A genetic algorithm based approach for fluctuating QoS aware selection of IoT services

Karima Khadir, Nawal Guermouche, Amal Guittoum, Thierry Monteil

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication 2022

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A genetic algorithm based approach for fluctuating QoS aware selection of IoT services

KARIMA KHADIR[1], NAWAL GUERMOUCHE[2], AMAL GUITTOUM.[3,4], AND THIERRY MONTEIL[5]

[1]EXPLEO GROUP, Montigny-Le-Bretonneux, France
[2]LAAS-CNRS, University of Toulouse, INSA, Toulouse, France
[3]Orange Lab, Meylan, France.
[4]Ecole National Supérieure d'Informatique, Algiers, Algeria
[5]IRIT, University of Toulouse, INSA, Toulouse, France

Corresponding author: Nawal GUERMOUCHE (e-mail: nawal.guermouche@laas.fr).

ABSTRACT In recent years, the Internet of Things (IoT) has evolved at an exceptional speed, which enables to interconnect a very large number of heterogeneous, distributed, and mobile devices. This number will exceed 70 billion by 2025 according to Statista[a]. Therefore, with this huge amount of connected objects, the fulfillment of complex IoT applications, which usually requires a combination of several IoT objects, remains a real challenge. Besides, several requirements of Quality of Service (QoS) must be fulfilled, which makes the problem of selecting the appropriate IoT services NP-hard.

In the literature, two main techniques for QoS-driven service selection are proposed: global selection characterized by a poor performance in dynamic and distributed huge environments and local selection which considers pre-defined local QoS constraints. Mainly, the existing works consider static QoS. However, in real life scenarios, QoS of IoT services can be fluctuating. To enhance the reliability of IoT applications, it is of paramount importance to consider the fluctuation dimension. In this context, we propose a QoS fluctuation-aware selection approach of IoT services. To do so, we propose a near-to optimal distributed approach that relies on decomposing the global QoS into distinct local constraints that serve as upper/lower bounds for selection while enhancing the reliability of the resulting composition by considering the QoS fluctuation of the candidate IoT services. The approach, we propose is based on a multi-objective evolutionary algorithm (MOEA) to solve the global QoS decomposition problem. Then a local selection using the obtained local constraints is performed in a parallel and distributed way.

The performance of the proposed approach is evaluated and validated via experiment series.

[a]https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

INDEX TERMS QoS fluctuation, Global QoS Decomposition, IoT Services, MOEA, Local Selection.

I. INTRODUCTION

THE Internet of Things (IoT) is defined as a complex infrastructure composed of a huge number of heterogeneous and ubiquitous devices. It relies on Information and Communication Technologies (ICT) and embedded systems to interconnect these devices that allows them to exchange their data to provide a variety of value-added services for several types of IoT applications. The Web of Things (WoT) enables the implementation of this vision by relying on the Web to virtualize heterogeneous IoT devices in a uniform representation, called *Avatar*, using a semantic description. An avatar is endowed with standalone reasoning, context management, collaboration capabilities and can be deployed in Fog-Cloud infrastructures [1]. The IoT services provided by avatars can be discovered, selected, and composed to fulfill a given IoT need, represented as a complex process composed of a set of abstract tasks.

The selection of IoT services is still a challenging problem. Several works have been proposed to cope with QoS-aware selection problem. Mainly, these approaches consider

that QoS parameters are static and ignore their variation through time and their potential impact on the built IoT services combination. However, IoT environments are highly dynamic where IoT devices can be mobile and can have limited resources. Consequently, QoS parameters of IoT services can change over time (i.e., they can be fluctuating). In this context, it is of paramount importance to consider the fluctuation dimension of QoS parameters when selecting IoT services. This will enhance the reliability of the selected IoT services and should decrease the impact of run time deviations. Moreover, with the proliferation of IoT devices and their services, it is essential to fulfill the targeted IoT needs efficiently even in huge systems.

In this paper, we propose a new distributed fluctuation-aware selection approach for IoT services. The goal is to satisfy efficiently an IoT need, specified as a an abstract process with global QoS constraints, by selecting a relevant combination of IoT services while considering the potential fluctuation of their QoS parameters. To do so, a Multi-Objective Evolutionary genetic based Algorithm (MOEA) for decomposing the global QoS constraints into local QoS constraints, associated to each process's task, is proposed. Then, local and parallel selection is performed, which is suitable for huge and complex IoT systems.

The proposed approach extends our previous work proposed in [2]. In this work, we have presented a new distributed avatar discovery approach that relies on social networking mechanisms and fuzzy c-means clustering algorithm. Social networking allows to regroup the avatars most susceptible to coordinate and accomplish the requested IoT need focusing on social measures as location and interests. Whereas the clustering algorithm classifies the social avatars into several clusters according to their functionalities to guide at the best the forwarding of discovery requests that meet the objective's of the awaited application. This step enables to pre-filter the search space and allows to consider relevant candidate discovered services. Based on the result of the discovery step, a selection approach is required to select the appropriate avatars to fulfill functional and QoS properties where considering fluctuating QoS parameters.

The rest of the paper is recognized as follows. Section II discusses the related works. A motivating use case is given in Section III. The approach overview is illustrated in Section IV. Section V details the semantic modeling of heterogeneous IoT objects and the IoT application to be accomplished. In Section VI, the IoT services discovery approach is given. The local selection approach based on the decomposition of the global QoS constraints is presented in Section VII. Experimental results are discussed in Section VIII. Finally, Section IX concludes the paper.

II. RELATED WORK

QoS-aware service selection problem has widely attracted attentions in multiple scientific researches and industrial developments. The approaches proposed in the literature are classified into two categories: (1) approaches based on global selection and, (2) approaches based on local selection.

The first category aims to find the optimal combination. It includes: the works using exhaustive algorithms such as Mixed Integer Linear Programming (MILP) methods [3], global planning algorithm [4] and WS-IP algorithm [5].

In [3], the authors use the MILP method to select the best matching services in the SaaS services library relying on Skyline computation. In [4], Rajeswari and al. have adopted global planning to find all the eligible service combinations and select the composition with the highest score. Concerning the WS-IP algorithm, in [5], the QoS selection problem is modeled as a 0-1 IP problem and uses the branch-and-bound method the find the optimal composition.

These works enable to find the optimal combination, however, they are NP-hard and their time complexity is exponential which makes them non-scalable. To address this issue, several works based on approximate methods using meta-heuristic algorithms that aim to select the near-to-optimal combination, have been proposed. In [6], a hybrid Honey Bees (hHBA) Mating optimization algorithm for selecting the near-to-optimal solution in semantic Web Service Composition is proposed. An improved Particle Swarm Optimization Algorithm (iPSOA) is provided in [7] to solve the QoS-aware selection problem. It proposes several improvements as the application of a Non-Uniform Mutation (NUM) strategy to the global best particle to enhance the population diversity and to overcome the prematurity of standard PSOA, the Adaptive Weight Adjustment (AWA), and Local Best First (LBF) strategies to improve the convergence speed. In [8], the authors propose a genetic algorithm (GA) for global QoS-aware service composition in cloud computing combined with data mining clustering techniques to reduce the search space. Although these works based on meta-heuristic algorithms achieve good performance with regard to exhaustive methods, however, they rely on a central manager, which suffers from a single point of failure. Moreover, these works assume static QoS parameters.

Alrifai and al. [9] have proposed an interesting approach for decomposing global QoS constraints considering the Sequential, Loop, Parallel, and Conditional pattern structures. Their method starts by dividing the value range of each QoS parameter into multiple discrete values (quality levels) which is used as QoS candidate local constraints. Then, they use the MILP technique to solve the optimization problem to find local constraints. However, this method suffers from scalablity particularly when the number of candidate services and QoS constraints is relatively large.

A new heuristic method that divides the decomposition problem into multiple sub-optimal ones is proposed in [10]. First, the initial solution which meets all global constraints is defined. Then, it is updated gradually until finding a near-to-optimal result. In [11], an adaptive global QoS constraint decomposition approach based on Fuzzy logic technology and Cultural Genetic Algorithm (CGA) has been proposed. This method aims to find near-to-optimal local constraints

TABLE 1. A synthesis of the studied works

Reference	Selection type	Method	Constraints decomposition	Quality levels calculation	Scalability	Fluctuation
[3]	Global	MILP	No	Centralized	No	No
[4]	Global	Global planning	No	Centralized	No	No
[5]	Global	WS-IP	No	Centralized	No	No
[6]	Global	hHBA	No	Centralized	Yes	No
[7]	Global	iPSOA	No	Centralized	Yes	No
[8]	Global	GA	No	Centralized	Yes	No
[9]	Local	MILP	Yes	Centralized	No	No
[10]	Local	Heuristic	Yes	Centralized	Yes	No
[11]	Local	CGA	Yes	Centralized	Yes	No
[12]	Local	CGA	Yes	Centralized	Yes	No

for selecting services during the running time. In [12], a novel hybrid intelligent algorithm called Culture Genetic Algorithm (CGA) is proposed.

Despite the proposed QoS-aware selection works enhance the performance of the selection problem, neither of them has considered the potential dynamic nature of QoS parameters of IoT services. The fluctuation of the QoS attribute values is ignored. Besides, these works rely on a central entity that fulfills the selection. This present the limit of a single point of failure. Also, these approaches does not rely on pre-processing step to reduce the search space, which is restrictive in huge systems.

To summarize, Table 1 gives an overview of the studied works and compares them according to six properties: **Selection type:** to indicate whether the work is based on the global or local selection category, **Method:** to designate the used method for resolving the selection optimization problem, **Constraints decomposition:** to indicate if the work adopts the global QoS constraints decomposition approach for services selection, **Quality levels calculation:** to specify how quality levels are calculated for methods based on global QoS constraints decomposition, **Scalability:** to indicate if the proposed method is scalable or not, and **Fluctuation:** to indicate if the work considers the QoS fluctuation or not.

In this paper, we propose a new decentralized selection approach for IoT services where QoS parameters can change over time (i.e., can be fluctuating). This approach relies a genetic algorithm for decomposing the global QoS constraints into local ones so that local selection can be fulfilled. This is particularly efficient, particularly in huge systems. This approach extends our previous work where we have proposed a discovery approach based on social networking mechanisms and clustering stage which enables to reduce the search space and restricts candidate services to social neighbors that are likely relevant to the targeted collaboration.

III. MOTIVATING USE CASE

To better illustrate the proposed discovery and QoS-aware selection approaches for IoT services, a motivating scenario of overtaking assistance system of connected vehicles is given in this section. In this scenario, we assume that the target vehicle, i.e., the one that seeks to overtake, is not equipped with all the required IoT devices that provide the visibility necessary for overtaking. It is also assumed that around the vehicle, in the smart city, there are a large number of IoT objects that can provide different QoS parameters that can change over time (i.e., they can be fluctuating). Therefore, the vehicle looks for finding (discovering) and selecting the appropriate IoT devices which can collaborate with it to accomplish the process of overtaking in a distributed way. The different devices are represented via their avatars that expose their corresponding IoT services with different and fluctuating QoS values. The overtaking process is subject to global end-to-end QoS requirements. For this example, we consider two QoS attributes :

- Response time: average execution time between sending requests and receiving its response.
- Throughput: total number of successful invocations (i.e., executed requests) for a given period.

We suppose that our scenario evolves in the context given in Figure 1:

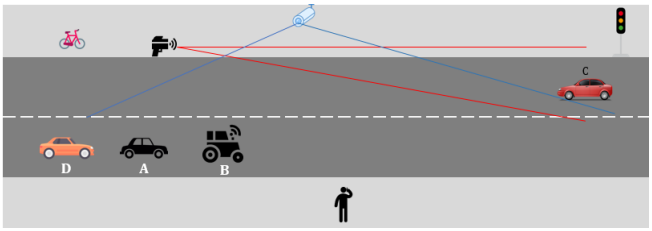

FIGURE 1. Overtaking use case situation

In this scenario, we assume that the target vehicle A, which wants to overtake the tractor B which rolling at low speed, doesn't have the perception mechanisms as cameras to perceive its environment. So, to perform the overtaking safely, vehicle A needs to collaborate with the devices of its environment. In such a scenario, we suppose that there are: a radar, a lidar and a camera installed by the roadside, a bicycle rolling beside, the pedestrian who uses a smartphone, a traffic light a few meters from host vehicle, the drivers of A, B, C and D have smartphones. A Global Navigation Satellite System (GNSS) is also available for vehicles tracking and an RSU (Road-Side Unit) which covers the region in which the host vehicle is located. We suppose also that the travel speed

of the host, front, rear and oncoming vehicles (A, B, D, and C respectively) is constant.

As example of avatars, we suppose that the vehicle D has an avatar that offers services which allow for instance to measure a distance with a given car, measure the speed, and perform analysis to check for example if the required safety distance is respected. For instance, the execution history of the response time of the avatar A_4 are 88, 7, 7, 51, 84, 71.

So, our goal is to select a close-to-optimal combination of avatars while considering the potential fluctuation of their QoS so that global QoS constraints are satisfied. An extract of the overtaking abstract process is illustrated in Figure 2.

