An Extended Genetic Algorithm Based on QoS Ontology for Virtual SaaS Selection in Cloud Computing

Kouchi Sana[1(✉)], Nacer Hassina[1], and Slimani Hachem[2]

[1] MOVEP Laboratory, Computer Science Faculty, University of Science and Technology Houari Boumedienne (USTHB), Algiers, Algeria
skouchi@usthb.dz, kouchi.sana21@gmail.com, hnacer@usthb.dz, sino_nacer@yahoo.fr

[2] LIMED Laboratory, Computer Science Department, University of Bejaia, 06000 Bejaia, Algeria
hachem.slimani@univ-bejaia.dz

Abstract. The selection of Software as a Service (SaaS) in cloud computing relies on providers' ability to semantically describe the non-functional properties of their services and the non-functional requirements specified by customers. In this work, we model the SaaS selection process as a multi-criteria NP-hard optimization problem. To solve this, we propose a hybrid approach that employs a genetic algorithm, leveraging semantically described micro SaaS services using an ontology. Each micro SaaS is stored alongside compatible SaaS offerings with the same functionality within a single cloud, based on the concept of compatibility. While a single micro SaaS may not fulfill all functional requirements in some cases, a composition of multiple SaaS(s), termed virtual SaaS, can meet the customer's needs. The selected SaaS must be optimal according to the customer's specified criteria. The proposed multi-criteria approach integrates a QoS ontology tailored to cloud environments. Simulation results further demonstrate that the proposed method efficiently enables significant SaaS selection within a cloud environment.

Keywords: Cloud Computing · SaaS selection · Quality of Service (QoS) · Ontology · Genetic algorithm

1 Introduction

Today, Cloud Computing has evolved beyond a simple online computing platform. Virtually anything can now be implemented and offered as a cloud service. Four primary service models have been established: *Software as a Service* (SaaS), *Platform as a Service* (PaaS), and Infrastructure as a Service (IaaS), with the recent expansion to encompass *Anything as a service* (XaaS). The emergence of Cloud Computing marks a significant shift in the delivery of services to businesses, continuing the natural evolution of distributed systems.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15648, pp. 197–210, 2025.
https://doi.org/10.1007/978-3-031-97000-9_12

According to NIST "*National Institute of Standards and Technology*", Cloud Computing is a model to enable convenient on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be quickly provisioned and released with minimal management effort or service provider interaction [1–3].

By applying Cloud Computing infrastructure, companies increase their business model capabilities and their ability to meet Cloud services demands. In fact, Cloud Computing opens a new dimension and becomes a foundation technology for developing and integrating Web applications offered as SaaS. In SaaS, the software is presented to the customers as services on demand. The concept of SaaS eliminates worries about application servers, storage, and application development and maintenance. Multiple tenants often share the software, which is automatically updated in the cloud, and no additional licence needs to be purchased [4]. Features can be requested on demand and are rolled out more frequently.

However, the number of customer services is expected to grow. Cloud service providers are required to overcome this situation by offering SaaS-based applications with several criteria. That is why the SaaS selection is becoming a big challenge and a hot topic in academia and industry. Furthermore, with increased SaaS success, many businesses provide similar services with overlapping functionalities. Thus, customers must choose between different SaaS(s) based on QoS. The latter refers to the non-functional properties of SaaS(s). Moreover, a micro SaaS may not meet a desired functional requirement in certain scenarios. However, integration and composition of a set of available SaaS may fulfill the requirement, leading to the obtaining of a virtual SaaS. Thus, selecting SaaS(s) is an NP-hard problem [5].

Given that, these services could be characterized using multiple criteria (Cost, Performance, etc.). It is important to have a methodology for selecting cloud services based on multi-criteria approaches. This work aims to present a model of semantic annotations for describing SaaS (micro or virtual [6]) and an algorithm that selects the relevant SaaS(s). The selection is useful as soon as the customer's request can be satisfied by some SaaS(s), and the relevant SaaS is the service that matches the QoS values of the service requester.

