AMACE: agent based multi-criterions adaptation in cloud environment

Sofiane Kemchi, Abdelhafid Zitouni, Mahieddine Djoudi

▶ **To cite this version:**

Sofiane Kemchi, Abdelhafid Zitouni, Mahieddine Djoudi. AMACE: agent based multi-criterions adaptation in cloud environment. Human-centric Computing and Information Sciences, 2018, 8 (1), 10.1186/s13673-018-0149-2 . hal-02453951

HAL Id: hal-02453951
https://hal.science/hal-02453951

Submitted on 24 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Kemchi *et al. Hum. Cent. Comput. Inf. Sci.* (2018) 8:26
https://doi.org/10.1186/s13673-018-0149-2

Human-centric Computing and Information Sciences

RESEARCH Open Access

AMACE: agent based multi-criterions adaptation in cloud environment

Sofiane Kemchi[1], Abdelhafid Zitouni[1*] and Mahieddine Djoudi[2]

*Correspondence: abdelhafid.zitouni@univ-constantine2.dz
[1] LIRE Laboratory, Computer Science Department, University of Constantine2-Abdelhamid Mehri, 25000 Constantine, Algeria
Full list of author information is available at the end of the article

Abstract

Efficient resource management in dynamic cloud computing is receiving more and more attentions. Many works are still ongoing, since federated cloud computing is an open environment. We present in this paper a flexible model, new cloud operators can join the federation or leave it. The model integrates interactions between broker agents organization, permitting a multi-criterion migration of a submitted customer request. To implement the presented flexible model, we propose AMACE (agent based multi-criterions adaptation in cloud environment) a run-time self-adaptive approach, oriented agent in dynamic cloud federation environment. The proposed approach is multi-criterion, where a various and no limited number of parameters can be considered during the self-adaptation strategy (i.e. computing the load balance of the mediator agent and geographical distance "network delay" between the customer and provider…). In addition, AMACE cares about maintaining run-time coherence in optimization evolution in order to guarantee a coherent system at all moment during system execution.

Keywords: Cloud computing, Open federated clouds, Dynamic clouds federation, Multi agent system, Multi agent resource allocation, Self-adaptive agent, Multi-criterions self-adaptation, Coherent self-organization

Introduction

Cloud computing is a rising technology. It represents a business model in which enterprises can make an enormous economy by reducing infrastructure investments. In cloud, users usually pay for the usage (i.e., CPU, storage, and network bandwidth, platforms, and application services) counted by the number of instance-hours incurred in a pay-as-you-go model.

Cloud federation comprises services from different aggregated providers offering clients, in addition to sharing a wide range of resources, the opportunity to choose the best cloud service providers. We identify many basic features of cloud federation such as the best cost, service flexibility and availability to meet a particular business or technological need within their organization.

An efficient multi-agent resource allocation technique is necessary for building an efficient system. Brokering agent resource allocation strategy must maximize the benefit in resource allocation, take advantage of the large services offered by the federation, which can be said to realize the best quality of service provision and improve

Springer Open

© The Author(s) 2018. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

the rate of client satisfaction. Open federated cloud is a dynamic environment that is subject to several changes due to its open nature. To face such an environment, AMACE uses a self-organization property in a multi-agent system to allocate providers resources. Users, providers and brokers are considered as agents: the user (CA: a consumer agent), the broker (BA: a brokering agent) and the provider (PA: a provider agent). BA will assume the complicated task of resource allocation between the users and providers of the clouds federation. Nowadays, the world of business is a highly dynamic environment. Haresh et al. [1] considered a static model system however; we present in this paper a flexible model, with a possible evolution of the federation. New providers can be acquired to satisfy new client's requests; other providers can leave the federation too. The model integrates interactions between broker agents organization, permitting a multi-criterion migration of a submitted customer request.

After checking all clouds provider contact's list, if the BA was incapable of satisfying the client resource demand and could not find the client request resource, this would be called a failure situation i.e., an unavailable resource and an unsuitable price. Interactions between BA organizations permit request's migration between BA neighbors to delegate unsatisfied costumer requests.

Users in clouds federation request more than one type of resources and from different providers which can leave or join the federation. In addition, the resource's prices are responding to a demand–supply model so they are highly variable. Consequently, users need continuously to be refreshed by updated information about all current available providers' services and status of each already present provider in the federation or probably newly joining this one. In open federated clouds, choosing the best provider is very difficult because they do not know the dynamic price of each resource in different clouds. Besides, due to dynamic environment of open federated clouds, users may miss a specific service from one leaving or new incoming, specific provider and so failing to satisfy a specific customer demand. While in cloud computing, it is aimed to increase the rate of client satisfaction and improve the quality of the service provisions. We propose AMACE (agent based multi-criterions adaptation in cloud environment) a run-time self-adaptive approach, oriented agent in open cloud federation environment, new cloud operators can join the federation or leave it. The proposed approach is multi-criterion, where a various and no limited number of parameters can be considered during the self-adaptation strategy (i.e. computing the load balance of the mediator agent and geographical distance "network delay" between the customer and provider…). In addition, AMACE cares about maintaining run-time coherence in optimization evolution in order to guarantee a coherent system at all moment during system execution.

The rest of the paper is organized as follows: "Related works" section introduces the related works; "AMACE: agent based multi-criterions adaptation in cloud environment" section presents the flexible system model, describes AMACE approach and explains in details the auto-adaptive mechanism in an open cloud federation. "Comparison" section makes summary of works in literature with our approach according to some choosing parameters; "Evaluation" section shows the emergent behavior of the broker agent society. Finally, "Conclusion and future works" section summarizes some remarks and plans for future works.

Related works

Unlike traditional cloud computing, Wu et al. [2] treated a novel computing paradigm, self-organizing cloud. To automate the mechanism accomplishing resource allocation problem in cloud environment with dynamic price, authors proposed two novel economic strategies based on mechanism design: the Modified Vickrey Auction and the continuous double auction. The proposed strategies conduct to a transparent self-organizing cloud. Jieun et al. [3] presented an adaptive resource provisioning method based on two main concepts. First, it provides resource provisioning for applications via profiling of scientific applications in a heterogeneous computing infrastructure. Second, it offers an adaptive resource updated according to the availability of resource changes. In [4] Singh et al. suggested an adaptive resource management model which assist in making decisions according to execution time of the workflow. The model and depending on usage history, reschedule resources to improve performance. Ravandi et al. introduced in [5] a novel framework based on separation between the data layer and control. Authors used the black box and self-learning approach to design a self-organized and self-adaptive resource provisioning. In [6] Ghobaei-Arani et al. developed dynamic and adaptive resource provisioning approach, authors use a hybridization of the autonomic computing and the reinforcement learning. The proposed approach deals with the unexpected states like work overload, over provisioning and under-provisioning.

There have been many approaches and algorithms proposed for multi agent resource allocations and self-organization multi agent systems in cloud computing environment. Many of these approaches and algorithms are only based on a non-flexible environment model: Haresh et al. [1] suggested a broker based multi-agent system. Choosing the best provider in federated clouds is a very difficult task because users do not know the price of each resource in different clouds, which is determined dynamically, based on a demand–supply model. In [1] method, the user does not care about both the identity of the cloud provider belonging to a federated clouds and the location of the resources needed. To know which cloud service the provider will perform is not important, as the consumer has to get the resources with the minimum price. Some approaches bring new concepts as 'borrowing' and 'leasing' resources from and to other clouds. Xu et al. [7] Proposed an approach of self-organizing based on multi-agent systems. To achieve the required macroscopic properties of locally interacting agents in cloud market, they suggest a three-layered self-organizing multi-agents mechanism to support cloud commerce parallel negotiation activities. Their consumer model running mechanism uses an algorithm as a protocol of negotiation. Chaabouni et al. [8] Aimed to build a fully automatic system in which the client has simply to fulfill his requirements then loads his work using a centralized architecture of agents based on simple exchange of messages between the supervisor agent and data center agents. The agent who will be charged of the resources management must obey a list of rules. In paper [9] Kecskemeti et al. analyzed different approaches to integrate a reactive knowledge management system and suitable federation mechanisms for an on-demand generation and autonomous management of hybrid clouds. In this paper, the authors extended federated cloud management architectures with autonomous behavior. This work focused on adaptation actions and their possible effects on cloud federations. In order to prevent service level agreements (SLA) violations and resource usage optimization, the knowledge management (KM)

system proposes reactive actions to minimize energy consumption. In the purpose of maintaining a balance between SLA violations and resource consumption, management of cloud infrastructures is done with an autonomous manner. In paper [10] Patel et al. presented methods for VM allocation in cloud providers federation, trying to maximize the cloud provider's profit. The proposed model here is an extension of the clouds simulator for federated scenarios. Two algorithms are developed. The first one allocates the resources to VM while the second allocates VM in order to balance the load in federated cloud environment. Comi et al. [11] proposed an evolutionary approach in addressing the problem of resource management in federated cloud providers by means of an agent cloning mechanism. Intelligent and adaptive software agents' features are made possible by the integration of learning techniques to multi agent systems. Learning software agents operate in order to enhance their performance. The learned knowledge and experience are used to help providers in taking decisions to manage their resources and some other agents with a better performance can reproduce the unsatisfactory agents. In paper [12] Manvi et al. suggested an agent-based resource allocation model (ARAM) for grid computing. Three types of agents are used: job agents (JAs), they are mobile and in charge of executing the jobs at suitable grid nodes. Brokering agents (BAs), they are static, incorporating an economic model and in charge of scheduling resources. Resource monitoring agents (RMAs), they are static, in charge of informing the local cluster servers about resource's status. Comparing to traditional resource management techniques, ARAM aimed at flexible, intelligent, and adaptable services by using mobile agents.