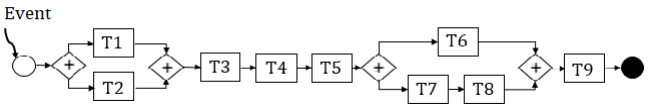

FIGURE 2. Overtaking abstract process

The beginning of the overtaking scenario is triggered when the driver of the target vehicle turns on the left-blinking to indicate his intention to overtake. At first, the driver tries to find out if the rear vehicle is already overtaking (T1) and if the front vehicle is too (T2). If so, the overtaking is not possible, otherwise, other conditions must be checked. The distance from the front vehicle is measured (T3), and if it is greater than 30 meters, the speed of this front vehicle is also measured (T4). After that, the speed difference between the target and the front vehicles is calculated (T5). If the target vehicle is traveling at 20km/h faster than the rear vehicle, an accelerated type overtaking is carried out. In order to ensure this, the distance available for overtaking is measured (T6) and minimum safety distance required for safe overtaking is calculated using the formulas given in [13] (T8). To perform the task (T8), the speed of the oncoming vehicle is required (T7). If the available distance is greater then the minimum required safety distance (T9), then, a message authorizing the driver to overtake is displayed.

We suppose also that the travel speed of the host, front, rear and oncoming vehicles (A, B, D, and C respectively) is constant.

IV. APPROACH OVERVIEW

As depicted in Figure 3, the end-to-end proposed approach to satisfy a given complex IoT application, specified as an abstract process, is handled via four main steps:

- **Semantic specification** : to model the awaited IoT need (i.e., IoT application) to fulfill and the heterogeneous IoT objects which are virtualized on the Web through autonomous avatars.
- **IoT services discovery** : to discover and to identify the most appropriate avatars that can collaborate to accomplish the IoT application based on social networking and clustering algorithms.
- **IoT services selection** : to select the near-to-optimal combination from the discovered avatars based on their QoS properties.
- **Execution plan building** : that allows to construct the final execution plan according to the abstract process.

V. SEMANTIC SPECIFICATION

In this section, we detail the concept of autonomous avatar and we describe the specification of the awaited IoT applications [1].

A. AVATAR DESCRIPTION

An avatar is defined as a semantically enriched virtual abstraction of a physical or software entity on the Web. Unlike previous works [14] [15] [16] which consider an avatar as a passive simple service provider, in this work, we extend it with autonomous reasoning capabilities. This enables avatars to be active in making decisions according to required tasks.

An avatar A_i is described by a tuple (S_i, F_i, Q_i, C_i), where:

- $S_i = \{s_i^1, s_i^2, .., s_i^b\}$ represents the services that A_i provides, i.e, its capabilities,
- $F_i = \{f_i^1, f_i^2, .., f_i^a\}$ is a set of functionalities that represent a generic description of what the services can achieve to provide a given service.
- $Q_i = \{q_i^1, q_i^2, .., q_i^r\}$ represents its QoS requirements, such as: r is the number of QoS attributes.
- $C_i = (I_i, L_i, O_i)$ is the social context of A_i which includes : **1)** a set of interest centers modeled via a vector $I_i = \{I_i^1, I_i^2, .., I_i^c\}$ where : c is the number of interest subjects and each I_i^j represents the interest degree for the j^{th} subject with a real value in [0,1], such as : a great value of I_i^j it indicates that A_i is largely involved in the works around the j^{th} subject, **2)** a location defined with longitude and latitude of the device represented $L_i = (LongA_i, LatA_i)$, and **3)** the owner of the object O_i.

The QoS attributes can be either positive/negative and static/dynamic. The positive parameters refer to attributes to be maximized, while negative ones are to minimize. Static parameters have a fixed value and don't vary through time, while dynamic attributes are the opposite. Each avatar QoS parameter represents the aggregation min/max of the QoS parameters of its services taking into account its type.

***Example** 1:* Let as consider the overtaking use case introduced in section III. Table 2 gives an overview of the set of candidate avatars A_i, their services S_{ij} for each abstract task of the overtaking process T_j, such as for each service S_{ij}, the avatar A_i can perform the abstract task T_j.

The set of interests depicted in the table are: I_1 is Transport, I_2 is Weather, I_3 is Perception, I_4 is Obstacle, I_5 Com-

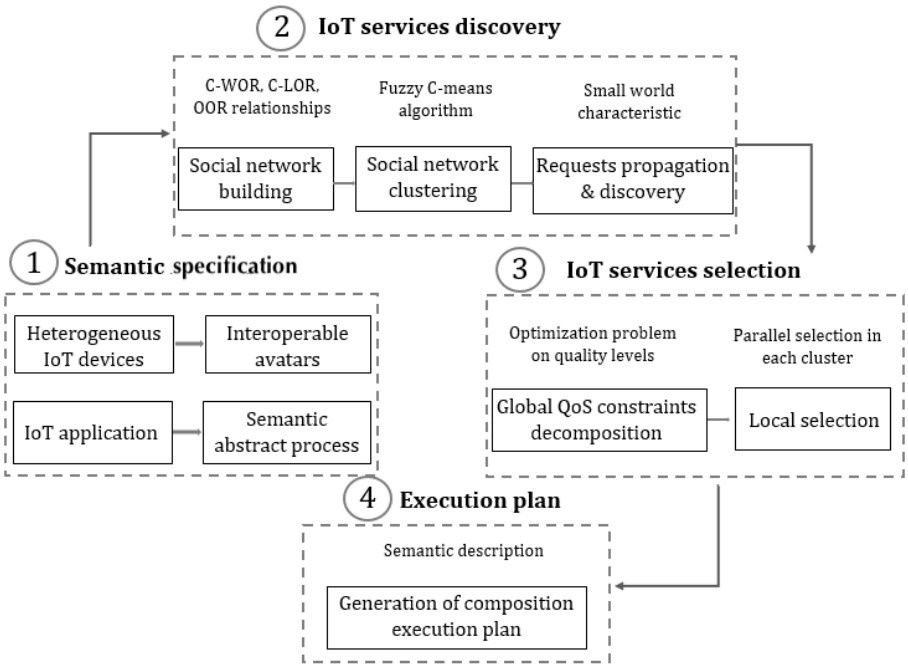

FIGURE 3. Approach overview

TABLE 2. Candidate avatars services of each abstract task

Avatar	Services	Interests	Location	Owner	Features
A_1: Vehicle A	S_{13},S_{14},S_{15}, S_{18},S_{19}	I_1(0.6),I_2(0.2), I_4(0.6)	43.57834 1.441185	$User_1$	f_2,f_3, f_4, f_5,f_6
A_2: Vehicle B	S_{22},S_{24},S_{25}, ,S_{28},S_{29}	I_1(0.9),I_2(0.3), I_3(0.5),I_4(0.6)	43.578291 1.441146	$User_2$	f_2,f_3, f_4, f_5,f_6
A_3: Vehicle C	S_{35},S_{38},S_{39}	I_1(0.9),I_2(0.3), I_3(0.4),I_4(0.5)	43.578074 1.441052	$User_3$	f_2,f_3, f_4, f_5
A_4: Vehicle D	S_{41},S_{45},S_{47}, S_{48},S_{49}	I_1(0.9),I_2(0.3), I_3(0.4),I_4(0.5)	43.578074 1.441052	$User_4$	f_2,f_3, f_4, f_5,f_6
A_5: Bike	/	I_1(0.9)	43.578315 1.441277	$User_5$	f_5,f_6
A_6: RSU	/	I_1(0.7),I_5(0.7), I_7(0.5)	43.577163 1.440768	$User_6$	f_5,f_6
A_7: Radar	S_{73},S_{74},S_{76}, S_{77}	I_1(0.7),I_4(0.9)	43.57827 1.4412062	$User_6$	f_2,f_3, f_5
A_8: Camera	S_{81},S_{82},S_{83}, S_{86}	I_1(0.6),I_2(0.4), I_3(0.9),I4(0.7)	43.578405 1.441300	$User_6$	f_1,f_2, f_5
A_9: Traffic light	/	I_1(0.7),I_8(0.9)	43.577899 1.440873	$User_6$	f_5
A_{10}: GNSS	S_{101},S_{102},S_{103}, S_{104},S_{106},S_{107}	I_1(0.8),I_4(0.7) I_7(0.9)	Server	$User_7$	f_1,f_2, f_3, f_5
A_{11}: Street lighting	/	I_1(0.2),I_4(0.3) I_6(0.9),I_8(0.4)	43.57822 1.441251	$User_6$	f_5
A_{12}: Phone A	S_{123},S_{125}, S_{128},S_{129}	I_1(0.4),I_5(0.9) I_7(0.7)	43.57834 1.441185	$User_1$	f_4, f_6
A_{13}: Phone B	S_{132},S_{134},S_{135}, S_{138},S_{139}	I_1(0.4),I_5(0.9) I_7(0.7)	43.578291 1.441146	$User_2$	f_4, f_6
A_{14}: Phone C	S_{147}	I_1(0.4),I_5(0.9) I_7(0.7)	43.578074 1.441052	$User_3$	f_4, f_6
A_{15}: Phone X	S_{151},S_{152}, S_{156},S_{157}	I_1(0.4),I_5(0.9) I_7(0.7)	43.578332 1.441082	$User_8$	f_4, f_6
A_{16}: Camera	S_{161},S_{162},S_{163}, S_{166},S_{167}	I_1(0.6),I_2(0.4), I_3(0.9),I4(0.7)	43.49881 1.441300	$User_6$	f_1,f_2, f_5
A_{17}: Lidar	S_{174}, S_{176}	I_1(0.7),I_4(0.9)	43.57827 1.4412062	$User_6$	f_2,f_3, f_5
A_{18}: Phone D	S_{181}	I_1(0.4),I_5(0.9) I_7(0.7)	43.578074 1.441052	$User_8$	f_4, f_6

munication, I_6 is Luminosity, I_7 is Geolocalization and I_8 is Circulation.

The set of functionalities are: f_1 is the environment perception, f_2 is the distance measure, f_3 is the speed measure, f_4 is the computing capacity (calculation capabilities), f_5 is the moving and f_6 represents the communication (network equipment).

For example, for the avatar A_1 that represents the target vehicle, it can realize the services: S_{13} to measure the distance from the front vehicle, S_{14} to measure the speed of the front vehicle, S_{15} to calculate the difference between the speeds of target and front vehicles, S_{18} to calculate the safe overtaking distance, and S_{19} to verify if the available distance is greater then the required safety distance. It is described by five functionalities : f_2 for the distance measure, f_3 for speed measure, f_4 for the calculation, f_5 for moving, and f_6 for communication. A_1 is mainly interested in three subjects: I_1 for transport, I_2 for weather to collect meteorological measurements, and I_4 for the obstacle detection.

For each avatar, a history of its QoS parameters are gathered. Table 3 shows an example of the last six QoS values of a set of avatars.

The internal architecture of an avatar is given in Figure 4. It includes the following components:

1) Knowledge Base (KB): it gathers the knowledge related to the represented IoT device, and its environment as location independently of the knowledge of the other avatars. This component relies on a generic instantiated ontology called *AvatarOnt* and a series of objectives and processes that reflect the functional part of the avatar. This ontology is structured through four main

Avatar	Response time	Average	Throughput	Average
A_4	88, 7, 7, 51, 84, 71	51.33	8, 4, 21, 25, 25, 17	16.66
A_{10}	61, 72, 15, 19, 93, 72	55.33	5, 7, 9, 25, 3, 15	10.66
A_8	87, 7, 2, 59, 98, 58	61.83	26, 22, 27, 5, 21, 12	18.83
A_{16}	31, 54, 98, 67, 20, 12	47	12, 2, 26, 13, 18, 26	16.16
A_{15}	52, 61, 52, 35, 83, 48	55.16	1, 11, 20, 27, 5, 23	14.5
A_{18}	79, 2, 64, 91, 18, 56	51.66	14, 15, 19, 2, 3, 12	10.83

TABLE 3. QoS information

modules:

a) A service module: that describes avatar services operations, their inputs, and outputs using MSM[1] ontology, REST service invocation methods using hRests [17], and the non-functional QoS attributes using WSOnto [18].

b) A sensor module: to describe the sensors and their observations based on SOSA[2] and IoT-O[3]

c) An actuator module: to describe the actuators and the actions that it can perform on its devices relying on SAN[4] and IoT-O, and

d) An avatar profile module: to describe the avatar profile which represents the set of characteristics that define it. It includes: its identifier, its name, its objectives, its deployment node, and its social context (the location of the device represented, its owner, and its interests).

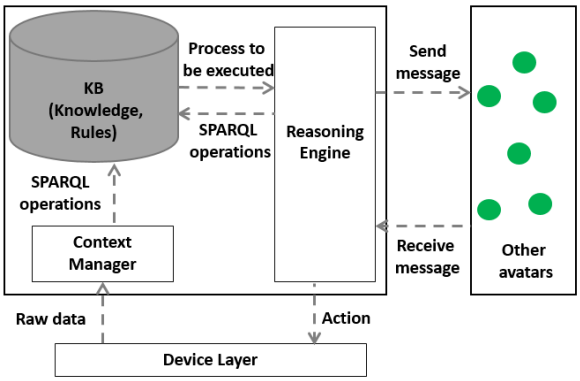

FIGURE 4. Avatar architecture

2) Context Manager: used to monitor and integrate changes that may occur at the level of the device represented by the avatar, i.e, its status or data (sensor measurement) or events coming from its environment (e.g., activation of the left-blinking by the driver). When changes or events are detected, this manager is responsible for updating the avatar knowledge base via SPARQL operations: creation, update and deletion.
3) Reasoning Engine: it takes care of the reasoning and decision-making functionalities. It allows analyzing the collected events in order to trigger the corresponding processes to be carried out. If the triggered process requires several IoT objects to be accomplished, this component is responsible for the discovery and selection of the relevant corresponding avatars to achieve the awaited process.