To describe the QoS of a micro SaaS, we use semantic annotations according to the specification of the *Ontology Web Language for Services* (OWL-S) and the Ontology of QoS. The request is referred to the same ontology of QoS. Ontology engineering plays a leading role in adding semantics to a service description. In the proposed solution, we use several SaaS(s) with their compatible services (i.e. those that offer the same functionality, but they can have different values of QoS). The proposed approach is a hybridization between the power of Genetic algorithms, local research, and exhaustive research. Furthermore, the approach considers the preferences of the user (objectives and constraints) and the properties of quality (quantitative and qualitative) in general.

The rest of the paper is organized as follows. In Sect. 2, we present the selection problem. Section 3 is devoted to the classification of related work concern-

ing SaaS selection. In Sect. 4, we describe the proposed approach. In Sect. 5, we exhibit a performance evaluation. Finally, in Sect. 6, we end with concluding remarks and future works.

2 Selection-Problem Complexity

When customers search for a specific SaaS, multiple options may fulfill their needs. In cases where a single micro SaaS (cloud service offered by a unique provider) is insufficient, multiple cloud service components can be combined to form a more complex service, known as a virtual SaaS [6,7]. Unlike micro SaaS, a virtual SaaS is a composite service created from various cloud services rather than being tied to a single provider. A process based on non-functional properties must be applied to select the best SaaS for a customer's needs. This paper focuses on the selection of virtual SaaS, highlighting that each component (micro SaaS) may have several compatible alternatives, each offering the same functionality but with varying QoS attributes. This increases the number of possible compositions exponentially, making the selection problem NP-hard [5] due to its multi-criteria nature.

Figure 1 illustrates the complexity of the virtual SaaS selection problem in cloud computing. Several possible execution plans, denoted as $P_1, P_2, \ldots, P_n$, represent different ways of organizing the tasks required to execute a request. Each task in a given plan is associated with a set of candidate services that can fulfill the corresponding functional requirement. To fully satisfy the request, it is necessary to select an appropriate service for each task that meets the client's QoS requirements, based on a specific execution plan. However, the number of possible combinations increases rapidly with tasks and available services, making the multi-task service selection and composition problem extremely complex.

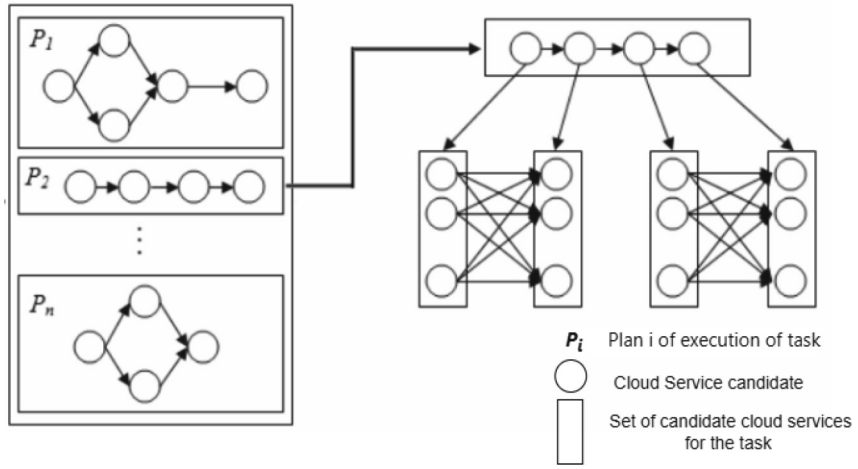

Fig. 1. Complexity of the selection problem.

3 Related Work

Several studies have investigated SaaS selection based on non-functional properties, offering a variety of approaches with distinct characteristics and capabilities. Among these, optimization techniques—particularly meta-heuristic algorithms—are notable for their effectiveness in identifying the most suitable service or combination of services to meet specific objectives.