In [13] Lee et al. a distributed resource allocation approach is proposed to solve resource competition in the federated cloud environment. This approach uses the pricing strategy to try to find the equilibrium implicitly. This approach decides which resource could be rented when the outsourcing occurs. The local provider will send an outsourcing request to another provider, which has, suitable resources with the minimal renting cost in case when the local cloud provider does not have available resources. If the outsourcing request is rejected, the local provider will try other providers with higher renting costs until the outsourcing request is accepted. In order to minimize communication overhead, the proposed approach groups tasks according to communication behavior, and tries to allocate grouped tasks to achieve equilibrium when resource competition occurs. Authors show that the cloud provider could obtain more profits by outsourcing resources in the federated cloud with enough resources. Federated-cloud-model in this paper [14] Ishikawa et al., means that two or more independent cloud service providers can be combined together to create one huge cloud environment. In the proposed approach, if a cloud service providers participating to the federated cloud does not have enough computing resources, it can satisfy its user's demand by borrowing resources from other participated providers having enough capacity within the agreed price. In order to deal with dynamic changes of utility spaces during negotiation, authors presented an approach and a prototype system based on simultaneous negotiations among cloud providers and their users to form a federated cloud. The priority of negotiating opponents is decided by a simple method based on a difference of expected utility values obtained by each offer. Zulkar Nine et al. [15] propose as a multi-criteria decision making approach to select the best possible option to route the requests, it is a decentralized

broker based approach that automatically decides about the incoming client requests and route to the most viable cloud member in the federation. The authors present a model that chooses the best destination for outsourcing demand without violating the service layer agreement. In [16] Petri et al. avoid migration to a single cloud environment thanks to the CometCloud system; authors facilitate the federation of several sites following the CometSpace concept thus creating a distributed cloud environment. Authors in this work provide hi level abstraction for supporting cloud federation and "cloud bridging". Carlini et al. [17] present a contribution, optimizing inter-exchange of services among clouds. They proposed a self-adaptive approach based on the Markov-chain model that allows a distributed exchange of services. Adaptation relies on global optimization through point-to-point exchanges between different pairs of clouds. In [18] Son et al. presented an autonomous multi-objectives SLA negotiation mechanism that includes an adaptive trade-off strategy by analyzing workload trends. This determines negotiation in accordance with cloud system's status. Authors in [19] use abstractions and cloud-like capabilities to manage geographically distributed resources. Montes et al. presented a model to support the dynamic federation of resources and coordinate execution of application workflows on such federated environments.

As seen in [1–19] most of the works in clouds concern the study of fixed system models or architecture. There is no possibility to the federated clouds to acquire new providers with challenged price. In addition, services offered are fixed from the start and the federated clouds cannot provide a new kind of services while the open clouds federation is more faithful to business reality.

Few works like: Andronico et al. [20], Hou et al. [21] and De Meo et al. [22] take into account openness features in clouds federation. In [20] work authors include in the federation different cloud middleware, providing a concrete model that looks at heterogeneous cloud systems. In this work, the described model is able to consider all implications in accomplishing and managing dynamic cloud federations. The model targets small cloud operators allowing them to easily join and leave the federation. In [21] they present a self-management approach for the cloud services with an autonomous and context-aware management of the resources by employing a number of service agents in the cloud environment. Based on this approach, they present a cloud-oriented services self-management framework with suitable mechanisms for service aggregations and service provisions. Two supporting algorithms are designed to implement the proposed services self-organization process and the service provision process. Several types of agents are involved in the cloud services self-management such as a service manager agent, a manager center agent, and a service broker agent. Connection of computational nodes enlarges classical grid architecture. De Meo et al. [22] proposed a model supporting open grid federations based on a multi agent architecture, a distributed approach to improve the quality of service in dynamic grid federations where each grid composing the federation is able to accept or refuse node's request for leaving or joining a virtual organization at any moment. Intelligent software agents reorganize the virtual organization according to two integrated parameters: (i) the nodes (past) behaviors, in terms of costs of the resources requested/offered and (ii) the trustworthiness of nodes, which is based on data automatically collected by software agents assisting grid nodes. In order to enhance the rate of requests satisfaction and reduce unallocated resources situations, the grid

formation algorithm ensures a balance function between offer and demand of resources. Although approaches [21, 22] provided good results, they stay limited to a fixed number of criteria: in [21] the service manager considers an optimization and a balance method and in [22], reorganization depends on the result of the combination between trust and historical behaviors into a unified convenience measure. It stays insufficient for a large coverage of all the system's sides such as: the quality of service "time response", computing the load balance "workload parameter", the geographical location or distance "network delay" between the costumer and provider…etc. However, the adaptive mechanism in AMACE takes into consideration optimization of no fixed number of various ranges and conflicting criteria and the list is not limited, it stays open and stretchable.

AMACE: agent based multi-criterions adaptation in cloud environment

Cloud computing is a rising technology; it represents a business model in which enterprises can make enormous economies by reducing infrastructure investments. In cloud, users usually pay for the usage (i.e., CPU, storage, and network bandwidth, platforms, and application services) counted by the number of instance-hours incurred in a pay-as-you-go model. Cloud federation comprises services from different aggregated providers offering clients, in addition to sharing a wide range of resources, the opportunity to choose the best cloud services providers. Many basic features of cloud federation are identified: the best cost, service flexibility and availability to meet a particular business or technological need within their organization.

The dynamicity feature of the considered open federated clouds environment calls for adaptation strategies to allow for flexibility in our system. AMACE supports a flexible model, considering entrance of new cloud providers to the market, offering new services and departure of other cloud providers. The presented flexible model integrates interactions between BA's organization. BA interactions permit migration requests among BA neighbors to delegate unsatisfied costumer requests. The implementation of the proposed flexible model is made possible by hybridization of two approaches.

Three types of agents are considered in Fig. 1: the consumer agent (CA_i *with* $i \in [1, n]$), resource brokering agent (BA_k *with* $k \in [1, m]$) and resource provider agent $\left(PA_j \text{ with } j \in [1, h]\right)$. BA will assume the complicated resource allocation task between providers and users in clouds federation.

In AMACE, each broker has his own contact list of providers. This list can be more or less rich in comparison to other contacts list of neighbors belonging to the broker's organization. The broker agent contains all information about cloud providers in his own contact's list including the location, prices, lowest cost of the resources and the provider, which provides a high quality of service in his own contact's list. The broker agent assigns a grade to the providers based on the feedback of the consumers to which it has done the allocations. All the time the broker agent will be checking the status of each of the cloud providers in his own list. The broker agent negotiates with provider agents and if a single provider does not fulfill the requirement of a consumer agent, it initiates a negotiation with another provider agent belonging to his contact's list.

In case of BA cannot find the client request resource. And in spite of checking all his clouds provider contact's list, BA is always unable to satisfy the client resource demand (i.e., resource still unavailable, price not suitable, a specific provider leaves the federation,

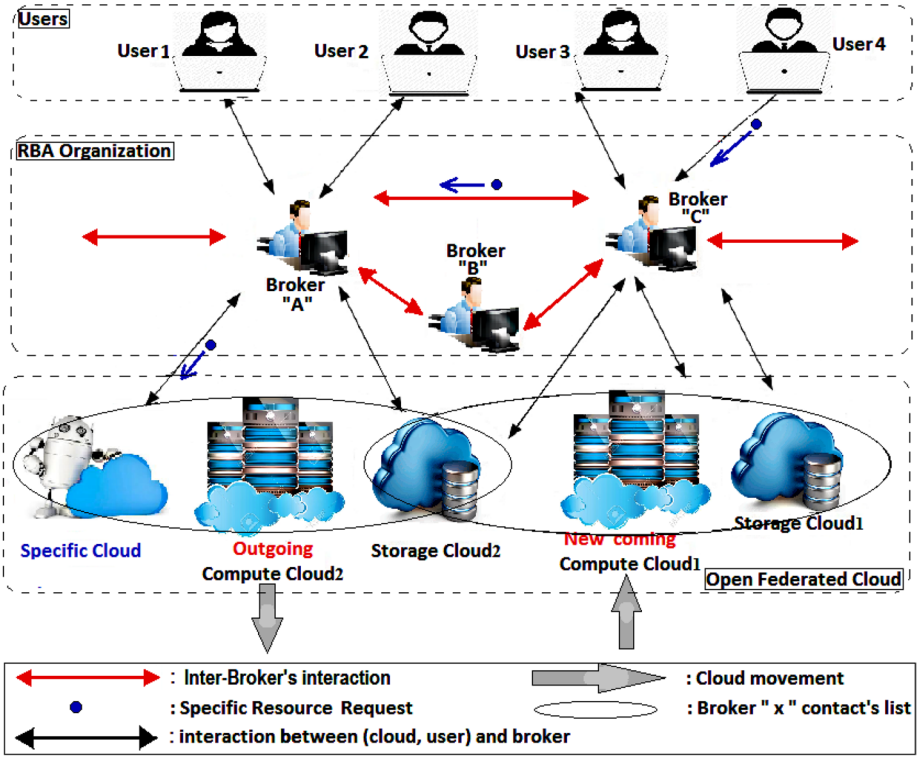

Fig. 1 Flexible system model

entry of a new provider with a particular business or technological need which is not still well known by all brokers…). Such a situation in AMACE, leads BA organization to change the latter. The allocation resource method in AMACE minimizes user intervention and adapts itself. The adaptation mechanism is done by a delegating submitted customer request from one BA to another via multi criterion migration of a submitted customer request. This adaptive mechanism takes into account various parameters such as computing the load balance of mediator agents and the geographical distance "network delay" between the costumer and provider…). For this aim, we propose an algorithm to implement our self-organization mechanism. Migration in AMACE preserves continuously the organization's coherence. Some preventive coherence constraints are verified before request's migration and other corrective coherence constraints are verified after request's migration.

BA self-organization

To achieve self-organization in BA organization, reorganization of the consumer's requests allocation to BA is considered as an adaptive mechanism and to fulfill that, request's migration between BA organizations is adopted as a technical means [16].

Since links between BA are known, a dominance relation based on multi criteria is used to resolve the conflict of choosing BA destination or direction between neighbors. The adaptive mechanism will be engaged in failure situation case, so the BA source delegates to the BA direction the task of satisfying his initial consumer. From this moment, the BA

direction will communicate directly with CA concerned and try to satisfy his demand. If even the new BA direction cannot satisfy the delegated request, the adaptive mechanism will be engaged again and so on until time limit of the request execution. The customer cannot know that his request migrates from the initial broker to another; the adaptive mechanism is done in a transparency manner. The major advantage is the possibility to take into account no fixed number of various conflicting criteria in choosing the BA destination. Where a multi criteria functions evaluations $\boldsymbol{f}(\boldsymbol{k}) = \{\boldsymbol{f}1(\boldsymbol{k}),\boldsymbol{f}2(\boldsymbol{k})\ldots\boldsymbol{fn}(\boldsymbol{k})\}$ require optimization of no limited number of parameters (which can be incompatible) in solving the conflict of choosing the BA direction (i.e., $\boldsymbol{f}1(\boldsymbol{k})$: the BA workload parameter between the BA source and his neighbor $\boldsymbol{k}$, $\boldsymbol{f}2(\boldsymbol{k})$: the BA rate transfer time between the BA source and his neighbor $\boldsymbol{k}$…etc.).