[1]http://iserve.kmi.open.ac.uk/ns/msm
[2]http://www.w3.org/ns/sosa/
[3]https://www.irit.fr/recherches/MELODI/ontologies/IoT-O
[4]http://www.w3.org/ns/ssn/

B. APPLICATION SPECIFICATION

An IoT application, specified by an abstract process, is defined as a series of multiple tasks that are of a composite or atomic type. These tasks are interconnected with each other via several operators: sequential (SEQ), parallel (AND), loop (Loop) or/and conditional (XOR) operators. Global QoS constraints can be associated to the process. The semantic description of a process relies on the ontology proposed in [19] while adding a node to indicate the state of the process execution: Waiting, Running or Pausing.

***Definition** 1: Abstract process*

An abstract process is modeled as a tuple $P = \{N, L, \gamma, E, W\}$ where :

- N : is a set of nodes which can be :
  - -- tasks $T = \{T_1, ..., T_n\}$
  - -- events $EV = \{EV_1, ..., EV_m\}$
  - -- operators $O = \{O_1, ..., O_g\}$
- $L \subseteq N * N$ is a set of edges.
- γ is a function that maps a node onto either an activity, event or a process flow (i.e., SEQ, AND, Loop, XOR).
- $E = \{E_1, E_2, \ldots, E_r\}$: is a set of global QoS constraints that gives the overall QoS constraints vector that must be achieved for the whole process P, such as $E_k(1 \leq k \leq r)$ is a global QoS constraint of the k^{th} QoS parameter.
- $W = \{w_1, w_2, ..., w_r\}$: is a set of application preferences that indicate the importance and the weight given to each parameter for the intended application. Such as: $w_j(1 \leq j \leq r)$ indicates the weight preference for the j^{th} QoS parameter, $w_i \in [0, 1]$ and $\sum_{i=1}^{r} w_i = 1$.

The overall QoS value of a given process depends on two principal factors: its QoS attributes and pattern structures that interconnect the tasks that compose it. Our approach considers the patterns commonly used by composition languages as BPEL (Business Process Execution Language) [20]: sequential, parallel, conditional, and loops. Each one of these structures can be mapped into a sequential model using the techniques presented in [21].

Table 4 gives the QoS aggregation function of each QoS attribute for each pattern structure mentioned above where α_i is the probability to execute the activity branch i such as

$\sum_{i=1}^{n} \alpha_i = 1$. The maximum number of iterations of each activity $A_i \in L$ is denoted by l_i.

TABLE 4. QoS aggregation function

	Sequential	Parallel	conditional	Loop
Additive	$\sum_{i=1}^{n} E_i$	$\sum_{i=1}^{n} E_i$	$\sum_{i=1}^{n} \alpha_i E_i$	$\sum_{i=1}^{n} l_i E_i$
Average	$\frac{1}{n}\sum_{i=1}^{n} E_i$	$\frac{1}{n}\sum_{i=1}^{n} E_i$	$\frac{1}{n}\sum_{i=1}^{n} \alpha_i E_i$	$\frac{1}{n}\sum_{i=1}^{n} l_i E_i$
Min-Operator	$\sum_{i=1}^{n} E_i$	$\min_{i=1}^{n} E_i$	$\sum_{i=1}^{n} \alpha_i E_i$	$\sum_{i=1}^{k} E_i$
Max-Operator	$\sum_{i=1}^{n} E_i$	$\max_{i=1}^{n} E_i$	$\sum_{i=1}^{n} \alpha_i E_i$	$\sum_{i=1}^{n} l_i E_i$
Multiplicative	$\prod_{i=1}^{n} E_i$	$\prod_{i=1}^{n} E_i$	$\prod_{i=1}^{n} \alpha_i E_i$	$\prod_{i=1}^{n} E_i^{l_i}$

***Example** 2:* Table 5 gives the QoS aggregation function for the response time (RT) and throughput (Th) parameters.

TABLE 5. QoS aggregation for the response time and throughput parameters

	Sequential	Parallel	conditional	Loop
Rt	$\sum_{i=1}^{n} Rt_i$	$\max_{i=1}^{n} Rt_i$	$\sum_{i=1}^{n} \alpha_i Rt_i$	$\sum_{i=1}^{k} l_i Rt_i$
Th	$\frac{1}{n}\sum_{i=1}^{n} Th_i$	$\frac{1}{n}\sum_{i=1}^{n} Th_i$	$\frac{1}{n}\sum_{i=1}^{n} Th_i$	$\frac{1}{n}\sum_{i=1}^{n} Th_i$

VI. DISCOVERY STEP

Once an event has occurred and received by an avatar, called **initiator**, the corresponding process is triggered. If the initiator cannot carry out the awaited process (i.e., there are one or several tasks that it cannot fulfill), it proceeds to the discovery of potential collaborators.

The discovery approach is realized via two main steps: **1) Social network building** to recover the socially closest avatars in terms of location and interests to the initiator avatar to reduce the search space and **2) Social network clustering** to classify the social avatars according to their functionalities used to direct at best the discovery requests forwarding to the most suitable cluster of avatars.

A. SOCIAL NETWORK BUILDING

In this work, the avatars publish their metadata (especially: interests, location, and owners) using semantic descriptions, in distributed regional repositories. To reduce the search space, only avatars that have at least one common interest are considered following the SPARQL query.

```
SELECT *
WHERE {
 ?resource avatarOnt:hasInterest ?Interest
 FILTER
 (?Interest IN (avatarOnt:I1 ,..., avatarOnt:Im))
}
```

Once the set of relevant avatars is recovered, social relationships of each of these avatars with the initiator avatar are calculated. We consider three social relationships which are : *Co-Work Object Relationship*, *Co-Location Objects Relationship*, and *Ownership Object Relationship*. The sharing of the same subjects of interest, the location, and the owner or group represents in our point of view the three most important social axes to consider in IoT among the relations studied in the related works. These relationships are detailed below :

- **Co-work Object Relationship (C-WOR):** evaluates social similarity distance between two avatars in terms of common interest. The C-WOR between two avatars A_i and A_j, denoted $CS(A_i, A_j)$, is computed using the cosine similarity metric [22] between the interests vectors I_i and I_j as given in equation 1 :

$$CS(A_i, A_j) = cos(I_i, I_j) = \frac{I_i . I_j}{\mid I_i \mid . \mid I_j \mid} \quad (1)$$

Where: $\mid I_i \mid$ and $\mid I_j \mid$ are the norm of I_i and I_j respectively, such as: $0 < CS(A_i, A_j) < 1$ with 0 value indicates that I_i and I_j are independent while 1 value designates that they are similar.

- **Co-Location Objects Relationship (C-LOR):** measures the geographical positions of the physical devices represented by the avatars. The C-LOR between A_i and A_j, denoted $LS(A_i, A_j)$, is calculated using as-the-crow-flies distance $D(A_i, A_j)$ as given in equation 2 :

$$LS(A_i, A_j) = \frac{1}{D(A_i, A_j) + 1} \quad (2)$$

with:

$$D(A_i, A_j) = R * acos[cos(LatA_i) * cos(LatA_j) + cos(LongA_j - LongA_i) * sin(LatA_i) * sin(LatA_j)] \quad (3)$$

where R is the Earth radius.

- **Ownership Object Relationship (OOR):** it is considered when it is preferable to use devices of the same owner or group to mainly ensure security and confidentiality. The OOR between A_i and A_j, denoted $OS(A_i, A_j)$, is calculated as given in equation 4 :

$$OS(A_i, A_j) = \begin{cases} 1 & \text{If } A_i \text{ and } A_j \text{ have the same owners} \\ 0 & \text{Else} \end{cases} \quad (4)$$

Based on co-work, co-location and ownership relationships, the social similarity distance between two avatars A_i and A_j is calculated via the weighted function $SD(A_i, A_j)$ on [0,1]:

$$SD(A_i, A_j) = \alpha * CS(A_i, A_j) + \beta * LS(A_i, A_j) + \gamma * OS(A_i, A_j) \quad (5)$$

with $\alpha, \beta, \gamma \in [0, 1]$ are the weighting coefficients which represent the application preferences for each distance, such as: α+β+γ=1. These parameters are set for each application according to its type and the preferences desired for it. After calculating the social similarity distances, a threshold, called the *social network threshold*, is defined and used to choose the avatars that will be part of the social network.

***Example** 3:* Let us again consider the overtaking scenario introduced in Section III. For instance, once the left-blinking is activated, the avatar A_1 that seeks to overtake starts by reasoning on its knowledge base to know the tasks that it

can accomplish. A_1 can perform: T_3, T_4, T_5, T_8 and T_9, but not T_1, T_2, T_6, T_7. Therefore, A_1 starts building its social network to find the avatars that can accomplish the missing tasks and even the tasks that it can perform to search if there are other IoT objects that can perform them with better QoS. For this, A_1 first tries to find the avatars that have at least one interest in common by executing on the regional server to which it is connected. Once these avatars are found, A_1 calculates the social distances that separate it from each one of them. Given the importance of the location in this scenario, the user preferences are set as follows: α=0.4, β=0.5 and γ=0.1 .

According to a social network threshold equal to 0.6 and the calculated social distances, the social network of A_1 includes: A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9 and A_{12}.

B. SOCIAL NETWORK CLUSTERING

The clustering techniques is one of unsupervised learning methods that allow partitioning the data points space into homogeneous groups called clusters. For clustering social avatars according to their functionalities, we have chosen the Fuzzy C-means algorithm. It uses a membership degree notion to indicate the belonging degree to a given cluster rather than a strict classifying. This choice is made because an avatar can provide multiple functionalities and therefore it can belong to several clusters.

***Definition** 2:* A cluster

A cluster $C_j = \{A_{1j}, A_{2j}, ..., A_{mj}\}$ is a set of avatars that have similar functionalities but differ in QoS parameters. These avatars can satisfy the T_j abstract task of process P. We denote by A_{ij} the avatar A_i belonging to the cluster C_j.

The clustering algorithm is given in Algorithm 1 where: N_a is the number of considered avatars, X is the data matrix where each line represents the functionality vector of an avatar in a binary form (the component is worth 1 if the avatar is able to perform the corresponding functionality and 0 otherwise), f_z represents the fuzzification parameter of the algorithm and *Dist* is the distance used to measure the similarity between centroids and avatars. It relies on iterative optimization of an objective function, with the updating of the membership coefficients and cluster centroids until the objective function value becomes less than ϵ.

Formally, for each avatar A_i, a coefficient giving the membership degree to the j^{th} cluster, noted u_{ij}, is defined. The sum of these coefficients must satisfy the following condition: $\sum_{j=1}^{C} u_{ij} = 1$, where C is the number of clusters to build. The c_j centroid of the j^{th} cluster represents the average of all the data points weighted by their degree of membership in this same cluster as given in the equation 6. After calculating the centroid, the membership degree of each avatar to each cluster is updated in each iteration of the algorithm as indicated in equation 7. This process is repeated until the sum of the distances between the data points and the centroids is less than ϵ.

Algorithm 1 Avatars Fuzzy C-means Algorithm

Inputs: number of clusters C, avatars data matrix $X = [x_{ij}]$

1. Initialize randomly the membership coefficients matrix U=[u_{ij}], $U^{(0)}$ Such as: $\sum_{j=1}^{C} u_{ij} = 1$;
2. Calculate the centroid C_j of each cluster j

$$Cj = \frac{\sum_{i=1}^{N_a} [u_{ij}]^{f_z} x_{ij}}{\sum_{i=1}^{N_a} [u_{ij}]^{f_z}} \tag{6}$$

3. Update the current membership matrix $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{z=1}^{C} \left(\frac{Dist(X_i, C_j)}{Dist(X_i, C_z)}\right)^{\frac{2}{f_z - 1}}} \tag{7}$$

4. If $\sum_{i=1}^{N_a} \sum_{j=1}^{C} [u_{ij}]^{f_z} Dist(X_i, C_j)^2 < \varepsilon$: Then Stop, Otherwise return to step 2.

After the clustering of the avatars according to their functionalities into several groups, an elected delegate avatar is designated for each cluster. It represents the avatar with the greatest membership degree. The role of this delegated avatar is to manage the requests sent to its cluster to discover the appropriate avatars meeting these requests. The objective is to guarantee a distributed discovery and a useful propagation of the discovery requests while minimizing as much as possible the redundancy of the messages exchanged between the avatars.