Kouchi et al. [5] proposed an optimization approach based on the Cuckoo Search Algorithm (CSA), combined with Lévy flight, to address the optimal service selection problem in cloud computing, considering QoS constraints. Their approach consists of three phases: initialization of the population, evaluation, and searching for relevant services. Dahan et al. [8] presented a hybrid approach combining the Artificial Bee Colony algorithm (ABC) with CSA. This method aims to overcome the bee algorithm's slow convergence issues while utilizing Cuckoo Search's advantages to improve solution quality. In another study, Khatoonabadi et al. [9] proposed a Genetic Algorithm based on the Pareto Principle (GAP2WSS) to enhance web service selection for composite tasks. Focusing on the top 20% of candidate services based on QoS metrics. Additionally, Sefati et al. [10] introduced a hybrid method for cloud service selection and composition using an adaptive penalty function within genetic algorithms and the Artificial Bee Colony. This approach further refines the selection process by balancing multiple optimization factors. Kouchi et al. [11] proposed a hybrid approach for cloud service selection that combines three complementary mechanisms. First, the Whale Optimization Algorithm is used to generate new service composition plans that fulfill the required tasks. Then, the Pareto dominance principle is applied to filter and reduce the number of generated plans by retaining only the non-dominated solutions. Finally, an exhaustive search is performed on this reduced set to identify the optimal solution—that is, the best composition of services that meets the user's requirements.

Table 1 presents a comparative analysis of the cited works related to cloud service selection. While the cited works have partially fulfilled customer demands, most of these approaches suffer from high response times, which can adversely affect the effectiveness of the service selection process. To address this limitation, our proposed approach employs a Genetic Algorithm (GA) aimed at reducing response time while simultaneously enhancing customer satisfaction.

4 Proposition: Genetic Optimization Based Approach for Selecting the Optimal SaaS(s)

In the proposed solution, we focus on the selection process of virtual SaaS(s) in a unique Cloud. In fact, the proposed approach considers the objectives and the constraints of the customer. The properties of QoS are defined by using an ontology of domain to have a standard to solve the heterogeneity problem between the concepts used in the customers' requests and those used by the service providers. Furthermore, the proposed method is a comprehensive strategy that uses a

Table 1. Comparative analysis of cloud service selection approaches

Authors, Year	Approach	Selection Strategy	Ontology	Objective
Kouchi et al. [5], 2022	CSA + Lévy flight	Global search	No	Mono
Dahan et al. [8], 2023	ABC + CSA	Global search	No	Mono
Khatoonabadi et al. [9], 2021	GA + Pareto Principle	Global search	No	Multi
Sefati et al. [10], 2022	GA + ABC	Global search	No	Mono
Kouchi et al. [11], 2024	WOA + Dominance + Exhaustive Search	Hybrid search	Yes	Multi

hybrid method using local research and global research based on the concept of predominance. Thus, the Extended Genetic Algorithm (EGA) is proposed in this work.

4.1 The Proposed Architecture

Various cloud computing architectures have been proposed in the literature, such as those by NIST, IBM, HP, CISCO, and Azure [12]. In our case, we propose a suitable hybrid architecture inspired by the HP and CISCO models. The proposed architecture consists of three layers: the *Customer Access Layer* (end-user), the *Provider Access Layer*, and the *Broker Layer*, which includes three components: *Discovery* and *Composition*, *Selection*, and the *Services Layer*. Figure 2 illustrates the architecture of the proposed system in the cloud environment.