Case of failure situation

Figure 1 shows a case of a failure situation where the adaptive mechanism must be engaged. To start executing his task, User 4 needs to acquire a specific resource provided by a specific provider (for example: a spare part provider of a specific motors company, a specific software provider…etc.). This provider exists in our cloud federation but User 4 makes his request to Broker "C". The problem is that Broker "C" cannot provide this specific resource to User 4 because he has no contact with the specific provider who can allocate the needed resource. In such case, the Broker organization must be capable of self-reorganizing and adapting its behavior to solve the failure situation. Therefore, the self-organization mechanism is automatically started by Broker "C" and starts the multi criterion process of choosing the neighbor destination. After that, the specific resource request migrates from Broker "C" to Broker "A" who is in a position to satisfy directly the specific resource request of User 4.

Figure 2 illustrates the general case of the failure situation where many broker agents interact among them. It shows the migration of a resource request from a broker source to a non-dominate broker "F".

After the choice of BA, the direction is done and before the authorization for migration of non-satisfied customer requests, some preventive constraints are checked to preserve a coherent BA organization. After the migration, other corrective constraints for a coherent reestablishment must be checked which ensures to have, at any moment, a coherent system evolution. Examples of preventive constraints are the avoidance of the migration of a non-satisfied costumer request to an BA direction with an empty provider contact's list and if it is known from the beginning that this BA has no cloud provider in his liable list to satisfy the migrated user's request. As a corrective constraint, after the migration of a non-satisfied customer's request, the workload parameter must be updated for both the BA source and direction. Algorithm 2: BA communication takes in charge self-adaptation in AMACE, which is an online adaptive mechanism.

Open federated cloud

It is noticed in the flexible system model (Fig. 1) interactions among BA organization members (inter organization communications). The request's migration mechanism and

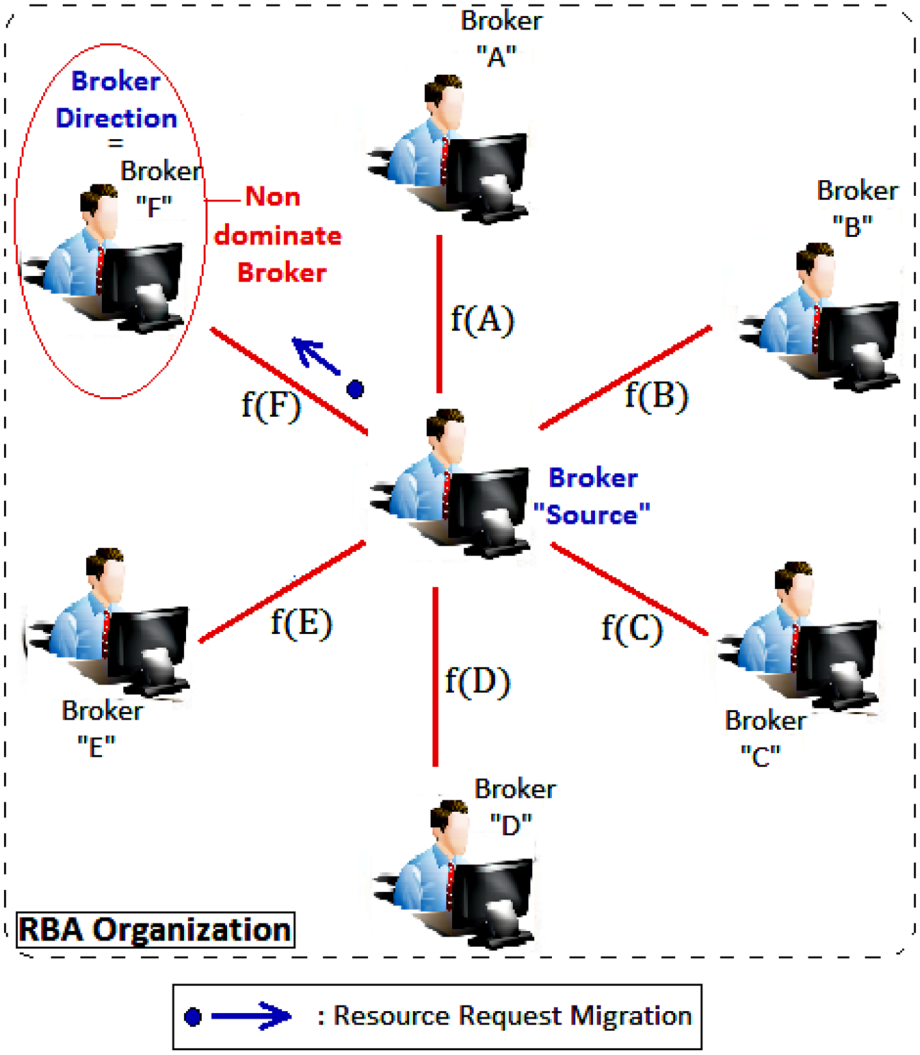

Fig. 2 BA self-organization in AMACE

BA organization interactions features used together in AMACE permit to implement open federated clouds.

Entrance of a new cloud provider

We assume that the integration of a new provider freshly arrived at the federation, implies the creation of contacts with at least one BA from the BA organization. This link with only one BA will be enough for the new coming provider to be reached by all the BA organization. Thanks to BA interactions and request's migration, all consumers will take benefits from this new provider services. For example, in Fig. 1 User 1 demands a request from Broker "A", this request needs a compute resource (as seen in the cloud federation it is provided exclusively by a ***new coming*** compute cloud provider) but Broker "A" have no contact with the new coming compute cloud provider. Interactions between the BA organizations will permit Broker "A" to delegate User's 1 compute resource request to neighbors of Broker "A" and so on until the delegated request will reach the

suitable Broker in contact with the new coming compute cloud provider to allocate this compute resource to User 1. In this case, it is Broker "C" who is in position to satisfy User's 1 compute resource request.

Departure of a cloud provider

In case when a provider will leave the federation before running the task, the PA will send back to BA the resources request that PA planned to provide to consumers. Two situations are possible:

a. If other providers belonging to the BA provider contact's list could provide resources requests sent back, the BA would review again all his providers' contacts list to fulfill resources requests sent back.
b. But if the cloud provider who leaves the federation was a specific and exclusive one in the BA providers' contacts list, the adaptive mechanism would delegate this requested resources to BA neighbors to be fulfilled and so on (the case in Fig. 1 when the compute cloud leaves the federation).

Problem modeling

The consumer agent has the following attributes:

- ***CA_id***: Consumer agent identification.
- ***Rreq***: Set of resources requested by the consumer. For each $r \in Rreq$, $q(r)$ is the quantity of resource r.
- ***Td***: Time duration of resource usage.
- ***Tl***: Time limit of the request execution, it indicated the *latest time* by which the request must be *started*. The request execution must finish before $(Tl + Td)$, or must be started before Tl.

The brokering agent has a Provider_list that contains the information about each resource, its cost and its quality of service. The brokering agent periodically updates the Provider_list. Cost of a resource $r \in Rreq$ is $Cost(r)$. The total cost of $Rreq$ at PA is:

$$C(Rreq) = Td \times \sum_{r \in Rreq} Cost(r) \tag{1}$$

The utility of a consumer agent depends on the resource quality and its payment. It is calculated by dividing the quantity of resources by the cost of resources.

$$U(Rreq) = \frac{q(Rreq)}{c(Rreq)}, \quad \text{while q(Rreq)} = \sum_{r \in Rreq} q(r) \tag{2}$$

Negotiation protocol

In Fig. 1 three protocols are used for negotiation

– Between CA: consumer agents, BA: brokering agents and PA: provider agents,
– In addition, between brokering agents organization members.

Algorithms 1, 2 and 3 are proposed to deal with a dynamic open federated clouds environment. Algorithm 2 permits provider's movement (departure and entrance), permitting inter BA organization communications and integrating an adaptive behavior in case of failure situation, where the BA is incapable to satisfy a user's request. Notice that there is another Algorithm 4 for resource allocations.

– Algorithm 1 is the protocol used for negotiation of

- $CA_{i\epsilon[1,...]}$: Consumer agents with: $BA_{j\epsilon[1,...]}$ brokering agents,
- $CA_{i\epsilon[1,...]}$: Consumer agents with: $PA_{k\epsilon[1,...]}$ provider agents.

Algorithm 1 description

Line 1–4: Consumer agent initializes the consumer request message including his identification then Sends its message to BA with CA_identification. CA waits to receive a cost proposition from BA.

Line 5–11: If the cost proposition is acceptable, CA sends Acceptance answer to BA then waits to receive proposition confirmation from BA. If the cost is higher than expected cost, CA sends reject answer due to cost limit then checks if it is not time limit (*TL*) of the request execution and if ***Rreq*** is not empty, so CA waits to receive new cost proposition from BA.

Line 12–24: If the cost proposition is acceptable, CA sends agreement acceptance to BA and waits to receive agreement confirmation from PA. If the agreement is not acceptable, CA sends agreement reject to BA and checks if it is not time limit (*TL*) of the request execution and if ***Rreq*** is not empty, so CA waits to receive new cost proposition from BA.

When CA finishes sending agreement confirmation to PA, CA checks if it is not time limit (*TL*) of the request execution and if ***Rreq*** is not empty, so CA waits to receive new cost proposition from BA.

Line 25–29: When reaching the final agreement, the consumer agent begins the execution. After completion of the task, CA calculates the utility of the resources and sends it as feedback information to BA.