***Example** 4:* Once the social network is built, the initiator avatar A_1 divides it into several clusters according to the functionalities that the avatars provide. By applying the Algorithm 1, the following four clusters are obtained:

- C_1 includes: A_4, A_2, A_8, A_{10}, A_{16}, A_{15}, A_{13}, A_{18} and A_5
- C_2 includes: A_7, A_8, A_{10}, A_{17}, A_{16}, A_{12} and A_{15}
- C_3 includes: A_3, A_7, A_4, A_2, A_{10}, A_{13}, A_{17}, A_{16} and A_{14}
- C_4 includes: A_2, A_3, A_4, A_{12} and A_{13}

The first avatar of each cluster and that has the greatest degree of membership will be designated as the elected representative of the corresponding cluster. Thus, A_4 will be the elected of C_1, A_7 the elected of C_2, A_3 the elected of C_3 and A_2 the elected of C_4. Therefore, to discover the avatars that can satisfy T_1 and T_2 which correspond to the perception functionality, A_1 sends a request to the elected A_4 that takes care of finding the services responding to T_1 and T_2 in C_1. A_4 finds that T_1 can be performed by itself and by A_{10}, A_8, A_{16}, A_{15} and A_{18}, and T_2 by A_2, A_{10}, A_8, A_{16}, A_{15} and A_{13}. Likewise, A_1 sends a discovery request to the elected A_7 of C_2 concerning T_6 corresponding to the distance measurement functionality. A_7 finds that T_6 can be performed A_8, A_{10}, A_7, A_{16} and A_{17}. In the same way, for T_7 relating to the speed measurement functionality, the elected A_3 of C_3 finds that T_7 can be realized by A_4, A_7, A_{10}, A_{15}, A_{16} and A_{14}. Table 6 shows the built clusters for the tasks

T_1, T_2, T_6, T_7 that can not be fulfilled by the initiator avatar A_1.

TABLE 6. Candidate avatars discovered for each task

Task	Candidate avatars/cluster
T_1	$A_4, A_{10}, A_8, A_{16}, A_{15}, A_{18}$
T_2	$A_2, A_{10}, A_8, A_{16}, A_{15}, A_{13}$
T_6	$A_8, A_{10}, A_7, A_{16}, A_{17}, A_{15}$
T_7	$A_4, A_7, A_{10}, A_{15}, A_{16}, A_{14}$

VII. SELECTION STEP

Once the clusters are built, we proceed to the selection based on global QoS constraints decomposition while considering QoS fluctuation.

QoS fluctuation is defined as the distribution of the historic QoS values of a given QoS attribute and how it varies over time. Services with higher values of fluctuation are not suitable because they may diverge at run time and those with lower values are the best. The fluctuation is a very efficient factor that ensures the reliability of the concrete composition. It is calculated based on a variation coefficient (VC) which represents the ratio between the standard deviation and the average of historical QoS parameter values. Formally, this coefficient is given in Equation 8.

***Definition** 3:* QoS Fluctuation

Let q^k be the k^{th} QoS measured parameter of A_{ij} and $\{q^k_{i1}, q^k_{i2}, \ldots, q^k_{ih}\}$ the last h historical QoS values. In the literature, h is generally fixed at 6 which allows to sufficiently observe the variation of the QoS parameters.

$$VC_{ij}(q^k) = \frac{Sd(q^k)}{Av(q^k)} \tag{8}$$

such as:

- $Sd(q^k) = \sqrt{\frac{1}{h-1}\sum_{j=1}^{h}(q^k_{ij} - Av(q^k))^2}$ represents the standard deviation of q^k.
- $Av(q^k) = \frac{1}{h}\sum_{j=1}^{h} q^k_{ij}$ represents the average of q^k.

In consequence, the multi-dimensional fluctuation for all QoS attributes of A_{ij} is calculated as given in Equation 9.

$$F(A_{ij}) = \sum_{k=1}^{r} w_k \frac{max(VC_{ij}(q^k)) - VC_{ij}(q^k)}{max(VC_{ij}(q^k)) - min(VC_{ij}(q^k))} \tag{9}$$

Therefore, the fluctuation value of avatars composition AC is given in Equation 10.

$$F(AC) = \sum_{i=1}^{n} F(A_i) \tag{10}$$

The selection problem is resolved through collaboration between the initiator avatar and the elected members of the clusters. It is realized via three main steps: **1)** Quality levels calculation to identify the QoS values candidates that can be local constraints and that for each QoS attribute and each cluster, **2)** Problem solving using a generic algorithm for choosing local constraints among quality levels, and **3)** Local selection that uses the local constraints as upper/lower bounds to select the optimal avatar from each cluster in a parallel way. Figure 5 provides a global overview of the selection process.

***Definition** 4:* QoS Utility

Since an avatar has several QoS parameters, we use the Simple Additive Weighting (SAW) technique [23] to evaluate its multi-dimensional QoS utility. This technique allows mapping the avatar QoS vector to a single value while considering the application preferences. The QoS attributes are of different units and ranges, therefore, we first proceed to their normalization to obtain uniform measurements in [0,1] using Equation 11.

$$q^k_{norm} = \begin{cases} \frac{max(q^k_j) - q^k}{max(q^k_j) - min(q^k_j)} & \text{For negative parameters} \\ \frac{q^k - min(q^k_j)}{max(q^k_j) - min(q^k_j)} & \text{For positive parameters} \\ 1 & max(q^k_j) - min(q^k_j) = 0 \end{cases} \tag{11}$$

Where: q^k is the k^{th} QoS attribute to normalize, $max(q^k_j)$ is the maximum value of the k^{th} QoS parameters in the cluster C_j, and $min(q^k_j)$ represents its minimum value.

After normalizing all QoS parameters, the weighted utility function of the avatars is computed. The weighted utility of an avatar A_i of the cluster C_j is provided in Equation 12.

$$U(A_{ij}) = \sum_{k=1}^{r} w_k . q^k_{norm} \tag{12}$$

In consequence, the utility value of avatars composition AC is formulated as given in Equation 13.

$$U(AC) = \sum_{i=1}^{n} U(A_i) \tag{13}$$

A. QUALITY LEVELS DETERMINATION

Quality levels calculation step aims to determinate the candidate local constraints. For this, the range values of each QoS parameters q^k is splitted into d discrete values called *quality levels* QL_{jk} in each cluster C_j:
$QL_{jk} = \{L^1_{jk}, L^2_{jk}, \ldots, L^d_{jk}\}$ and $min(q^k_j) \leq L^1_{jk} \leq L^2_{jk} \leq \ldots \leq L^d_{jk} \leq max(q^k_j)$, such as: $min(q^k_j)$ and $max(q^k_j)$ are the maximum and minimum QoS value of q^k in the cluster C_j respectively.

Quality levels calculation is given in the Algorithm 2, (which is inspired by the algorithm proposed in [9]. For each QoS attribute q^k and for each cluster C_j, the algorithm starts by sorting the candidate avatars of C_j using QoS values (lines 6,7). Then, these avatars are divided into d subsets (line 8). After that, an avatar is chosen randomly from each subset line 10), and its value q^k is added as quality levels to the set QL_{jk} (line 11).

Selecting a quality level L^z_{jk} as a local constraint among several others depends on the assessment of two factors : *utility* and *fluctuation*, presented bellow.

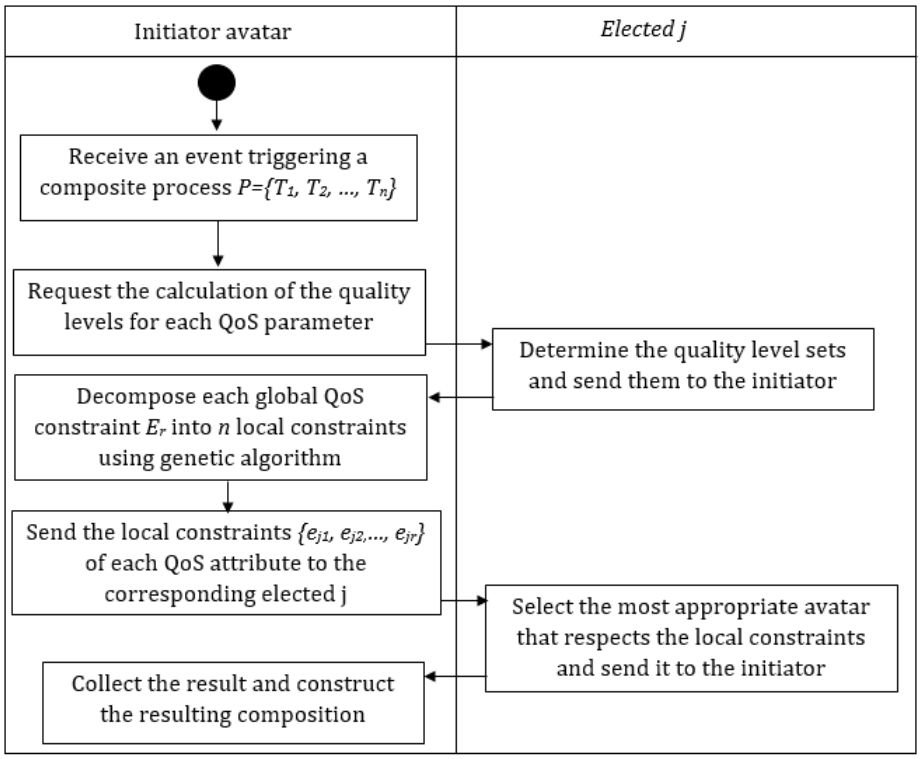

FIGURE 5. Overview of process interactions

Algorithm 2 Quality Levels Determination

Inputs: n clusters, l: the number of avatars in each cluster, d: the number of quality level such as $d \leq l$, r: the number of the global QoS constraints
Outputs: r set of quality levels
BEGIN
for Each QoS attribute q^k **do**
  for Each Cluster C_j **do**
    $QL_{jk} \leftarrow \emptyset$;
    $C_j \leftarrow Sort(C_j, q^k)$;
    Divide C_j to d subsets: $\left\{C_{jk}^1, C_{jk}^2, ..., C_{jk}^d\right\}$;
    for $z \leftarrow 1$ to d **do**
      $L_{jk}^z \leftarrow RandomSelection(C_{jk}^z)$;
      $QL_{jk} \leftarrow QL_{jk} \cup \left\{L_{jk}^z\right\}$;
    end for
  end for
end for
END

Evaluation of the quality level utility: to evaluate the utility of a quality level L_{jk}^z, a fitness function p_{jk}^z in [0,1] is used. It represents the probability with which it is interesting to use L_{jk}^z as a local constraint. It is calculated as given in Equation 14.

$$p_{jk}^z = \frac{h(L_{jk}^z)}{l} \cdot \frac{u(L_{jk}^z)}{u_{max}} \tag{14}$$

Such as:

- $h(L_{jk}^z)$ indicates the number of qualified avatars (i.e., the avatars that feet the local constraint) when L_{ik}^z is selected as a local constraint.
- l represents the total number of avatars in the cluster C_j.
- $u(L_{jk}^z)$ is the greatest utility when only the avatars qualified for L_{ik}^z are considered.
- u_{max} is the greatest utility when all the avatars of C_j are considered.

Evaluation of quality level fluctuation: The fluctuation function of a quality level L_{ik}^z is a function f_{jk}^z in [0,1], that evaluates the inconvenience if L_{ik}^z is chosen as a local constraint. It is calculated as given in Equation 15.

$$f_{jk}^z = \frac{h(L_{jk}^z)}{l} \cdot \frac{f_{min}}{f(L_{jk}^z)} \tag{15}$$

Such as:

- $h(L_{jk}^z)$ indicates the number of qualified avatars when L_{ik}^z is selected as local constraint.
- l represents the total number of avatars in the cluster C_j.
- f_{min} is the smallest fluctuation obtained while considering all the avatars of C_j.
- $f(L_{jk}^z)$ indicates the smallest fluctuation when only the avatars qualified for L_{ik}^z are considered.

Example 5: We further use the overtaking scenario to explain this step. For instance, we consider the response time q^1 for

the first cluster C_1 with $d = 3$. As shown in Figure 6, in the cluster C_1, there are 6 candidate avatars (given in table 3). To identify the quality levels for the response time q^1, we sort the avatars according to their q^1 values. After that, the sorted avatars are divided into 3 groups. Then, we select randomly an avatar from each subgroup and its value is added to the QL_{11} quality levels list. Therefore, the resulting list of quality levels is: $QL_{11} = \{51.33, 55.16, 61.83\}$.

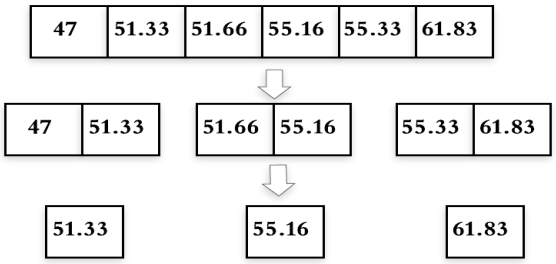

FIGURE 6. Quality levels calculation process

Likewise, we get the other quality levels as shown in Table 7:

TABLE 7. Quality levels calculation example

Cluster	Response time	Throughput
C_1	51.33, 55.16, 61.83	10.66, 14.5, 18.83
C_2	39.5, 45.33, 64.66	12, 12.83, 21.33
C_3	29.33, 58.5, 61.16	10.33, 12.66, 16.5
C_4	45.33, 46.66, 47.5	9.66, 16.16, 19
C_5	43.66, 50.16, 60	11, 14.33, 16
C_6	36.33, 50, 50.5	10, 14.83, 18.5
C_7	46.66, 49.5, 61	10, 16.16, 19
C_8	52.16, 53.16, 71.33	11.83, 12, 16.66
C_9	28, 40.66, 69.66	14.83, 18, 18.16

B. LOCAL CONSTRAINTS CALCULATION

The global QoS constraints decomposition problem seeks to find an optimal quality level combination. As each QoS attribute has a set of quality levels for each cluster, there are several possible schemes. This problem aims to simultaneously optimize the utility and fluctuation objectives under a set of global QoS constraints, i.e., a multi-objective (bi-objective) optimization problem.

