, the heterogeneity, scalability and simultaneous accessibility.

In future research, we are looking for enhancing our approach by using more techniques of resources in IoT application services and extending the procedures by exploiting other approaches as: Search Approximation Algorithms, Artificial Intelligence and Fog environments.

References

[1] Anithakumari, S., Chandrasekaran, K., (2017). Interoperability based resource management in cloud computing by adaptive dimensional search, in: IEEE International Conference on Cloud Computing in Emerging Markets, CCEM. pp. 77-84. https://doi.org/10.1109/CCEM.2017.23

[2] Artan, M., Minarolli, D., Bernd, F., (2017). Distributed Resource Allocation in Cloud Computing Using Multi-Agent Systems. Telfor 9, 110-115. https://doi.org/10.5937/telfor1702110M

[3] Bajo, J., De la Prieta, F., Corchado, J.M., Rodríguez, S., (2016). A low-level resource allocation in an agent-based Cloud Computing platform. Appl. Soft Comput. 48, 716-728. https://doi.org/10.1016/j.asoc.2016.05.056

[4] Botta, A., De Donato, W., Persico, V., Pescapé, A., (2016). Integration of Cloud computing and Internet of Things: A survey. Futur. Gener. Comput. Syst. 56, 684-700. https://doi.org/10.1016/j.future.2015.09.021

[5] Chen, J., Han, X., Jiang, G., (2014). A Negotiation Model Based on Multi-agent System under Cloud

Computing, in: In The Ninth International Multi-Conference on Computing in the Global Information Technology. pp. 157-164.
[6] Ezugwu, A.E., Buhari, S.M., Junaidu, S.B., (2013). Virtual Machine Allocation in Cloud Computing Environment. Int. J. Cloud Appl. Comput. 3, 47-60. https://doi.org/10.4018/ijcac.2013040105
[7] Fayazi, M., Reza, M., Enayatollah, S., (2016). Resource Allocation in Cloud Computing Using Imperialist Competitive Algorithm with Reliability Approach. Int. J. Adv. Comput. Sci. Appl. 7, 323-331. https://doi.org/10.14569/IJACSA.2016.070346
[8] Gawanmeh, A., April, A., (2016). A Novel Algorithm for Optimizing Multiple Services Resource Allocation. Int. J. Adv. Comput. Sci. Appl. 7, 428-434. https://doi.org/10.14569/IJACSA.2016.070655
[9] Ghanbari, Z., Jafari Navimipour, N., Hosseinzadeh, M., Darwesh, A., (2019). Resource allocation mechanisms and approaches on the Internet of Things. Cluster Comput. 22, 1253-1282. https://doi.org/10.1007/s10586-019-02910-8
[10] Gutierrez-Garcia, J.O., Sim, K.M., (2011). Agents for cloud resource allocation: An amazon EC2 case study. Commun. Comput. Inf. Sci. 261 CCIS, 544-553. https://doi.org/10.1007/978-3-642-27180-9_66
[11] Jing, L., Weicai, Z., Licheng, J., (2006). A multiagent evolutionary algorithm for constraint satisfaction problems. IEEE Trans. Syst. Man Cybern. Part B 36, 54-73. https://doi.org/10.1109/TSMCB.2005.852980
[12] Lu, D., Ma, J., Xi, N., (2015). A universal fairness evaluation framework for resource allocation in cloud computing. China Commun. 12, 113-122. https://doi.org/10.1109/CC.2015.7112034
[13] Ma, X., Gao, H., Xu, H., Bian, M., (2019). An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing. Eurasip J. Wirel. Commun. Netw. 2019. https://doi.org/10.1186/s13638-019-1557-3
[14] Mataoui, M., Sebbak, F., Beghdad Bey, K., Benhammadi, F., (2015). CSP formulation for scheduling independent jobs in cloud computing. CLOSER 2015 - 5th Int. Conf. Cloud Comput. Serv. Sci. Proc. 105-112. https://doi.org/10.5220/0005438801050112
[15] Mezache, C., Kazar, O., Bourekkache, S., (2016). A Genetic Algorithm for Resource Allocation with Energy Constraint in Cloud Computing, in: International Conference on Image Processing, Production and Computer Science (ICIPCS'2016) London (UK), March 26-27, 2016 Pp.62-69 A. pp. 62-69. https://doi.org/10.17758/UR.U0316020
[16] Mora, H., Signes-Pont, M.T., Gil, D., Johnsson, M., (2018). Collaborative working architecture for IoT-based applications. Sensors (Switzerland) 18. https://doi.org/10.3390/s18061676
[17] Nair, A.S., Hossen, T., Campion, M., Selvaraj, D.F., Goveas, N., Kaabouch, N., Ranganathan, P., (2018). Multi-Agent Systems for Resource Allocation and Scheduling in a Smart Grid. Technol. Econ. Smart Grids Sustain. Energy 3, 1-15. https://doi.org/10.1007/s40866-018-0052-y
[18] Rivera, W., (2017). Sustainable cloud and energy services: Principles and practice. Sustain. Cloud Energy Serv. Princ. Pract. 1-268. https://doi.org/10.1007/978-3-319-62238-5
[19] Roogi, R.H., (2015). Big Data Solution by Divide and Conquer technique in Parallel Distribution System using Cloud Computing. Orient. J. Comput. Sci. Technol. 8, 9-12.
[20] Shrimali, B., Bhadka, H., Patel, H., (2018). A fuzzy-based approach to evaluate multi-objective optimization for resource allocation in cloud. Int. J. Adv. Technol. Eng. Explor. 5, 140-150. https://doi.org/10.19101/IJATEE.2018.542020
[21] Singh, A., Viniotis, Y., (2017). Resource allocation for IoT applications in cloud environments. 2017 Int. Conf. Comput. Netw. Commun. ICNC 2017 719-723. https://doi.org/10.1109/ICCNC.2017.7876218
[22] Singh, M.P., Chopra, A.K., (2017). The Internet of Things and Multiagent Systems: Decentralized Intelligence in Distributed Computing. Proc. - Int. Conf. Distrib. Comput. Syst. 1738-1747. https://doi.org/10.1109/ICDCS.2017.304
[23] Son, S., Sim, K.M., (2012). A price-and-time-slot-negotiation mechanism for cloud service reservations. IEEE Trans. Syst. Man, Cybern. Part B Cybern. 42, 713-728. https://doi.org/10.1109/TSMCB.2011.2174355
[24] Suciu, G., Suciu, V., Martian, A., Craciunescu, R., Vulpe, A., Marcu, I., Halunga, S., Fratu, O., (2015). Big Data, Internet of Things and Cloud Convergence - An Architecture for Secure E-Health Applications. J. Med. Syst. 39. https://doi.org/10.1007/s10916-015-0327-y
[25] Zahoor, S., Mir, R.N., (2018). Resource management in pervasive Internet of Things: A survey. J. King Saud Univ. Inf. Sci. https://doi.org/10.1016/j.jksuci.2018.08.014