Algorithm 1: CA Communication

```
Begin
    Initialize Consumer Request = (CA_id,Rreq,Td,Tl)
    Send Consumer Request (CA_id,Rreq,Td,Tl) to BA
    While (t<Tl) and (Rreq ≠ ϕ) do
        Wait to receive Cost Proposition = C(Rreq) from BA
        If (cost proposition is acceptable) then
            Send Acceptance answer to BA
        Else
            Send Reject answer due to (cost-limit) to BA
            Goto line 3.
        End
        Wait to receive Proposition Confirmation from BA
        If Cost Proposition is acceptable then
            If Proposition is confirmed then
                Send Agreement acceptance to BA
            Else
                Send Agreement reject to BA
                Goto line 3.
            End
        Else
            Goto line 3.
        End
        Wait to receive Agreement confirmation from PA
        Accept agreement and send Agreement confirmation to PA
    End
    If (t<Tl) and (Rreq = ϕ) then
        Running Request Execution
        Calculate utility by Eq(2)
    End
    Send feedback information to BA

End
```

- Algorithm 2 is the protocol used for the negotiation of

  – BA $_{j\epsilon[1,\ldots]}$ brokering agents with: CA $_{i\epsilon[1,\ldots]}$ consumer agents,
  – BA $_{j\epsilon[1,\ldots]}$ brokering agents with: PA $_{k\epsilon[1,\ldots]}$ provider agents,
  – BA $_{j\epsilon[1,\ldots]}$ brokering agents with: other BA $_{j\epsilon[1,\ldots]}$ brokering agents neighbors of BA organization.

Algorithm 2 description

Line 2–5: BA waits to receive a consumer request message. BA extracts the resources from the consumer request then updates his current provider list of contacts (to take in account a probably new entrance or departure of providers). BA initializes a temporary provider contact's list with his current provider contact's list.

Line 6–8: BA checks if it is not time limit (TL) of the request execution and if both the temporary contact's list and ***Rreq*** are not empty.

Line 8–10: BA searches each resource in his own temporary provider contact's list based on, maximum QoS and minimum unit cost to select the best provider.

After BA gets the resource list, it calculates the total cost of resource set ***Rreq*** then sends to consumer agent cost proposition and waits to receive an answer from CA.

Line 11–12: If BA receives the acceptance answer from CA, BA initializes a consumer request message and sends it to provider agent.

Line 13–16: If BA receives a reject answer from CA, BA removes the best-selected provider from his temporary provider contact's list and go back to line 6–8.

Line 17–37: In case where it is not time limit (*TL*) of the request execution and if ***Rreq*** is not empty but BA temporary provider contact's list is empty the adaptation mechanism is engaged automatically by executing BA self organization segment part and a feedback about the failure situation is generated to update BA provider contact's list.

Line 38: BA waits to receive response from PA.

Line 39–40: If BA receives acceptance answer from PA then sends proposition confirmation to CA then waits to receive agreement answer from CA.

Line 41–46: If BA receives reject answer from PA, BA sends proposition not confirmed to CA, updates both temporary and current provider contact's list with this demand price, remove best provider from temporary provider contact's list and go back to line 6–8.

In case where is not time limit (*TL*) of the request execution and if ***Rreq*** is not empty but BA temporary provider contact's list is empty the adaptation mechanism is engaged automatically by executing BA self organization segment part and a feedback about the failure situation is generated to update BA provider contact's list.

Line 47–50: BA waits to receive agreement answer from CA. If it is agreement acceptance, BA sends the agreement confirmation to PA, update ***Rreq*** (set of resources requested).

BA check if it is not time limit (*TL*) of the request execution and if both the temporary contact's list and ***Rreq*** are not empty BA try to select another best provider from the temporary contact's list. In case where it is not time limit (*TL*) of the request execution and ***Rreq*** is not empty but BA temporary provider contact's list is empty go back to line 17–37.

Line 51–55: If agreement reject, BA sends agreement reject to PA, remove the best selected provider from his temporary provider contact's list, checks both if is not time limit (*TL*) of the request execution and if the temporary contact's list is not empty, so BA repeats the protocol from the start to select another best provider from the temporary contact's list. In case where it is not time limit (*TL*) of the request execution but BA temporary provider contact's list is empty go back to line 17–37.

Line 56–58: BA updates his own provider contact's list based on the received CA feedback information.

Algorithm 2: BA Communication

```
Begin
  Loop
    Wait to receive Consumer Request = (CA_id,Rreq,Td,Tl) from CA or BA
    Extract resources from Rreq
    Update provider contact's list     //Update new entrance or departure of providers //
    Temporary_provider_list ← Provider_list
    While (t<Tl) and (Rreq ≠ ϕ) do
      If Not(Temporary_provider_list is empty)then
          Select best provider from Temporary_provider_list
          Send Cost Proposition = C(Rreq) to CA
          Wait to receive answer from CA
          If "Acceptance answer" then
             Send Consumer Request = (CA_id,Rreq,Td,Tl) to PA
          Else          //Case "Reject answer (cost-limit)"from CA
             Remove best provider from Temporary_provider_list
             Goto line 6.
          End
       Else // Failure situation, adaptive mechanism engaged //
            If nbre_mig < max_mig then
               Get information from neighbors
               f(0) = {0,0,...,0}
               For all k ϵ neighbors do
                    f(k) = {f1(k),f2(k),...,fn(k)}
               End For
               Repeat
                    direct = q   with not dom (f(q),f(k))
                                          k ≠ q ^ k ϵ neighbors
                    Chosen ← verify_constraint(direct)
                    Remove f(direct)
               Until chosen
               If direct ≠ Source then
                   Send Consumer Request = (CA_id,Rreq,Td,Tl)  to direct
                   Re_establish_coherence (direct)
               End
            End
            Goto line 57.
       End
       Wait to receive response from PA
       If Acceptance answer then
            Send Proposition Confirmation to CA
       Else
            Send Proposition Not Confirmed to CA
            Update cost resources in Temporary_provider_list and Provider_list
            Remove best provider from Temporary_provider_list
            Goto line 6.
       End
       Wait to receive Agreement answer from CA
       If Agreement acceptance then
            Send Agreement acceptance to PA
            Update Rreq
       Else
            Send Agreement Reject to PA
            Remove best provider from Temporary_provider_list
       End
    End While
    Wait to receive feedback information from CA
    Update provider_list with feedback
  End loop

End
```

- Algorithm 3 is the protocol used for the negotiation of
  - $PA_{k\epsilon[1,...]}$ provider agents with: $BA_{j\epsilon[1,...]}$ brokering agents,
  - $PA_{k\epsilon[1,...]}$ provider agents with: $CA_{i\epsilon[1,...]}$ consumer agents.

Algorithm 3 description

Line 2: PA waits to receive consumer request message from BA.

Line 3–5: If all resources are available for assignment, PA sends acceptance answer to BA.

Line 6–8: If one of resources is not available, PA sends reject answer BA. PA goes back to line 2 and repeats the protocol from the start waiting to receive new consumer request from BA.

Line 9: PA waits to receive agreement answer from BA.

Line 10–14: When PA get the agreement acceptance from BA, it sends the agreement confirmation to CA, waits to receive agreement confirmation from CA then PA goes back to line 2 and repeats the protocol from the start waiting receive a consumer request message from BA.

Algorithm 3: PA Communication

```
Begin

  Loop
    Wait to receive Consumer Request = (CA_id,Rreq,Td,Tl) from BA
    If (all r ϵ Rreq are free) then
      Send Acceptance answer to BA
    Else
      Send Reject answer to BA
      Goto line 2.
    End
    Wait to receive Agreement answer from BA
    If Agreement acceptance then
      Send Agreement confirmation to CA
      Wait to receive Agreement confirmation from CA
    End
  End loop
End
```

Figure 3 illustrates algorithm's interaction and communication protocol between algorithms: CA, BA and PA, the flowchart is helpfull for a clearer reading and best analysis.

Data flow diagram

Figure 4 illustrates the communication protocol between algorithms: CA, BA and PA. Before beginning every search, we first make an updating action to the provider's contacts list. CA request migration action to a new BA direction is done in case of a failure situation (provider_list is empty). This situation leads to execute BA self-organization mechanism and a feedback report about failure reasons is sent to providers. The migration process will be repeated again until satisfaction. The self-organization mechanism in AMACE is executed discreetly. Customers do not feel that their requests are migrating from one broker to another and there is no need to know who performed their request.

Comparison

To compare works in literature with our approach, it is interesting to compare them according to some chosen parameters as shown in Table 1.

The most critical difficulties faced during development process of adaptive multi agent system, is definition of the triggering event or the moment when the adaptive

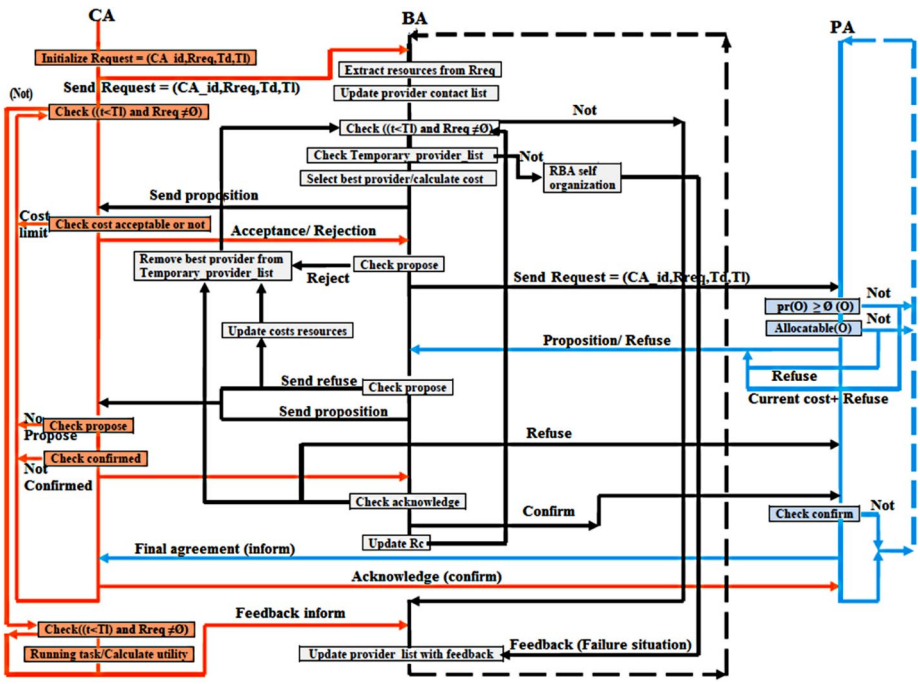

Fig. 3 Approach programming flowchart

mechanism must be engaged. There are two kind of adaptive system: *reactive* system, guided by the events on the system or its environment such as new arrival agent or outgoing one. *Proactive* system, aware about the state of the system, and if it is necessary make decision to adapt it.