Based on the identified quality levels, the utility and fluctuation functions, the QoS decomposition optimization problem is formulated as indicated in Equation 16.

$$\begin{cases} max(\sum_{j=1}^{n}\sum_{k=1}^{r} p_{jk}^{z}) & \\ min(\sum_{j=1}^{n}\sum_{q_k}^{r} f_{jk}^{z}) & \text{For dynamic attributes} \\ Agg(L_{jk}^{z}) \leq E_k & \text{For negative attributes} \\ Agg(L_{jk}^{z}) \geq E_k & \text{For positive attributes} \end{cases} \tag{16}$$

In this work, we propose a MOEA/D-IEpsilon (MOEA/D-Improved Epsilon) based decomposition approach. Indeed, Constrained Multi-Objective Evolutionary Algorithms (CMOEAs) represent an efficient solution to solve such type of problems in a large-scale researching space. Based on comparative studies [24] [25], the decomposition based CMOEAs (MOEA/D) is a promising solution for Constrained Multi-Objective Problem (CMOP). It consists in decomposing the CMOP into a set of constrained single-objective optimization sub-problems that can be solved collaboratively. The comprehensive experimental results presented in [25] indicate that MOEA/D-IEpsilon method (MOEA/D-Improved Epsilon) [26] offers the best balance between diversity and convergence.

For simplicity, we use the acronym MOEA instead of MOEA/D-IEpsilon in the rest of this paper.

The proposed method begins by generating the initial population while considering the non-feasible solutions tolerated according to an epsilon threshold. Then, it decomposes the global QoS constraints problem into a set of simpler sub-problems. The epsilon level is adjusted dynamically in each generation according to the ratio between feasible and total (RFS) solutions in the population. The solutions of these sub-problems evolve from one population to another by applying genetic operators (selection, crossover, and mutation). The obtained new solutions replace the most degraded solutions of the current population based on an epsilon value.

So, MOEA is made up of three components: **1)** Problem decomposition, **2)** Population evolution, and **3)** constraints handling. Figure 7 gives an overview of the proposed method.

1) Problem decomposition: MOEA starts by decomposing the problem into N sub-problems via Tchebycheff method using N uniformly spread weight vectors $\{\lambda^1, \ldots, \lambda^N\}$, such as $\sum_{y=1}^{2} \lambda_y^m = 1$ and $\lambda_y^m \geq 0$ for each $y \in \{1, 2\}$ with 2 is the number of sub-objectives (utility and fluctuation), and $m \in \{1, .., N\}$. The m^{th} sub-problem is defined in Equation 17.

$$\begin{cases} \min g^{te}(\mathbf{x}|\lambda^m, z^*) = \max\{\lambda_1^m|-f_u(\mathbf{x}) - z_1^*|, \\ \lambda_2^m|f_f(\mathbf{x}) - z_2^*|\} \\ \mathbf{x} \in Sol \end{cases} \tag{17}$$

Where: g^{te} is sub-problems function, Sol is the set of all solutions, f_u, f_f are utility and fluctuation objective functions, $z^* = (z_1^*, z_2^*)$ is the ideal point, i.e. the best solution obtained so far, such as: $z_1^* = min{-}f_u\{x|x \in Sol\}$ and $z_2^* = min f_f\{x|x \in Sol\}$.

A sub-problem allows to evolve a solution from the population. It is defined by its profile that contains: the best-obtained solution x_m, the sub-problem objective function defined in Equation 17, the weight vector λ^m and the neighborhood set B_m.

The sub-problem profile is initialized at the beginning of the optimization process using the Algorithm 3. This algorithm consists of randomly generating, for each sub-problem, a weight vector (line 5), and individuals from the

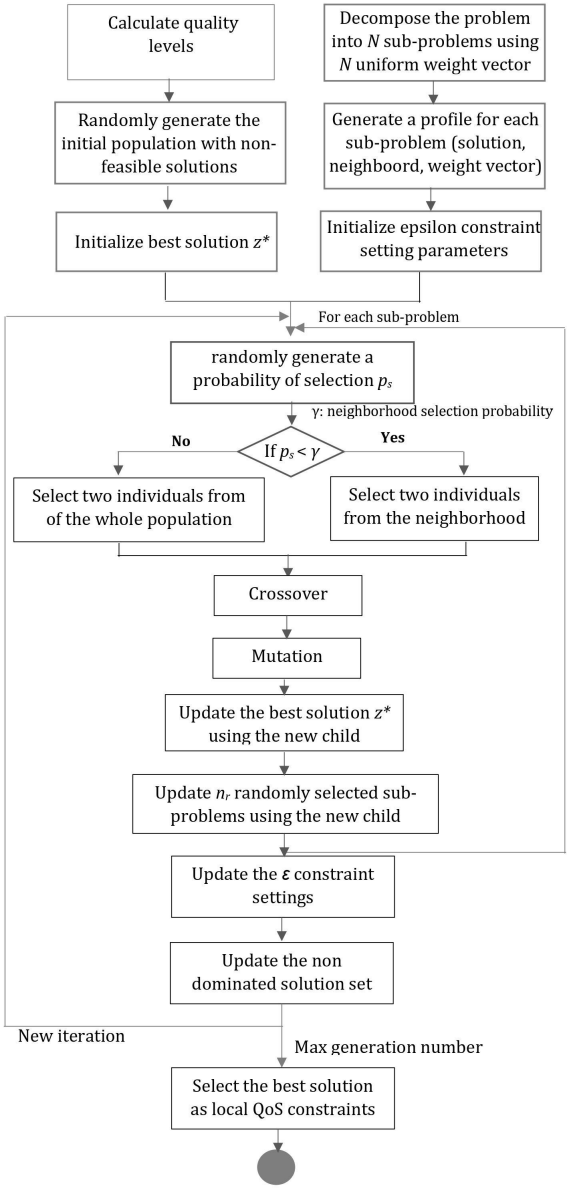

FIGURE 7. MOEA overview

initial population (line 6) to constitute its neighborhood (lines 9,10) and consequently formulate the objective function to make the solution evolve in the next generations.

2) Population evolution: the population of each sub-problem evolves from one generation to another by applying genetic operators: encoding, evaluation, selection, crossover, and mutation.

- **Encoding:** a solution $\mathbf{x}$ for the global QoS constraints decomposition problem, i.e. a chromosome instance, consists of one chosen quality level L^z_{jk} for each QoS parameter q^k of each cluster C_j. Therefore, bi-dimensional coding is used as shown in Figure 8, such

Algorithm 3 Sub-problems profile initialization

1: **Inputs:** N: the number of sub-problems, initial population, T: neighborhood size.
2: **Outputs:** Set of N sub-problems, each sub-problem m defined by a triplet (x_m: initial solution, λ^m: a weight vector, B_m: a neighborhood set)
3: **BEGIN**
4: **for** $i \leftarrow 1$ to N **do**
5: Generate randomly a weight vector $\lambda^m = (\lambda^m_1, \lambda^m_2)$ for the m^{th} sub-problem
6: Initialize x_m the initial solution of the m^{th} sub-problem chosen from the introduced population set.
7: **end for**
8: **for** $i \leftarrow 1$ to N **do**
9: Calculate the euclidean distance between the m^{th} sub-problem weight vector λ^m and the other $N-1$ sub-problems weight vectors
10: The sub-problems associated to the T closest weight vectors to λ^m are chosen to build the neighborhood set i.e; $B_m = \{m_1, \ldots, m_T\}$, where $\lambda^{m_1}, \ldots, \lambda^{m_T}$ are the T closest weight vectors to λ^m
11: **end for**
12: **END**

that: n is the number of clusters and r is the number of QoS parameters.

	C_1	C_2	C_3		C_n
$\mathbf{q}^1$	L^a_{11}	L^b_{21}	L^c_{31}	$\cdots$	L^e_{n1}
$\mathbf{q}^2$	L^f_{12}	L^g_{22}	L^h_{32}	$\cdots$	L^i_{n2}
$\mathbf{q}^3$	L^j_{13}	L^o_{23}	L^t_{33}	$\cdots$	L^u_{n3}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{q}^r$	L^v_{1r}	L^w_{2r}	L^x_{3r}	$\cdots$	L^y_{nr}

FIGURE 8. Chromosome instance

- **Evaluation Function:** the evaluation function of a given chromosome x aims to calculate the bi-objective function value of x in the population. It is defined by the vector: $(-f_u(x), f_f(x))$.
- **Selection Operation:** the tournament Selection strategy is adopted for individual selection operation in our approach. It consists to select the fittest candidates from the current generation to consider them in the next generation. For this, a partition of individuals from the population is randomly selected and running tournaments among them are performed. The winner of each tournament which is the one with the best fitness is selected to be in the next generation.
- **Crossover Operation:** due to the use of bi-dimensional encoding, the block-exchange based method is used as a crossover operator. It consists of exchanging random

size bloc between two selected parents to produce two new individuals, as illustrated in Figure 9 with a 2X3 bloc-exchange.

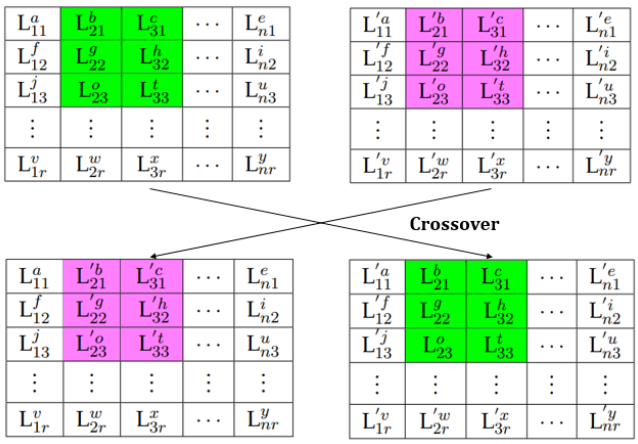

FIGURE 9. Crossover Operation

- **Mutation Operation:** regarding mutation operator, a randomly chosen quality level (genoa) in the chromosome to be muted, is replaced by another one contained in quality levels set for the same cluster and the same QoS parameter. This operation enhances population diversity. An example is given in Figure 10.

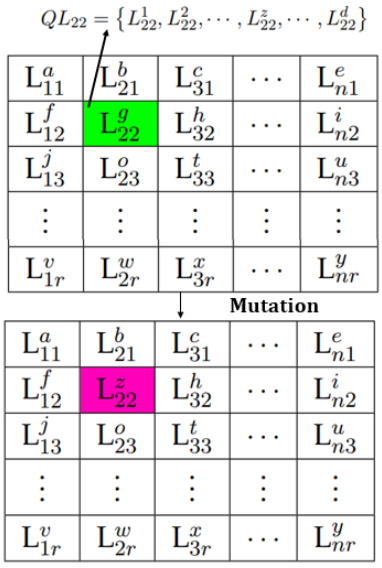

FIGURE 10. Mutation Operation

3) Constraints handling: the third component of our approach is the improved epsilon-based constraint handling mechanism that integrates the QoS global constraints. It uses a constraint violation function ϕ as described in Equation 18 so that a solution x is feasible if and only if $\phi(x) = 0$.

$$\phi(\mathbf{x}) = \sum_{i=1}^{q} |\min\left(Sup_i(\mathbf{x}) - E_k, 0\right)| + \sum_{j=1}^{p} \left|\max\left(Inf_j(\mathbf{x}) - E_k^{'}, 0\right)\right| \quad (18)$$

Where : Sup, Inf are superior and inferior constraints in the form $Sup_i(\mathbf{x}) \geq E_k$, $Inf_j(\mathbf{x}) \leq E_k^{'}$, $i \in [1, q]$, $j \in [1, p]$, p and q represent the number of superior and inferior constraints.

During the first iterations, a violation threshold is tolerated for the epsilon value which is updated according to the ratio RFS as depicted in Equation 19. This allows considering the infeasible solutions during the evolution of the population for better convergence with $\phi(x) \leq \varepsilon(g)$ at the g^{th} generation such as:

$$\varepsilon(g) = \begin{cases} \phi\left(\mathbf{x}^{\theta}\right), & \text{if } g = 0 \\ (1-\tau) * \varepsilon(g-1), & \text{if } r_g < \alpha \text{ and } g < T_c \\ (1+\tau) * \phi_{\max}, & \text{if } r_g \geq \alpha \text{ and } g < T_c \\ 0, & \text{if } g \geq T_c \end{cases} \quad (19)$$

Where :

- $\phi_g\left(\mathbf{x}^{\theta}\right)$ is the overall constraint violation of the top θ individuals in the initial population,
- r_g is the ratio RFS in the g^{th} generation,
- $\tau \in [0, 1]$ is used to control the speed of reducing the relaxation of constraints. Also, it is used to control the scale factor multiplied by the maximum overall constraint violation,
- $\alpha \in [0, 1]$ allows to control the searching preference between the feasible and infeasible regions,
- $\phi_{\max}$ is the maximum overall constraint violation found so far,
- $\varepsilon(g)$ is updated until the generation counter g reaches the control generation T_c when $\varepsilon(g) = 0$

The best-obtained solution x_m is updated whenever a new individual is obtained for the sub-problem.