The adaptation process responds to a logic following which it decides, if a system requires adaptation. This logic can be *predefined* when designing the adaptation mechanism or an *adaptable* logic that can change. This logic may be maximizing a utility function, maximize the utility function while minimizing the cost of the reorganization process.

Two types of adaptation implementation are defined; the implementation is called *centralized* if an agent is able to decide on its own whether an adaptation is necessary, on the other hand, it is said *distributed* if it is the whole of the agents who must decide if an adaptation is yes or not necessary. *Emergent* behavior is behavior of a system that does not depend on its individual parts, but on their relationships to one another.

There are features that are not related to the adaptation mechanism, but rather link to the system that can be an open system: where an agent can leave the system or a new agent can also join the system. *Open clouds federation* is more faithful to reality; some limits results from the non-flexibility of the clouds federation. In closed federated cloud, there is no possibility to offer new services in the future, if the service is unavailable from the start so it will stay unavailable. There is no evolution of the federation; no new providers can be acquired to satisfy new client's requests. In addition of the risk to be in a vendor lock-in situation, in case where unfortunately in our cloud federation there is a unique provider offering a specific service. Not only there is no possibility of price challenge but also the worst thing is in

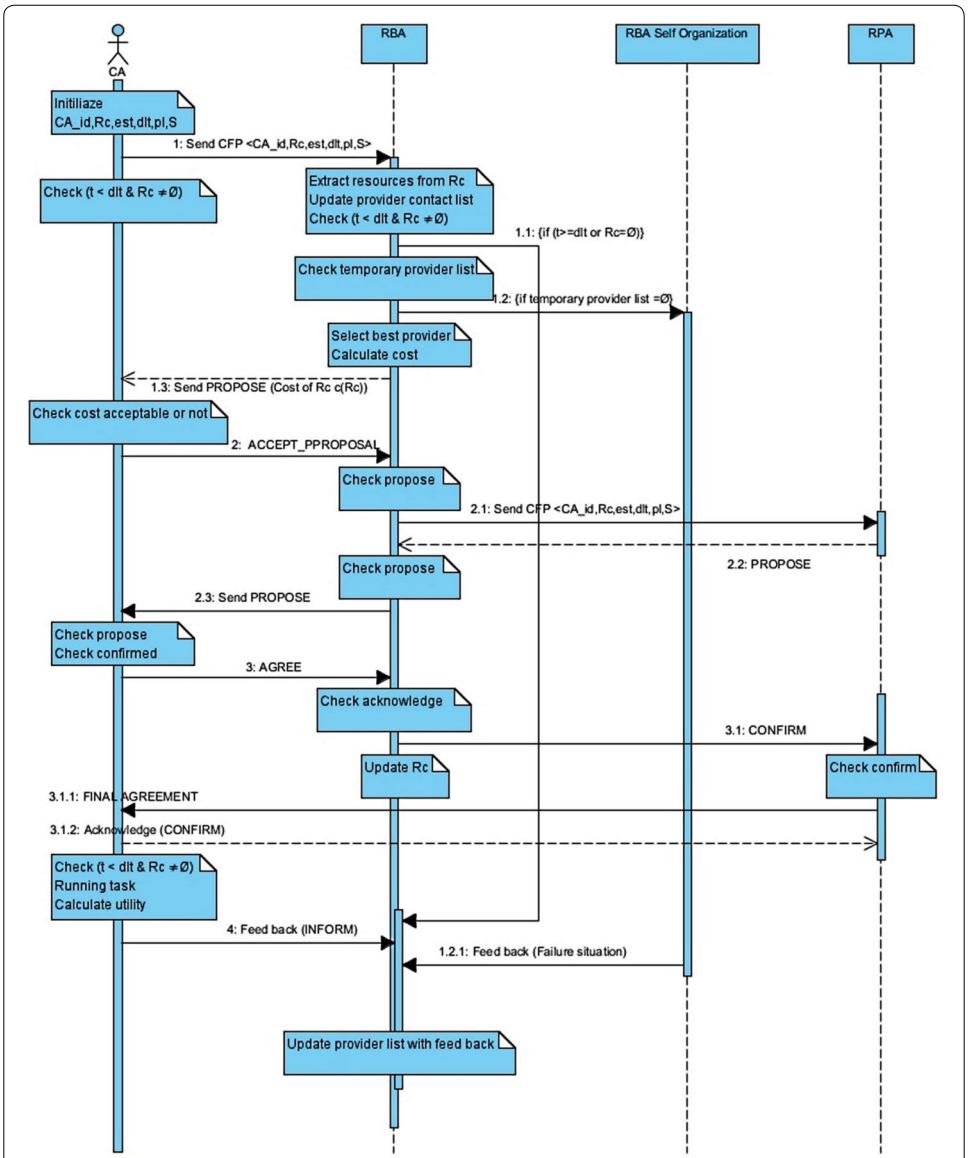

Fig. 4 Sequence diagram

case if this provider decides to stop definitely providing services or decide to leave federation, the whole federation will be enabling to satisfy this specific service.

To maintain the consistency or *coherence* of a system, preventative constraints must be verified before the adaptation mechanism. In addition, other corrective re-establishment constraints should be checked after adaptation.

Evaluation

The choice of Netlogo [23] environment Fig. 5 to implement AMACE is judicious because Netlogo is a programmable modeling environment, excellent to observe natural and social phenomena that emerge from BA organization's interaction. Netlogo permits to give instructions to hundreds or thousands of "agents" all operating independently.

Table 1 Comparative table

		[19]	[14]	[15]	[13]	[17]	[18]	[16]	AMACE
Year		**2014**	**2015**	**2015**	**2016**	**2016**	**2016**	**2017**	**2018**
Adaptation	Strategy	Pro-active	Pro-active	Pro-active	Pro-active	Pro-active	Pro-active	Pro-active/ multi-criterion	Pro-active/ multi-criterion
	Logic	Pre-defined	Pre-defined	Pre-defined	Pre-defined	Pre-defined	Pre-defined	Pre-defined	Pre-defined
	Implementation	Distributed	Centralized	Centralized	Centralized	Distributed	Centralized	Distributed	Centralized
System	Open	×	×	×	×	×	×	×	✓
	Emergent	×	×	×	×	×	×	×	✓
Coherence		×	×	×	×	×	×	×	✓

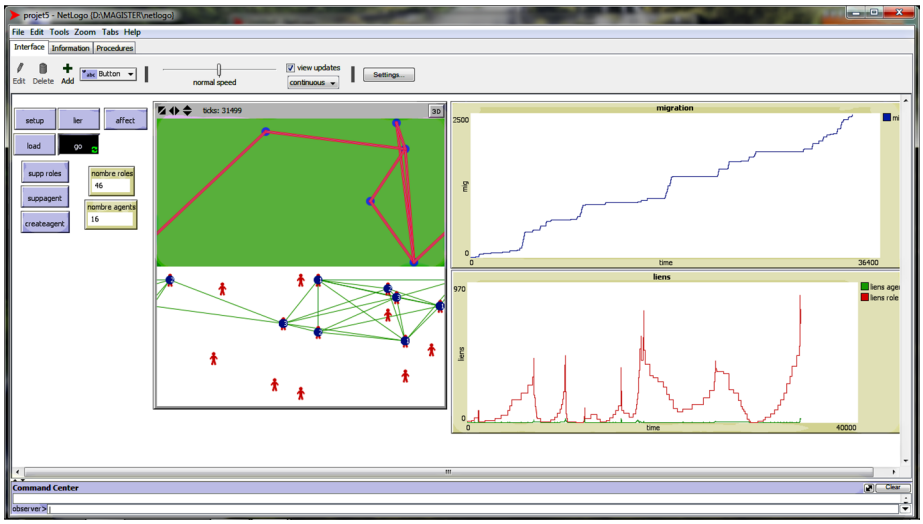

Fig. 5 Tourist services application environment

This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns.

System setting environment

Starting first by world initialization:

- First, each broker agent of the initial population plays only a single role.
- In order to choose the candidate's role for migration from an agent source to an agent destination, a relation of dominance is applied according to multi-criteria specified before. The same relation is used to determine between neighbors of the source, which one will be the agent direction.

Table 2 Experimental setting

Migration counter zi	<set zi 0>, when the role is generated
maxmigs	<set *maxmigs* (random 40) + 1>
Broker Agent (BA)	Initialization <create-agents 10>
Costumer Requests	Initialization <create-rolles 5>

Table 3 List of mathematical notations

F1	Minimal number of tourist services considered as a minimal threshold
F2	Number of tourist services provided which differs with those provided by the agent source's'
F3	Available memory
F4	How many roles broker plays
Dominance relation	X dominates Y if the two following conditions meet: (A) $\forall k \in \{1,2,\ldots, m\}\ f^X_{gk}(ti) \leq f^Y_{gk}(ti)$ 'X is not worse than Y for all criteria' (B) $\exists k \in \{1,2,\ldots, m\}\ f^X_{gk}(ti) < f^Y_{gk}(ti)$ 'X is strictly better than Y for at least one criterion'

As a preventive coherence verification

- This restriction imposes the condition that an agent must provide a minimal number of tourist services (considered as a minimal threshold guarantee) in order to play a role *ri*. In this experimental threshold is fixed at 10 services. **"touristinfo > 10"**.
- The minimum number of tourist services provided by agent B_y which is different than the one provided by agent B_x is higher than a threshold that is fixed before starting the mechanism adaptation at least 2 services. "**nbrdiff > 2"**.
- Each agent has a maximum number of roles fixed at 10 requests that he can play. This restriction aims to have equilibrium in role allocation. **"NBR < 20"**.

Tables 2 and 3 illustrate other experimental parameters setting and list of mathematical notations.

As shown in Fig. 6, a random number of BAs and roles compose the world initially. The "Setup" button permits the generation of simulation environment. Independent roles are on the green part and independent agents are on the white part. Figure 6 shows for example "5" independent roles and "9" independent BAs.

After that, the "*lier*" button will create links between roles as you can see with the red color on Fig. 7; these links permit synchronization and communication between roles. For example, role "3" needs, before execution, some information resulting from role "1" execution and so on.

Initially, the button "*affect*" engages a random role's affectation to BAs with the initial condition of "each agent hosts at more one role". New agents' links will result from this affectation depending on roles links and BA host, they are shown in the green color in Fig. 8.

Buttons "load" and "supp roles" permit respectively to load new roles to the system or to remove roles from the system. The button "go" starts the adaptive mechanism execution. Buttons "create agent" and "*suppagent*" permit to add or remove agents to or from the system.

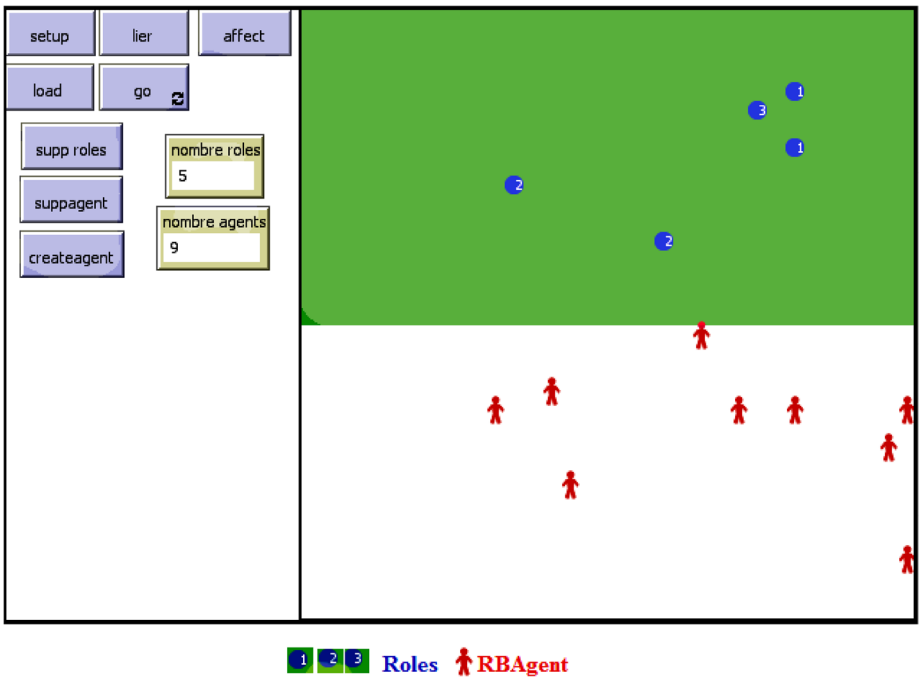

Fig. 6 Initialization of simulation world composed by roles &BA

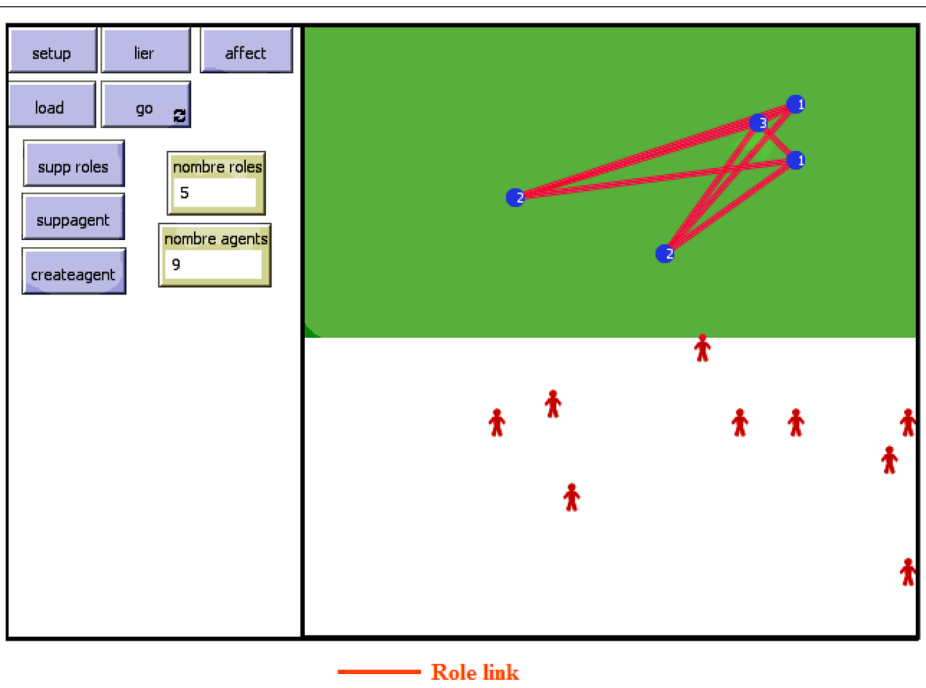

Fig. 7 Initialization of roles links

In order to evaluate AMACE, two aspects must be considered. The first one is an emergent behavioral side of broker mediator organizations. From the auto-adaptive mechanism execution, an emergent behavior of a broker agent is observed. The second

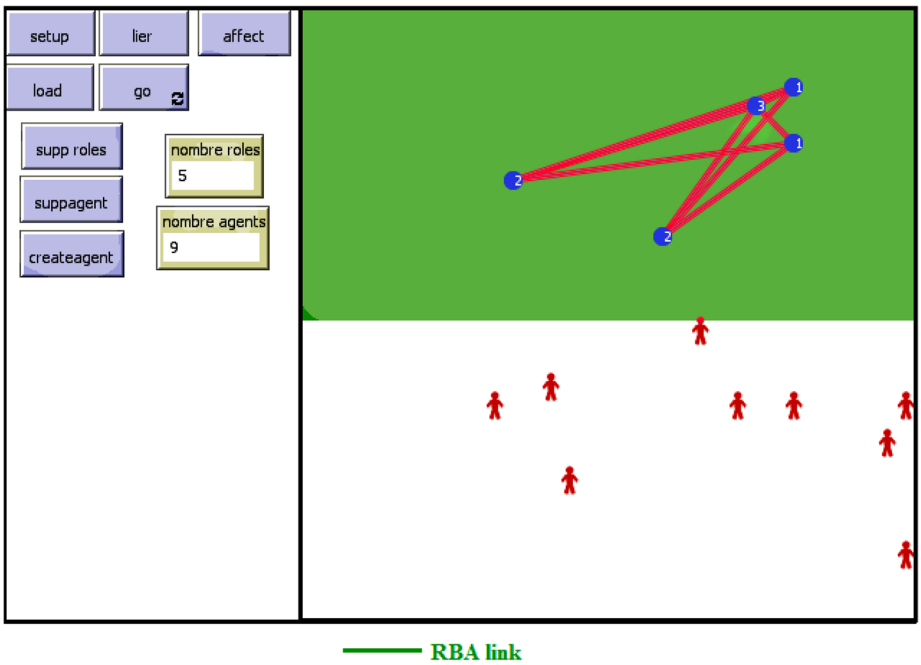

Fig. 8 Initial affectation of roles to BA

one is an empirical performance side, which will demonstrate improvement of AMACE numerical results in comparison with other approaches. After combining the two resulting adapted algorithms, the auto-adaptive mechanism and negotiation protocol communication, the complete system will be implemented to show AMACE effectiveness in terms of client request satisfaction. Evaluation in this paper focuses on the BA organization by implementing the same auto-adaptive mechanism. It aims at observing the BA organization behavioral aspect to underline the emergent collective intelligent behavior observed on the broker mediator organization. The second aspect of evaluation will be done in future work.

Use case

Examples of community cloud in E-commerce are used based on an application of tourist services, which will be shared among a number of users with similar interests and requirements. Three agent's organizations are considered in this application, presented as follow: user agent's organization, broker agent's organization and provider agent's organization. User agent's organization: user agents, model clients looking for tourist services and information about booking of hotels, flights, trains and boats. User agent's organization interacts with broker agents organization by sending services and information requirement. Provider agent's organization: provider agents, model clouds federation as hotels, airlines, trains and boats companies, etc. Provider agent's organization interacts with broker agent's organization by sending services and information requirement. Broker agents organization: broker agents, assume mediating role between user agents and provider agents, our focus will be on the organization of broker agents and its emergent behavioral aspect.

E-commerce is a very dynamic environment that changes continuously. In general, it is the "offer and demand" low that controls the commerce market. Such a high variable parameter involves systems capable of adapting their functionalities to the change of the environment. In our example, the demand of users' agents changes with each newcomer or client. This means new books, requests for broker agents, and other clients may change or cancel their initial booking requests too. In addition, providers change their services continuously, which mean: new services information to update for a broker agent at any time: hotels, flights, trains or boats that could become free or the opposite. Flights, trains or boats can be canceled, time departure can also be changed or a new one scheduled. New broker agents can join our initial multi agent system and other old broker agents can leave it too. All these changes must be taken into account to adapt our MAS and provide a robustness system, which does not breakdown even in a dynamic environment.

Organization transition

First, each broker agent of the initial population plays only a single role. With each role *ri*, a migration counter *zi is* associated that is initialized to zero when the role is generated, and incremented each time the role is migrated. A constant called *maxmigs* defines the maximum number of migrations permitted. A migration takes place only if $xi = 1$ with:

$$xi = \begin{cases} 1, & if\ r \geq \frac{zi}{maxmigs} \\ 0, & otherwise \end{cases}$$

where r is a random variable drawn from a uniform distribution over the interval [0, 1).

Each broker agent executes continuously the algorithm.

$M \leftarrow \{i \in T..|xi = 1\}i \in [1, \ldots, n]\}$ where n: the number of roles. M: migratable roles.

Adaptation in this approach is made by adopting a multi-criteria method. Explicit constraints are introduced, which can be varied, conflicting and of different scales. They are combined to constitute a result of all constraints with only a single criterion. In the example of broker agents, some criteria can be considered such as:

- $(f1)$: determines for each neighbor of 's' whether or not they provide a minimal number of tourist services considered as a minimal threshold.
- $(f2)$: calculates for each neighbor of 's' the number of tourist services provided which differs with those provided by the agent source's'.
- $(f3)$: determinates for each neighbor of 's' if they still have a free memory or not.
- $(f4)$: calculates for all neighbors of 's' how many roles each one plays to compare between them after.

In order to choose the candidate's role for migration from an agent source to an agent destination, a relation of dominance is applied according to multi-criteria specified before. The same relation is used to determine between neighbors of the source, which one will be the agent direction.

An evolution preserving the organization's coherence requires coherence verification for some constraints (before role moving), and the coherence re-establishing for others (after role moving).