The Algorithm 4 illustrates the updating process that considers both the objective function and the constraint violation function given in Equation 18. The current solution x_m is replaced by a new individual y_m in three cases: a) both two individuals violation constraint respect the epsilon level value (line 3) and the objective function value of y_m is better than that of x_m (line 4), b) both have the same violation constraint value (line 7) and y_m is more optimal then x_m (line 8), and c) y_m violation constraint is smaller then constraint violation of x_m (line 10).

The global MOEA algorithm for global QoS constraints decomposition is given in Algorithm 5. It consists of three main steps: in the first step, the initial population is generated (line 11), the main problem is decomposed using Equation 17 (line 12), and algorithm parameters are initialized (lines 13 to 15). The second step consists in population evolution using an evolutionary operator (selection, crossover, mutation)

Algorithm 4 Sub-problems profile update

Inputs: x_m: current solution for the m^{th} sub-problem, y_m: the newly obtained individual, $\varepsilon(g)$: the current epsilon level value.
BEGIN
if $(\phi(\mathbf{y}_m) \leq \varepsilon(g)$ and $\phi(\mathbf{x}_m) \leq \varepsilon(g))$ **then**
  if $(g(\mathbf{y}_m|\lambda^m, z^*) \leq g(\mathbf{x}_m|\lambda^m, z^*))$ **then**
    $\mathbf{x}_m$ is replaced by $\mathbf{y}_m$
  end if
else if $\phi(\mathbf{y}_m) == \phi(\mathbf{x}_m)$ **then**
  if $g(\mathbf{y}_m|\lambda^m, z^*) \leq g(\mathbf{x}_m|\lambda^m, z^*)$ **then**
    $\mathbf{x}_m$ is replaced by $\mathbf{y}_m$
  end if
else if $\phi(\mathbf{y}_m) < \phi(\mathbf{x}_m)$ **then**
  $\mathbf{x}_m$ is replaced by $\mathbf{y}_m$
end if
END

(lines 16 to 23). The last step considers algorithm parameters updates (lines 25 to 36): the epsilon level as well as sub-problems profile updates. Each sub-problem has a profile that is updated during the execution. It contains its λ vector, its current solution, and its neighborhood.

***Example** 6:* For better understanding, let us consider the overtaking use case. To select the local QoS constraint scheme among the calculated quality levels, we first transform the process to satisfy into a sequential process. To do so, we rely on the work presented in [21] and we create compound tasks to assemble the complex structures. For example, T_{678} = ((T_7 sequential T_8) parallel T_6). The transformation steps are illustrated in Figure 11.

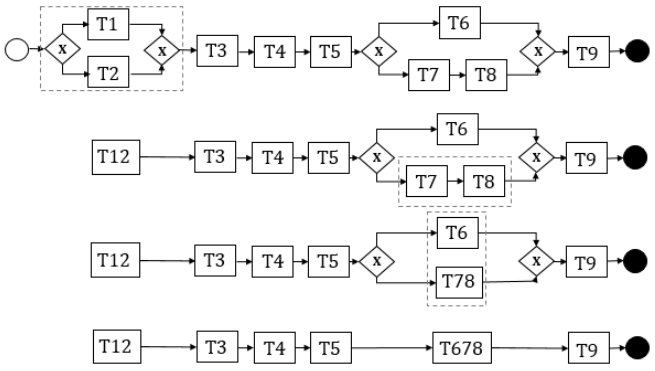

FIGURE 11. Transformation into a sequential process

After calculating the quality levels for the different elementary tasks composing the process (Table 7), the quality levels of the composite tasks are obtained by using the aggregation (cf. Table 4) of the quality levels of their elementary tasks, and that, according to the operators that connect them and the type of the QoS parameter. For example, for the composite task T_{12}=(T_1 [parallel] T_2), the quality levels

Algorithm 5 MOEA/D-IEpsilon: Global constraints decomposition

Inputs:
N: the number of sub-problems
$T_{\max}$: the maximum number of generations
N weight vectors: $\lambda^1, \ldots, \lambda^N$
γ: neighborhood selection probability
n_r : the maximum number of sub-problems updated during an iteration
Parameters τ, α, r_g, T_c and θ for epsilon Level Setting.
Outputs:
n local QoS constraints
BEGIN
1. Generate an initial population $Pop = \{x_1, \ldots, x_N\}$
2. Decompose the problem into N sub-problems according to the Algorithm 3 and Equation 17
3. Initialize $\varepsilon(0), r_g$ and $\phi_{\max}$
4. Initialize the ideal point $z^* = (z_1^*, z_2^*)$
5. Set $iter = 0$ and $g = 0$.
while $iter \leq T_{\max}$ **do**
  for $m \leftarrow 1$ to N **do**
    Generate randomly the selection probability p_s in [0,1]
    if $p_s \leq \gamma$ **then**
      Select two parents $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ from the neighborhood B_m with the tournament selection method
    else
      Select two parents $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ from the whole population with the tournament selection method
    end if
    Apply the crossover operator on $\mathbf{x}_{p1}$ and $\mathbf{x}_{p2}$ and produce a new solution $\mathbf{y}_{cr}$ with probability equal to p_c
    Apply the mutation operator to $\mathbf{y}^{cr}$ with probability equal to p_{mu} and produce a new solution $\mathbf{y}_{mu}$
    $iter = iter + 1$
    Update $\phi_{\max}$
    Update the ideal point z^*:
    if $z_1^* < f_u(y_m)$ **then**
      $z_1^* = f_u(y_m)$
    end if
    if $z_2^* > f_f(y_m)$ **then**
      $z_2^* = f_f(y_m)$
    end if
    Update n_r randomly selected sub-problems using the new child with Algorithm 4
  end for
  $g = g + 1$
  Update r_g
  Update $\varepsilon(g)$ with Equation 19
  NS = Non dominated solutions in (NS $\cup Pop$)
end while
Select the best solution from NS as local QoS constraints
END

for the response time are obtained using the $maximum$ aggregation between the quality levels of T_1 and T_2, i.e., for the first quality level $L^1_{12,1} = max(\{L^1_{1,1}, L^1_{2,1}\} = max(\{51.33, 39.5\} = 51.33$. Regarding the quality levels for the throughput, the aggregation $minimum$ is used. For example, the first quality level $L^1_{12,2} = min(\{L^1_{1,2}, L^1_{2,2}\} = min(\{10.66, 12\} = 10.66$ and the same for the other quality levels.

The evaluation of the quality levels of the composite tasks relies on the use of two functions : *the utility function* that must be maximized, and the *the fluctuation function* that must be minimized. So, we fix the aggregate utility of each quality level of a composite task at the minimum utility value of the tasks that compose it and its fluctuation at the maximum fluctuation value. We can generalize this calculation as follows :

Let T_c be a composite task and $\{T_1, T_2, \ldots, T_y\}$ the set of its elementary tasks. The utility and the fluctuation of z^{th} quality level of k^{rd} QoS attribute, denoted $p^z_k(T_c)$ and $f^z_k(T_c)$ respectively, is given according to Equation 20.

$$\begin{cases} p^z_k(T_c) = min(p^z_{jk}), 1 \leq j \leq y \\ f^z_k(T_c) = max(f^z_{jk}), 1 \leq j \leq y \end{cases} \quad (20)$$

After the process transformation, several decompositions will be operated to determine the local constraints for each atomic task, and this by determining the local constraints of the compound tasks which will serve as global constraints for the simple tasks which compose them.

For example, for the first decomposition, we apply the MOEA algorithm on the process $P_1 = \{T_{12}, T_3, T_4, T_5, T_{786}, T_9\}$ under the global constraints to get the local constraints for each task in this process :

Task	L^{opt}_{j1}	L^{opt}_{j2}
T_{12}	64.66	12.83
T_3	58.5	16.5
T_4	45.33	16.16
T_5	43.66	16
T_{786}	102.66	12
T_9	40.66	18

Next, for the second decomposition, we apply MOEA on $P_2 = \{T_1, T_2\}$ under the constraints: 64.66 ms for the response time and 12.83 r/s for the throughput to get the local constraints of P_2:

Task	L^{opt}_{j1}	L^{opt}_{j2}
T_1	61.83	18.83
T_2	45.33	12.83

After that, the result of applying MOEA on $P_3 = \{T_6, T_{78}\}$ under the constraints: 102.66 ms and 12 r/s is given :

Task	L^{opt}_{j1}	L^{opt}_{j2}
T_6	50	14.83
T_{78}	102.66	12

The last step consists in applying MOEA on the process $P_4 = \{T_7, T_8\}$ under the constraints: 102.66 ms and at 12 r/s to obtain the local constraints :

Task	L^{opt}_{j1}	L^{opt}_{j2}
T_7	46.66	16.16
T_8	53.16	16.66

C. LOCAL SELECTION

Once the local QoS constraints for each QoS attribute q^k and each cluster C_j are obtained, the initiator avatar forwards them to the corresponding elected avatars. These constraints serve as bounds for selecting from each cluster the avatar with the highest utility and lowest fluctuation. This selection is done in parallel and distributed way in each cluster. Formally, let $E = \{e_{j1}, e_{j2}, \ldots, e_{jr}\}$ be the set of local QoS constraints, the selected avatar A_{ij} of each cluster C_j must satisfy:

$$\begin{cases} \max(U(A_{ij}) - F(A_{ij})) & \\ q^k_{ij} \leq e_{jk} & \text{For negative attributes} \\ q^k_{ij} \geq e_{jk} & \text{For positive attributes} \end{cases} \quad (21)$$

Algorithm 6 gives the local selection method carried out by the elected avatars in each cluster C_j in parallel. Its input is the set of local QoS constraints E. It first computes the utility (line 5) and the fluctuation (line 6) for each avatar of the cluster. Then, it sorts the avatars (line 9) with highest utility and lowest fluctuation (line 7). The avatar with highest utility and lowest fluctuation that respects the local QoS constraints (loops 10, 11) is selected (line 19).

Algorithm 6 Local selection method

1: **Inputs:** E **a set of** r **local QoS constraints**
2: **Outputs: Selected avatars**
3: **BEGIN**
4: **for Each avatar** A_i **in cluster** C_j **do**
5: $U(A_{ij}) \leftarrow ComputeUtility(A_{ij})$
6: $F(A_{ij}) \leftarrow ComputeFluctuation(A_{ij})$
7: $Calculate(U(A_{ij}) - F(A_{ij}))$
8: **end for**
9: $SortAvatars(A_{ij})$
10: **for Each** A_i **in** C_j **do**
11: **for Each** q^k **do**
12: **if** q^k **is positive AND not(**$q^k_{ij} \geq e_{jk}$**) then**
13: **Break**
14: **end if**
15: **if** q^k **is negative AND not(**$q^k_{ij} \leq e_{jk}$**) then**
16: **Break**
17: **end if**
18: **end for**
19: $Select(A_{ij})$
20: **Break**
21: **end for**
22: **END**

After selecting the fittest avatars, each elected sends its choice to the initiator to construct the execution plan of the resultant composition according to the abstract process.

VIII. EXPERIMENTAL EVALUATION

The proposed approach has been implemented and evaluated. In this section, we first study its complexity. Then, we present and analyze the experimental results that focus on the computation time and the optimality.

A. COMPLEXITY EVALUATION

The complexity of the selection approach based on the decomposition of global QoS into local ones depends on four parameters: 1) the number of clusters n, 2) the number of avatars per cluster l, 3) the number of global QoS constraints r, and 4) the number of quality levels d.

We study the complexity of the different steps which are: the quality levels calculation, the MOEA algorithm for global QoS constraints decomposition, and the local selection.

The complexity of the quality levels calculation and the local selection steps are linear. It depends only on the number of avatars per cluster l. Indeed, each elected of each cluster (n clusters) performs these steps locally and in parallel. Therefore, the complexity of these two steps is $O(2l)$.

The complexity of the evolutionary MOEA algorithm is also linear and depends on the number of generations g, the population size P_{size} and the variables number of the problem $n.r.d$, therefore, the complexity of this step is $O(n.r.d.g.P_{size})$.

B. EXPERIMENTAL SETUP

The proposed approach has been implemented following a microservice architecture. The avatars are implemented using spring boot microservices[5] and the messages exchanged between them are carried out via Apache HTTP components [6] and XML format. To ensure asynchronous communication, java threads are used.

The tests execution are realized in a multiprocessor Ubuntu Server environment with 64GB RAM.

1) DataSet

For the experiments, we have used the IoT dataset provided in [27] which adapts the Web service dataset WS-DREAM proposed in [28]. It provides a multi-matrix of the response time and throughput of 339 users for 5 825 IoT services.

This dataset provides a large number of observations (1 974 675 observations). Besides, this dataset provides time-varying QoS attributes caused by network congestion which allows taking into account the QoS fluctuation. To adapt the use of this dataset to our approach, we randomly divide the dataset into n clusters where n is the number of abstract tasks of the considered process.