As a preventive coherence verification

- The role $ri/i = \{1, 2, 3\}$ is delegated from an initial agent source 's' $= B_x$ to another agent B_y neighbor of B_x if that agent B_y provides a minimal number of tourist services. This restriction imposes the condition that an agent must provide a minimal number of tourist services (considered as a minimal threshold guarantee) in order to play a role *ri*.
- The role $ri/i = \{1, 2, 3\}$ is delegated from an initial agent source's' $= B_x$ to another agent B_y neighbor of B_x if the number of tourist services provided by agent B_y which is different than the one provided by agent B_x is higher than a threshold that is fixed before starting the mechanism adaptation.
- Each agent has a maximum number of roles that he can play. He cannot receive more roles than this maximum. This restriction aims to have equilibrium in role allocation.

As corrective constraints:

- After the migration of each role, relations between agents must be updated to maintain the MAS correct at any moment. As an example:

Plays $(B5, r1)$: $B5$ plays role $r1$, Plays $(B3, r2)$: $B3$ plays role $r2$ and $(r2, r1)$: $r2$ played by $B3$ is in relation with $r1$ played by $B5$.If $r1$ migrate from $B5$ to B6, this implicates: first to delete relation $(r2, r1)$ from $B3$ to $B5$, then add new relation $(r2, r1)$ from $B3$ to $B6$.

Criterion calculate procedure

```
to calculatecriterion
 set agid who
 set n count rolles-here
 ask rolles-here [set listeF (list )]
 foreach sort lienag-neighbors [
    ask ? [ set agid1 who
    ask  agent agid  [ foreach sort rolles-here with
                [xi = 1] [ ask ? [ set listeF lput agid1 listeF ] ]]
    ifelse (touristinfo > 30) [ ask  agent agid  [ foreach sort
  rolles-here with [xi = 1][ ask ? [ set listeF lput 1 listeF   ] ] ] ]
                          [ ask  agent agid  [ foreach sort rolles-here with [xi = 1][ ask ? [ set listeF
lput 0 listeF   ] ] ] ];;F1
     set nbrdiff 0
     foreach liste [ set temp2 ?
                  set trouve false
                  ask agent agid  [ foreach liste   [ if (? = temp2) [ set trouve true]]]
     if (trouve = false) [ set nbrdiff (nbrdiff + 1)]]
     ifelse ( nbrdiff > 5 ) [ ask agent agid[ foreach sort rolles-here with [xi = 1][ ask ? [ set listeF lput
1 listeF  ] ] ] ]
                        [ ask agent agid[ foreach sort rolles-here with [xi = 1][ ask ? [ set listeF lput
0 listeF  ] ] ] ] ;;F2
       ifelse ( (capacite - touristinfo) > 30) [ask agent agid [  foreach sort rolles-here[ ask ? [ if (xi =
1) [set listeF lput 1 listeF]  ] ]  ] ]
                                          [ask agent agid [  foreach sort rolles-here[ ask ? [ if (xi
= 1) [set listeF lput 0 listeF]  ] ]  ] ] ;;F3
       ifelse( n > count rolles-here )  [ ask agent agid [  foreach sort rolles-here[ ask ? [ if (xi = 1) [set
listeF lput 1 listeF]  ] ]  ] ]
[ ask agent agid [  foreach sort rolles-here[ ask ? [ if (xi = 1) [set listeF lput 0 listeF]  ] ]  ] ] ;;F4]] end
```

Discussion

After role's affectation to agents, the number of links between agent '***liensag***' and the number of links between roles '***liens***' is similar. The number of links between agents is always upper or equal to the number of links between roles and this is all along the system's evolution. In consequence of adding new roles to our environment, the number of links between '***liensag***' agents increases always with a rate inferior or equal to the number of links between roles increases '***liens***'. The number of '***liensag***' will reach its maximum (as shown in Fig. 9) after adding continuously roles to the system, reaching this level '***liensag***' will stay stable even if '***liens***' do not stop rising. If the number of roles is stable, the number of '***liens***' will stay fixed while '***liensag***' can increase, decrease or stay steady but always inferior or equal to '***liens***'.

Emergent behavior

Figure 10 shows the roles rate migration curve. After several iterations and tests, it is noticed that the role's migration mechanism has the tendency to prefer migrating roles to broker agents who have more links with the other agents of the broker society. While the quantity of information stored by broker agents about services offered by providers, available storage space and a number of different providers' services stored by the agent do not really influence the emergence of role's reorganization. It is also observed that the system stays steady (no emergent behavior) in case where broker agents have more or less the same number of links with the other agent of the broker organization.

Repeated tests (as in an example seen in Fig. 11) demonstrate that the dominant agent is generally the one who has more links with the other organization's agent. The dominant relation and multi criterion adaptive mechanism conducts the broker agent system to a behavior, which favors the agent having the most contacts in comparison with his colleagues of the same organization; Table 4 shows simulation statistics results. This behavior can be qualified by emergence because agents do not have any conscious or information about the neighbor's contact number with the other agent's organization. This parameter is not included as explicit criteria in making the choice of who will receive the role candidate to migration. This behavior is a response of the broker organization to changes in the environment; it is a kind of collective intelligence of the broker organization. Rightly, this emergent behavior permits a better performance for our application since the agent who has more contacts has more chances to satisfy the users' request than the other broker agents do.

The instruction: "**observer> ask agents [show count lienag-neighbors]**" gives order to every broker agent to show how many links he has with other broker neighbors.

The instruction: "**observer> ask agents [show count rolles-here]**" gives order to every broker agent to show how many roles assigned to him to be accomplished.

At the end of the test iterations in Fig. 10, it is clearly shown that all unsatisfied requests migrated to the non-dominate BA who is identified as (agent 6) with "17" request's migrations. (Agent 6) is one of the brokers with the highest number of links with the other organization's agent equal to "6" links. This behavior maximizes the probability of users request satisfaction, so a more profitable application.

Nowadays, many demands have emerged. Cloud requirements lead to build a system able to support efficiently a complete collaboration between different cloud providers

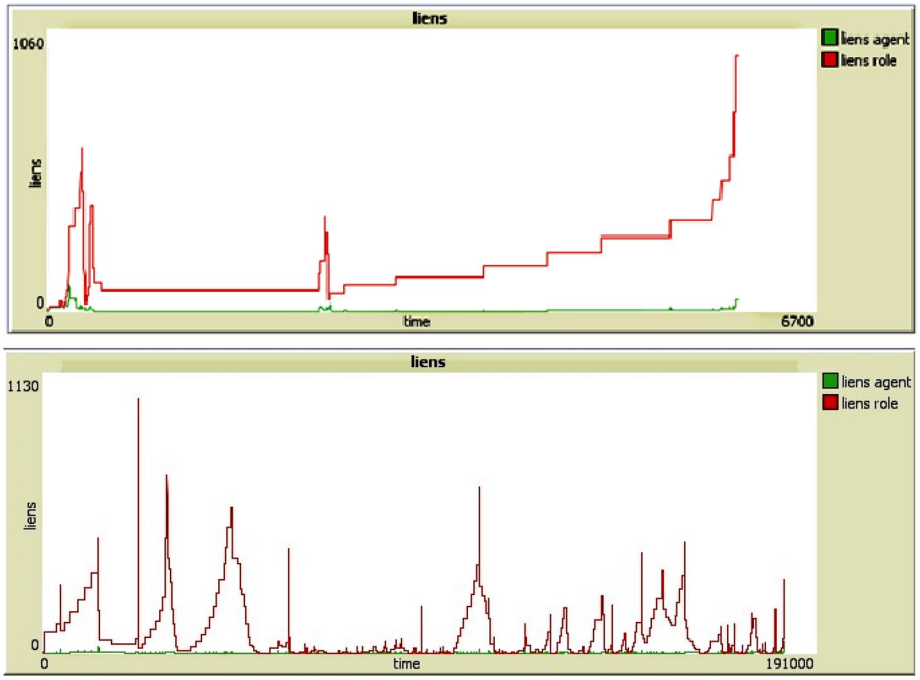

Fig. 9 *Agent's links* and *role's links* comparison curve

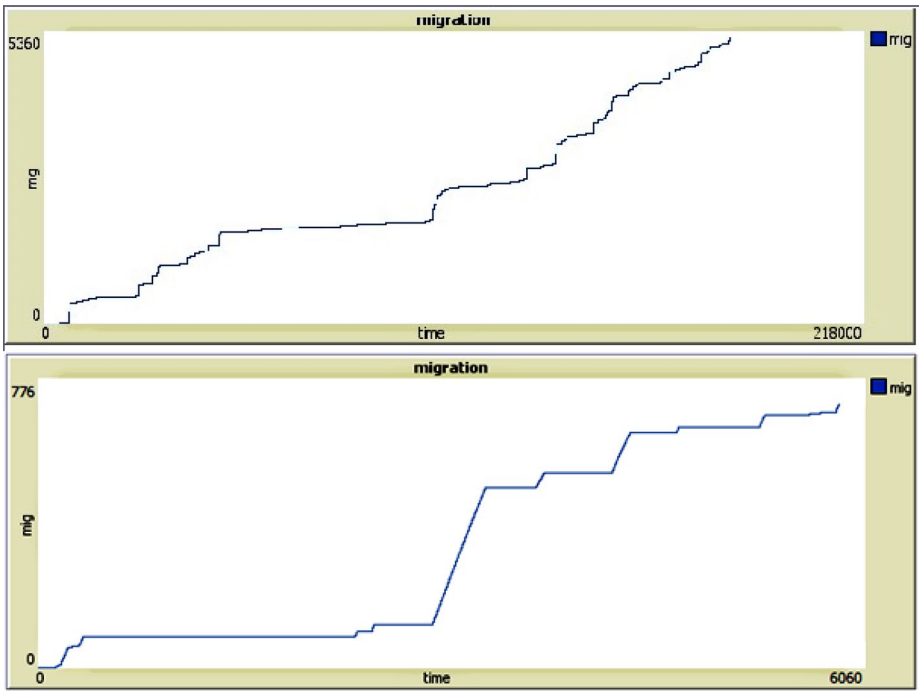

Fig. 10 Roles rate migration curve

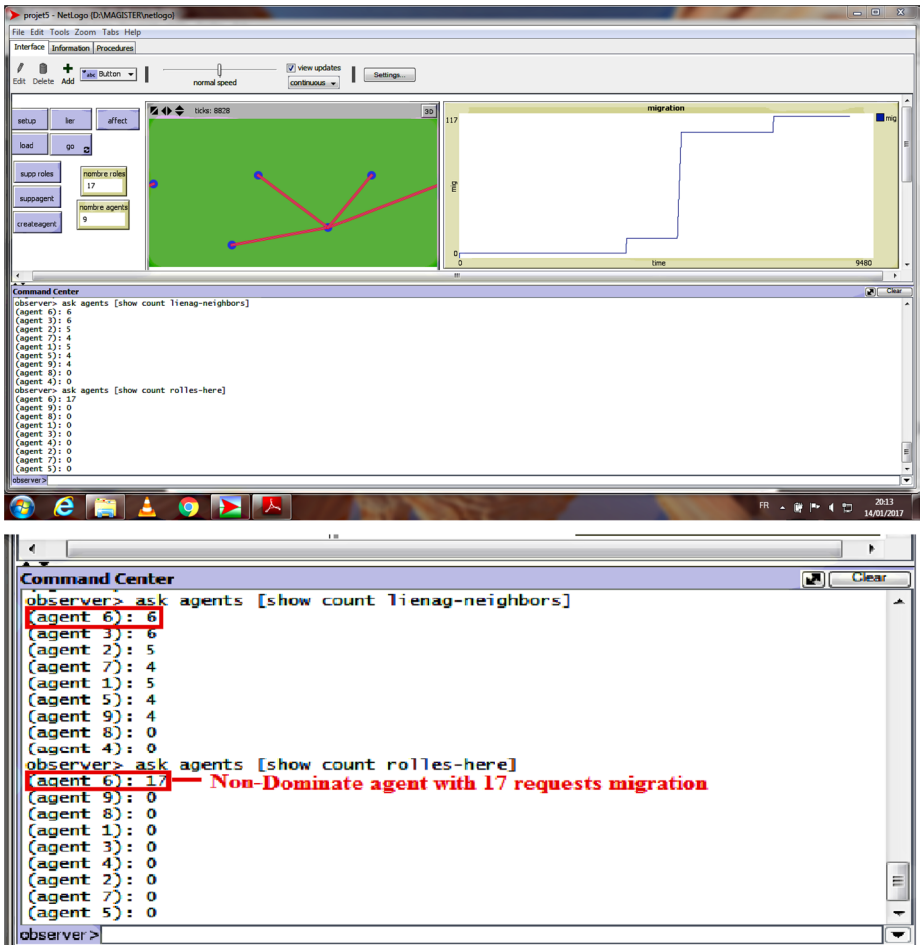

Fig. 11 Test example results

Table 4 Extensive simulation statistics results

1000 Repeated tests	Emergent behavior	Non emergent behavior	Static organization
Migration mechanism (%)	85	8	7
Observation	Intelligent behavior	Other behavior	No behavior

focusing on various aspects of the federation like ensuring flexibility, a multi criterion adaptive mechanism and coherent evolution features of clouds federation. AMACE, opposite to works seen before, cares more about maintaining run-time coherence in optimization evolution: preventive and corrective constraints are verified before and after the adaptation mechanism execution to guarantee at all moments a coherent system during system execution. Comparatively to the existing works, the main characteristics of our strategy are:

- Adopting a flexible system model supporting entrance or departure of new providers, to or from the clouds federation.

- Large coverage of all the system's sides, optimization of no fixed number of various ranges and conflicting criteria during the adaptive mechanism.
- Caring about maintaining run-time coherence in optimization evolution to guarantee at all moments a coherent system during system execution.

Conclusion and future works

Actually, business is a very dynamic environment. It is considered in this paper an open clouds federation environment that is in constant evolution. AMACE supports a flexible model integrating interactions between BA organization, new cloud providers can enter the market and offer their new services while other cloud providers can leave it. BA's interaction permits request's migration between BA neighbors to delegate unsatisfied costumer requests. An adaptive approach is proposed to support the presented flexible model.

AMACE makes adaptation and hybridization of two approaches to deal with dynamic federated cloud computing. The adaptive mechanism is inspired from an organization's optimization approach for self-organization in broker agent organization [16] and negotiation protocol communication from Multi agent resource allocation approach on federated clouds [1]. AMACE is auto adaptive and multi criterion, which can take in account various parameters (i.e. computing the load balance of mediator agents and the geographical distance "network delay" between the costumer and provider…).

Looking in the future to implement the whole system with the negotiation protocol communication to get empirical results and demonstrate AMACE effectiveness in addition to trying to provide new features in open federated clouds accomplishment.

Abbreviations

AMACE: agent based multi-criterions adaptation in cloud environment; MAS: multi agent system; MARA: multi-agent resource allocation; CA: consumer agent; BA: brokering agent; PA: provider agent; SLA: service level agreements; KM: knowledge management; ARAM: agent-based resource allocation model.

Authors' contributions

Authors contributed in various important aspects. All authors read and approved the final manuscript.

Author details

[1] LIRE Laboratory, Computer Science Department, University of Constantine2-Abdelhamid Mehri, 25000 Constantine, Algeria. [2] TECHNE Labs, University of Poitiers, 1 Rue Raymond Cantel, 86073 Poitiers Cedex 9, France.

Acknowledgements

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Ethics approval and consent to participate

Not applicable.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 22 April 2018 Accepted: 27 August 2018
Published online: 12 September 2018

References

1. Haresh MV, Kalady S, Govindan VK (2011) Agent based dynamic resource allocation on federated clouds. Recent advances in intelligent computational systems (RAICS). Trivandrum, IEEE, pp 111–114

2. Xiaotong W, Meng L, WanChun D, Longxiang G, Shui Y (2016) A scalable and automatic mechanism for resource allocation in self-organizing cloud. Peer-to-Peer Netw Appl 9(1):28–41
3. Jieun C, Yoonhee K (2017) Adaptive resource provisioning method using application-aware machine learning based on job history in heterogeneous infrastructures. Cluster Comput 20(4):3537–3549
4. Singh H, Randhawa R (2018) ARMM: Adaptive Resource Management Model for Workflow Execution in Clouds. In: Negi A, Bhatnagar R, Parida L (eds) Distributed Computing and Internet Technology ICDCIT 2018. Lecture Notes in Computer Science, vol 10722. Springer, Cham, pp 315–329
5. Ravandi B, Papapanagiotou I (2018) A self-organized resource provisioning for cloud block storage. Future Gen Comput Syst 89:765–776
6. Ghobaei-Arani M, Jabbehdari S, Ali Pourmina M (2018) An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach. Future Gen Comput Syst 78(1):191–210
7. Xu J, Cao J (2014) A broker-based self-organizing mechanism for cloud-market. International Federation for Information Processing (IFIP). Springer, Berlin, pp 281–293
8. Chaabouni T, Khemakhem M (2013) Resource management based on Agent technology. In: Proceeding of IEEE on cloud computing, NOORIC, pp 372–375
9. Kecskemeti G, Maurer M, Brandic I, Kertesz A, Nemeth ZS, Dustdar S (2012) Facilitating self-adaptable inter-cloud management. In: The 20th Euromicro international IEEE conference on parallel distributed and network-based processing, (PDP), pp 575–582
10. Patel K S, Sarje A K (2012) VM provisioning policies to improve the profit of cloud infrastructure service providers. In: Proceeding of the 3th international conference on computing communication and networking technologies (ICCCNT'12), Coimbatore, India, IEEE, pp 1–5
11. Comi A, Fotia L, Messina F, Pappalardo G, Rosaci D, Sarné GML (2015) An evolutionary approach for Cloud learning agents in multi-cloud distributed contexts. In: The 24th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE'15), IEEE, Cyprus, pp 99–104
12. Manvi SS, Birje MN, Prasad B (2005) An Agent-based Resource Allocation Model for computational grids. Multi-Agent Grid Syst Int J 1(1):17–27
13. Lee YH, Huang KC, Shieh MR, Lai KC (2017) Distributed resource allocation in federated clouds. J Supercomput 73(7):3196–3211
14. Ishikawa T, Fukuta N (2015) Federated cloud-based resource allocation by automated negotiations using strategy changes. In: Fujita K, Ito T, Zhang M, Robu V (eds) Next frontier in agent-based complex automated negotiation. Studies in computational intelligence, vol 596. Springer, Tokyo, pp 111–125
15. Zulkar Nine Md SQ, Abul Kalam Azad Md, Abdullah S, Ahmed N (2015) Dynamic load sharing to maximize resource utilization within cloud federation. In: CloudCom-Asia, Springer International Publishing Switzerland, pp. 125–137
16. Petri I, Rana OF, Beach T, Rezgui Y (2017) Performance analysis of multi-institutional data sharing in the Clouds4Coordination system. Comput Elec Eng 58:227–240
17. Carlini E, Coppola M, Dazzi P, Mordacchini M, Passarella A (2016) Self-optimising decentralised service placement in heterogeneous cloud federation. In: 10th IEEE international conference on self-adaptive and self-organizing systems (SASO), pp 110–119
18. Son S, Kang DJ, Huh SP, Kim WY, Choi W (2016) Adaptive trade-off strategy for bargaining-based multi-objective SLA establishment under varying cloud workload. J Supercomput 72(4):1597–1622
19. Montes JD, Rodero I, Zou M, Parashar M (2014) Federating advanced cyber infrastructures with autonomic capabilities. In: Cloud computing for data-intensive applications. Springer, New York, pp 221–227
20. Andronico G, Fargetta M, Monforte S, Paone M, Villari M (2014) A model for accomplishing and managing dynamic cloud federations. In: 7th international conference on utility and cloud computing, IEEE/ACM, pp 744–749
21. Hou F, Mao X, W. Wu W, Liu L, Panneerselvam J (2014) A cloud-oriented services self-management approach based on multi-agent system technique. In: Proceedings of the 7th international conference on utility and cloud computing, IEEE/ACM, pp 261–268
22. De Meo P, Messina F, Sarné Rosaci D (2015) An agent-oriented, trust-aware approach to improve the QoS in dynamic grid federations. Concurr Comput Practice Exp 27(17):5411–5435
23. Wilensky U, Netlogo (1999) Center for connected learning and computer-based modeling. Northwestern University, Evanston, IL, NetLogo, http://ccl.northwestern.edu/netlogo

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com