[5]https://spring.io/projects/spring-boot
[6]https://hc.apache.org/httpcomponents-asyncclient-4.1.x/index.html

2) Empirical study

We start our study by choosing the adequate parameters of the MOEA algorithm based on the convergence and diversity of the obtained final population. For this, two indices are used: Inverted Generational Distance (IGD) index to measure the population convergence and the Hyper-volume (HV) index to measure its diversity. Note that a smaller value of the IGD index and a larger value of HV index indicates better convergence and diversity respectively. These two indices are combined in a global one named Quality index (QI) defined as follows: $QI = HV + \frac{1}{IGD}$, such as a larger value of QI denotes the better quality of MOEA.

After fixing the MOEA parameters, a comparative study between the proposed approach and the exact method developed using the vOptSolver[7] is carried out. This study evaluates and compares response time and optimality criteria.

Population evolution parameters

For the following graphs and as we treat service selection and composition problem in the IoT context, we consider big size problem instance with 10 clusters, 1000 avatars per cluster, and 20 quality levels, so the search space size is $d^{(r*n)} = 20^{2*10}$.

The graph in Figure 12 represents the variation of quality index value as a function of population size and neighborhood size. The results show that 300 and 10 seem to be the

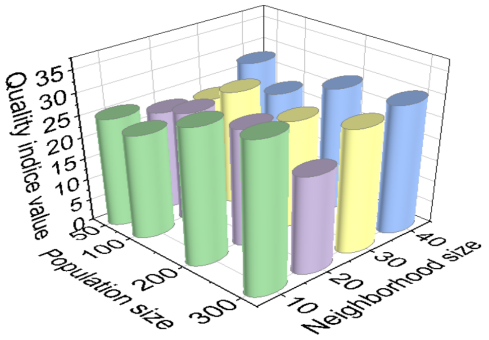

FIGURE 12. Variation of quality index value as a function of population size and neighborhood size

best values to select for population size and neighborhood size respectively as the index value is maximized for these values. This is justified by the fact that a sufficient size of the population makes it possible to reach a considerable number of other solutions by crossing and mutation, and a small neighborhood size allows to have sufficient coverage of the search space via the different sub-problems that seek to improve their best solution.

Regarding the graph in Figure 13, it represents the variation of quality index value as a function of the number of generations and the number of replaced solutions.

The results indicate that the adequate values for the number of generations and the replaced solutions are respectively

[7]https://github.com/vOptSolver

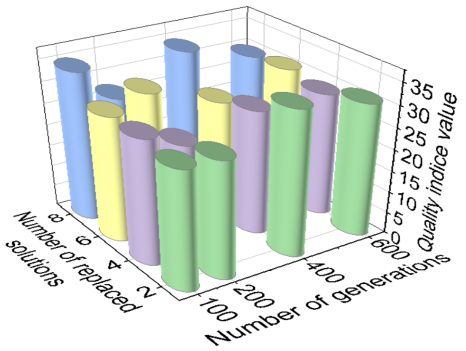

FIGURE 13. Variation of quality index value as a function of number of generations and number of replaced solutions

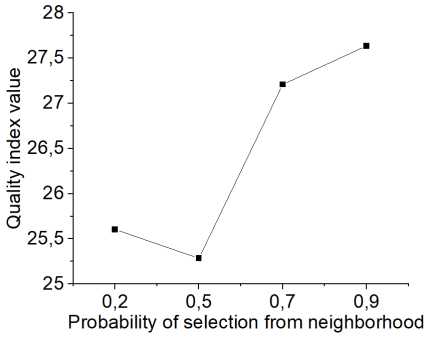

FIGURE 16. Variation of quality index value as a function of selection probability

400 generations and 8 individuals. These values guarantee a better convergence and diversity of MOEA. This is because we consider a large instance of problem, so it took a large number of generations and a large number of subproblems to update at each iteration to ensure convergence and diversity respectively.

The graphs in Figure 14, Figure 15 and Figure 16 illustrate the variation of quality index as a function of the evolution operators probabilities: crossover probability, mutation probability and selection probability respectively.

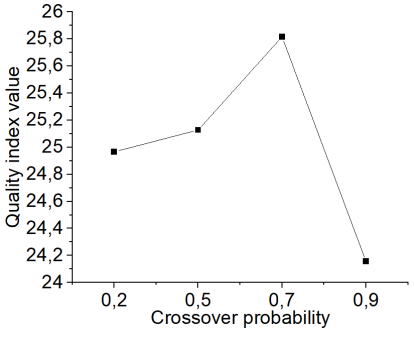

FIGURE 14. Variation of quality index value as a function of crossover probability

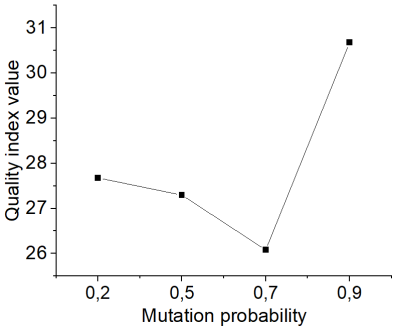

FIGURE 15. Variation of quality index value as a function of mutation probability

The results show that the adequate values for crossover and mutation probability are 0.7 and 0.9 respectively. This is justified by the fact that the high crossover and mutation probabilities have a positive impact on the convergence and diversity of the current population. Regarding the probability of selection from the neighborhood, the value 0.9 is the best value to choose because the sub-problem neighborhood allows information exchange between sub-problems for better convergence.

For the graphs in Figure 17, Figure 18 and Figure 19, they represent the variation of quality index value as a function of the epsilon level setting parameters. The empirical study shows that setting the searching preference between the feasible and infeasible regions α to 0.8, τ to 0.1, and the generation control to 180 iterations gives the best diversity and convergence. This means that to get better convergence and diversity for big-size problem instance, we should consider $(1 - \alpha)$ equal to 0.2 as the probability of searching in the infeasible solutions region with the speed of constraints relaxation τ equal to 0.1 and this for the 180 first iterations.

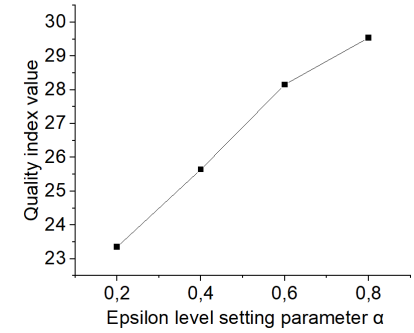

FIGURE 17. Variation of quality index value as a function of epsilon level setting α

Comparative study

In this section, a comparative study is conducted to compare our local selection approach based on the decomposition of global QoS constraints into local constraints using MOEA

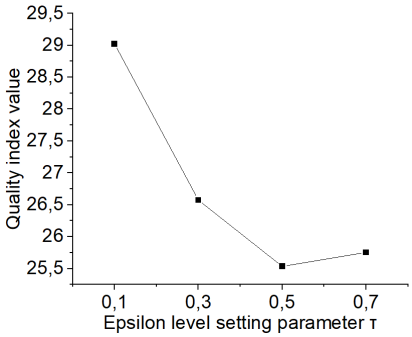

FIGURE 18. Variation of quality index value as a function of epsilon level setting τ

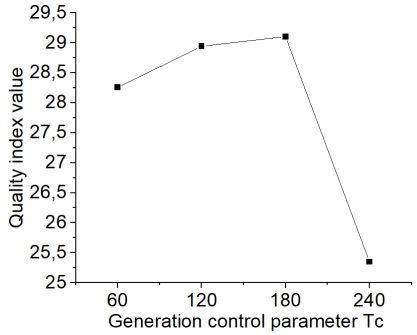

FIGURE 19. Variation of quality index value as a function of the generation control parameter

with two other approaches in terms of computation time and optimality. For the first approach, it relies on the vOptSolver to solve the problem of decomposing global QoS constraints. The second approach is the exhaustive global selection method.

MOEA Vs vOptSolver

The methods MOEA and vOptSolver are both based on local selection based on the decomposition of global QoS constraints but they differ in the way of solving the optimization problem to obtain local constraints. vOptSolver is an exact resolution method that makes it possible to choose the optimal local constraints scheme from the quality levels, while MOEA is an approximate meta-heuristic method.

The graph in Figure 20 shows the evaluation of the optimality of the global QoS constraints decomposition stage of the approach based on MOAE compared to the exact method. This optimality is defined by: $DO = \frac{U_{MOEA}}{U_{Exact}} * 100$ such as U_{MOEA} and U_{Exact} are the utilities of local constraints schemes obtained by the MOEA based approach and the exact decomposition-based approach respectively. Note that the exact decomposition-based approach offers the optimal decomposition (its optimality is equal to 100%).

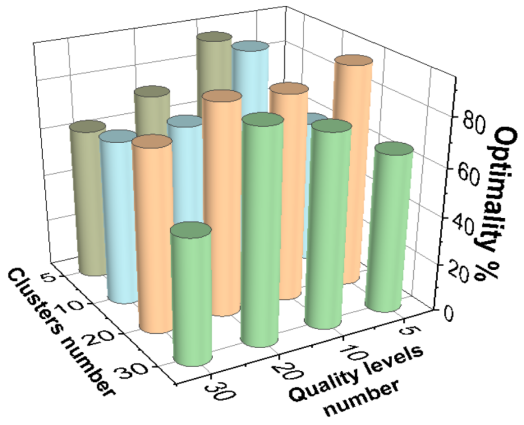

FIGURE 20. Variation of the MOEA optimality as a function of both number of clusters and quality levels number

The results show that the MOEA global constraints decomposition-based approach offers good optimality for the global constraint decomposition stage with an optimality between [80%,97%], for a number of quality levels equal to approximately 10% of the number of avatars per cluster.

The graphs in Figure 21 and Figure 22 represent a comparative evaluation between our approach and the exact method based approach according to response time as a function of clusters number and quality levels number.

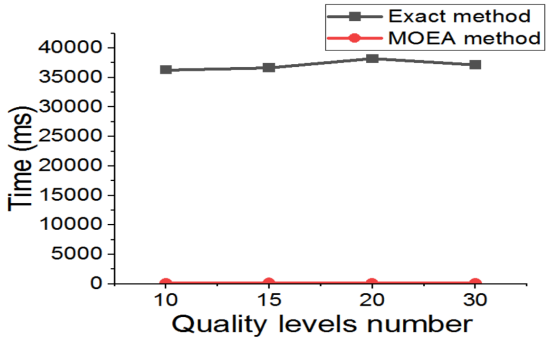

FIGURE 21. Variation of computation time as a function of number of clusters for both MOEA and exact based approaches

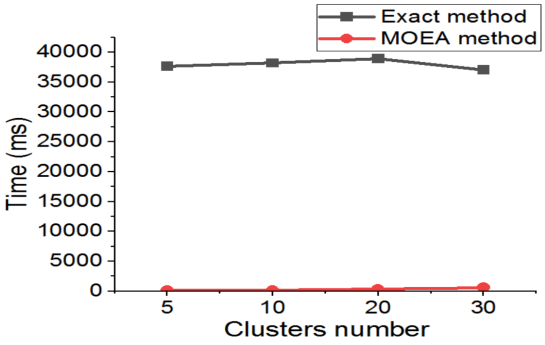

FIGURE 22. Variation of computation time as a function of number of quality levels for both MOEA and exact based approaches

The results show that the evolutionary genetic algorithm (MOEA) is about 116 times faster than the exact method and this with high-quality results. This is due to the fact that the exact method based on vOptSolver explores the whole search space ie tests all possible local constraint schemes from quality levels and therefore it consumes a significant computation time, while MOEA allows a reduced but smarter search.

MOEA Vs Global selection

This part is dedicated to the evaluation of the whole proposed selection approach between the two decomposition-based approaches (MOEA and vOptSolver) and the exhaustive global selection method in terms of optimality and computation time.

The graphs in Figure 23 and Figure 24 represent the evaluation of the MOEA and exact global constraints decomposition-based approaches in term of optimality compared to the global selection method (whose optimality is equal to 100%). The optimality of selection stage is defined as: $DS = \frac{U_{MOEA}}{U_{global}} * 100$ ($DS = \frac{U_{exact}}{U_{global}} * 100$ for exact method), such as: U_{global}, U_{MOEA} and U_{exact} are the utilities of the solution (i.e., of the selected composition of avatars) obtained by the global selection method, the MOEA and the exact global constraints decomposition-based approaches respectively.

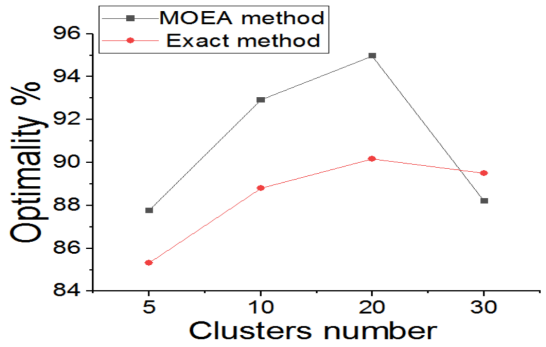

FIGURE 23. Variation of selection optimality as a function of number of clusters

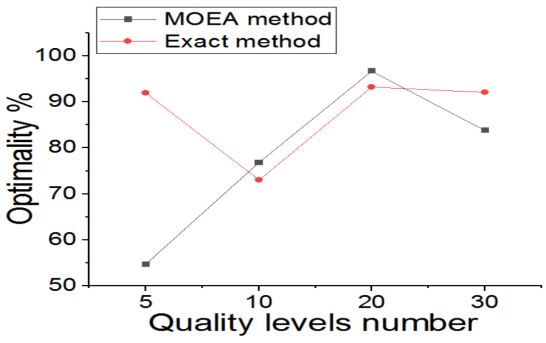

FIGURE 24. Variation of selection optimality as a function of number of quality levels

The results show first that our approach offers better results in terms of optimality. This is because the exact based approach selects very restrictive local constraints during the global constraints decomposition process, so many clusters are found with very few avatar proposals. Therefore, it impacts negatively the utility of the solution. The graphs show also that local selection methods based on the decomposition of global QoS constraints, either MOEA or exact, provide good optimality and therefore allow the selection of a near-optimal solution.

The performance evaluation in term of computation time between the three approaches: our approach MOEA and the exact method for local selection based on the decomposition of global QoS constraints, and the global selection approach is depicted in the two graphs given in Figure 25 and Figure 26. This study is carried out as a function of the number of clusters and the number of avatars per cluster. The evaluation according to the number of quality levels is not considered because the global selection is independent of this parameter.

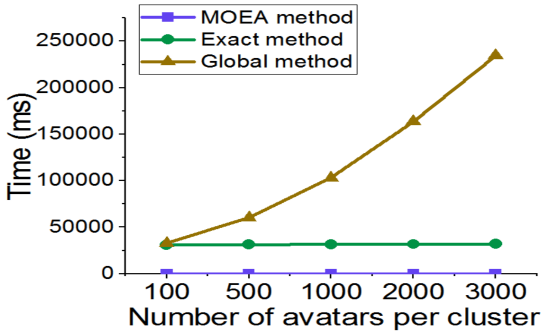

FIGURE 25. Variation of response time as a function of number of avatars per cluster

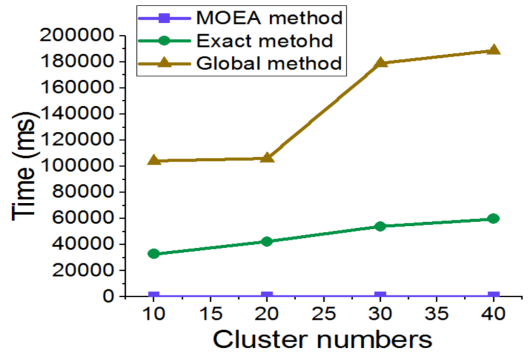

FIGURE 26. Variation of response time as a function of number of clusters

The results demonstrate that our local selection approach based on decomposing global QoS constraints using MOEA is extremely efficient in terms of computing time compared to the other two approaches. This is especially important in large systems.

MOEA Vs genetic based approach

We also conducted comparative experiments of the proposed MOEA approach with classical genetic algorithm based approach. Unlike the MOEA approach, the classical genetic algorithm do not use epsilon to manage constraints while searching feasible solutions and ignores regions with non-feasible solutions.

Graphs depicted in Figure 27 and Figure 28 illustrate the variation of quality index value and response time respectively according to the population size.

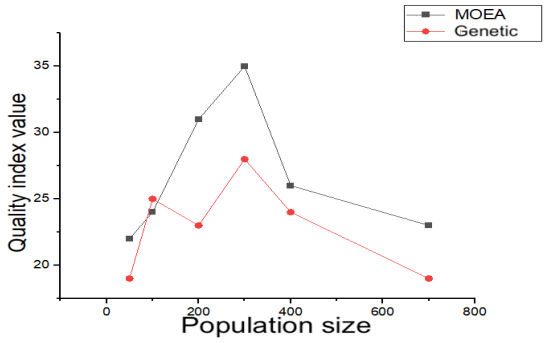

FIGURE 27. Variation of quality index value as a function of population size

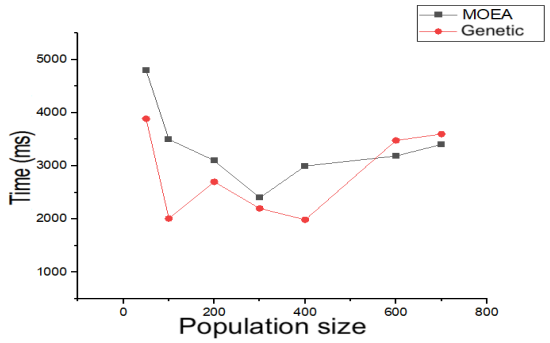

FIGURE 28. Variation of response time as a function of population size

The results show that the MOEA based approach ensures a best diversity and convergence than the classical genetic algorithm. Indeed, unlike the classical genetic based approach, the MOEA approach explores also regions with unfeasible solutions. This has obviously an impact on the response time of the MOEA approach, which remains close to the classical genetic algorithm, especially for big-size problems for which there is a light improvement.

IX. CONCLUSION AND FUTURE WORK

Through this paper, we have presented an efficient variant of the genetic algorithm for solving the fluctuating QoS-based IoT service selection problem that relies on avatars for virtualizing IoT devices. It is based on the decomposition of global constraints into local constraints so that the elected avatars of the distributed clusters locally select the fittest avatar to achieve a near-to-optimal solution. Unlike existing works, our method deals with QoS parameter fluctuation to ensure the reliability of the resulting composition. This is particularly meaningful for IoT applications with a dynamic environment and QoS requirements. This work extends our previous work which proposes two pruning mechanisms: social network building and clustering allowing to reduce the search space and therefore the problem size. The experimental results show a significant improvement in terms of computation time and optimality.

The proposed work aims to discover and select IoT services while considering flucating QoS parameters. We are extending the proposed approach by QoS-aware on the fly self-configuration mechanisms to face changes and fluctuations that can violate the awaited QoS parameters. In this context, we will evaluate the impact of the proposed selection approach during the execution against classical selection approaches that do not consider the fluctuation aspect.

Moreover, the proposed approach is carried out actively on existing avatars when an IoT application must be fulfilled. We plan to extend the approach to enable predictive discovery, selection, and self-configuration of IoT avatars while considering the mobility dimension and its impacts on spatio-temporal properties.

ACKNOWLEDGMENT

This work is funded by Continental Digital Service France (CDSF) in the framework of the eHorizon project.

REFERENCES

[1] K. Khadir, N. Guermouche, and T. Monteil, "Autonomous avatar-based architecture for value-added services provision," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 2019, pp. 291–298.

[2] K. Khadir, N. Guermouche, T. Monteil, and A. Guittoum, "Towards avatar-based discovery for iot services using social networking and clustering mechanisms," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–7.

[3] Y. Liu, R. Yang, and S. Zhang, "Service selection method based on skyline in cloud environment," *International Journal of Performability Engineering*, vol. 13, no. 7, pp. 1039–1047, 2017.

[4] M. Rajeswari, G. Sambasivam, N. Balaji, M. S. Basha, T. Vengattaraman, and P. Dhavachelvan, "Appraisal and analysis on various web service composition approaches based on qos factors," *Journal of King Saud University-Computer and Information Sciences*, vol. 26, no. 1, pp. 143–152, 2014.

[5] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, pp. 6–es, 2007.

[6] V. R. Chifu, C. B. Pop, I. Salomie, and E. S. Chifu, "Hybrid honey bees mating optimization algorithm for identifying the near-optimal solution in web service composition," *Computing and Informatics*, vol. 36, no. 5, pp. 1143–1172, 2017.

[7] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication," *International Journal of Computational Intelligence Systems*, vol. 3, no. sup01, pp. 18–30, 2010. [Online]. Available: https://doi.org/10.1080/18756891.2010.9727750

[8] M. B. Karimi, A. Isazadeh, and A. M. Rahmani, "Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm," *The Journal of Supercomputing*, vol. 73, no. 4, pp. 1387–1415, 2017.

[9] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, pp. 1–31, 2012.

[10] Z. Yanwei, N. Hong, D. Haojiang, and L. Lei, "A dynamic web services selection based on decomposition of global qos constraints," in *2010 IEEE Youth Conference on Information, Computing and Telecommunications*. IEEE, 2010, pp. 77–80.

[11] Y. Yuan, W. Zhang, X. Zhang, and H. Zhai, "Dynamic service selection based on adaptive global qos constraints decomposition," *Symmetry*, vol. 11, no. 3, p. 403, 2019.

[12] Z. Z. Liu, Z. P. Jia, X. Xue, and J. Y. An, "Reliable web service composition based on qos dynamic prediction," *Soft Computing*, vol. 19, no. 5, pp. 1409–1425, 2015.

[13] C. Mo, Y. Li, and L. Zheng, "Simulation and analysis on overtaking safety assistance system based on vehicle-to-vehicle communication," *Automotive Innovation*, vol. 1, no. 2, pp. 158–166, 2018.

[14] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. R. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," *IEEE communications magazine*, vol. 51, no. 6, pp. 102–111, 2013.

[15] D. Kelaidonis, A. Somov, G. Poulios, V. Foteinos, V. Stavroulaki, P. Vlacheas, and P. Demestichas, "A cognitive management framework for smart objects and applications in the internet of things," in *International Conference on Mobile Networks and Management*. Springer, 2012, pp. 196–206.

[16] S. Malik, S. Ahmad, and D. Kim, "A novel approach of iot services orchestration based on multiple sensor and actuator platforms using virtual objects in online iot app-store," *Sustainability*, vol. 11, no. 20, p. 5859, 2019.

[17] D. Roman, J. Kopeckỳ, T. Vitvar, J. Domingue, and D. Fensel, "Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis," *Journal of Web Semantics*, vol. 31, pp. 39–58, 2015.

[18] S. Chhun, N. Moalla, and Y. Ouzrout, "Qos ontology for service selection and reuse," *Journal of Intelligent Manufacturing*, vol. 27, no. 1, pp. 187–199, 2016.

[19] V. Tietz, G. Blichmann, S. Pietschmann, and K. Meißner, "Task-based recommendation of mashup components," in *International Conference on Web Engineering*. Springer, 2011, pp. 25–36.

[20] M. B. Juric, B. Mathew, and P. G. Sarang, *Business process execution language for web services: an architect and developer's guide to orchestrating web services using BPEL4WS*. Packt Publishing Ltd, 2006.

[21] S.-G. Wang, Q.-B. Sun, and F.-C. Yang, "Web service dynamic selection by the decomposition of global qos constraints," *Ruanjian Xuebao/Journal of Software*, vol. 22, no. 7, pp. 1426–1439, 2011.

[22] A. Mei, G. Morabito, P. Santi, and J. Stefa, "Social-aware stateless forwarding in pocket switched networks," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 251–255.

[23] A. Afshari, M. Mojahed, and R. M. Yusuff, "Simple additive weighting approach to personnel selection problem," *International Journal of Innovation, Management and Technology*, vol. 1, no. 5, p. 511, 2010.

[24] G. Chiandussi, M. Codegone, S. Ferrero, and F. Varesio, "Comparison of multi-objective optimization methodologies for engineering applications," *Computers Mathematics with Applications*, vol. 63, no. 5, pp. 912 – 942, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0898122111010406

[25] Z. Fan, Yi Fang, W. Li, Jiewei Lu, X. Cai, and Caimin Wei, "A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 209–216.

[26] Z. Fan, H. Li, Caimin Wei, W. Li, Han Huang, X. Cai, and Z. Cai, "An improved epsilon constraint handling method embedded in moea/d for constrained multi-objective optimization problems," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec 2016, pp. 1–8.

[27] G. White, A. Palade, C. Cabrera, and S. Clarke, "Iotpredict: Collaborative qos prediction in iot," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2018, pp. 1–10.

[28] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating qos of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.

KARIMA KHADIR is a research and development consultant in the field of connected mobility in the Expleo Group. She holds a PhD in 2021 from University of Toulouse, LAAS-CNRS and IT engineering degree from ESI (Computer Science High School) in Algeria in 2017. She is interested in several IoT related problems such as heterogeneity of IoT devices, dynamic discovery, and selection. She has been involved in several research and development projects such as in the OM2M eclipse open source project.

NAWAL GUERMOUCHE is associate professor at INSA (National Institute of Applied Sciences) since 2011. She is member of the laboratory LAAS-CNRS (National Center for Scientific Research). She holds a PhD in computer science since 2010 from Lorraine University, LORIA-INRIA at Nancy, France. Her research focuses on Service-oriented architecture and computing, business process management, Cloud/Fog based architectures, Internet of Things, and Cyber-physical systems. She is involved in several national and international research and development projects for a wide range of applications including service-oriented systems and IoT. She published several research papers in well-known international conferences and journals. She is member of program committees of several international conferences and journals.

AMAL GUITTOUM received her IT engineering degree in computer science from the High national school of computer science of Algiers, Algeria, in 2020. She was a research intern at the SARA team, in the laboratory of LAAS-CNRS, France, 2020. She is currently a Ph.D student at Orange Lab, Grenoble, France. She works on IoT device management.

THIERRY MONTEIL is professor in computer science at INSA – University of Toulouse and researcher at IRIT (Institut de Recherche en Informatique de Toulouse). He works on parallel computing, Cloud resources management, autonomic middleware, machine-to-machine and Internet of Things architecture. He had represented CNRS at ETSI (European Telecommunication Standards Institute), OneM2M consortium and at Eclipse foundation. He has created the OM2M project and co-leads the OM2M eclipse open source project. He is author of more than 150 regular and invited papers in conferences and journals.