The life-cycle process of the selection process in the cloud environment is presented as follows:

1. By using the customer's layer, a customer submits a request that contains two parts of requirement: functional and non-functional properties;
2. The request is sent to the Broker layer;
3. The Broker layer returns a set of SaaS(s) which answers the functional properties of the request. The services can be micro SaaS(s) or virtual SaaS(s), where each virtual SaaS is represented by a composition plan denoted by $P_i,\ i \in \{1, ..., n\}$, with n is the number of composition plans;
4. The various composition plans $P_i,\ i = 1, ..., n$ are sent to the Selection component;
5. After the receipt of the different plans $P_i,\ i = 1, ..., n$ by the Selection component, the Broker layer generates other compatible plans $\tilde{P}_i,\ i = 1, ..., \tilde{n}$ by using Genetic Algorithm where $\tilde{n}$ is the total number of the new plans such that $\tilde{n} > n$. Then, the broker only returns the relevant composition's plans

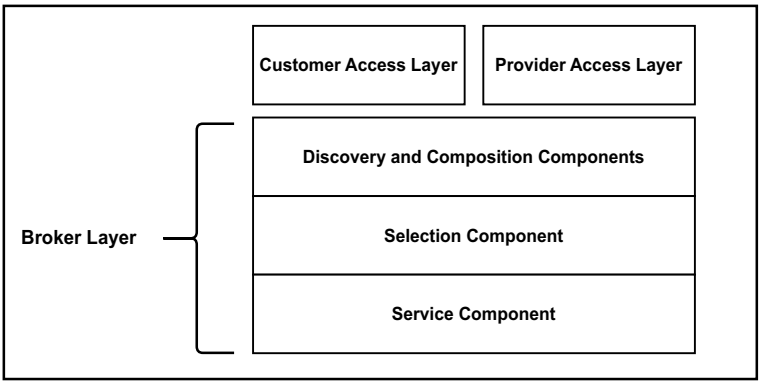

Fig. 2. The proposed architecture in the cloud environment.

among the generated compatible plans, $\tilde{P}_i,\ i = 1, ..., \tilde{n}$, which satisfy as much as possible the non-functional properties of the customer's request. This is done by using dominance selection and exhaustive research.

4.2 QoS of SaaS in Cloud Environment

Quality of Service (QoS) plays a crucial role in evaluating the selection of cloud services, as it represents the non-functional criteria of each service. Due to the diverse performance criteria for SaaS, collecting these metrics is challenging. QoS categories are often derived from customer experiences and literature on web services technology. Common QoS criteria include availability, performance, reliability, scalability, security, and usability, among others. A widely accepted model for cloud service criteria is the *Cloud Service Measurement Index* (SMI) [13].

However, in practice, SaaS(s) are developed independently by different providers, which can specify the criteria of QoS by using various vocabularies. To identify the criteria in a way that eases both the customer and the provider to express their needs and offers, we propose to use semantic annotations to define QoS criteria more clearly for both customers and providers. An ontology for the QoS domain is introduced, tailored for SaaS and cloud services, including configuration and other relevant selection and composition criteria. These additional metrics include the number of satisfied constraints, matching similarity, best-matching services, and the number of services in a composition. The proposed ontology is illustrated in Fig. 3.

4.3 SaaS Management

In order to reduce the complexity of the selection problem, we suggest that published micro SaaS(s) are ranked in the Services Layer. The micro SaaS(s) services can be ranked regarding their functionalities in several registries $E_k,\ k \in$

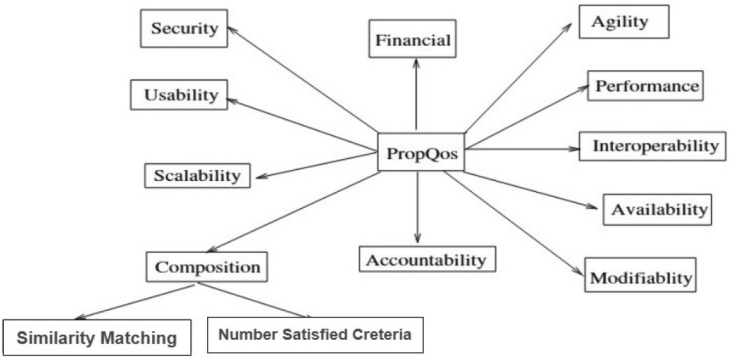

Fig. 3. Ontology of QoS.

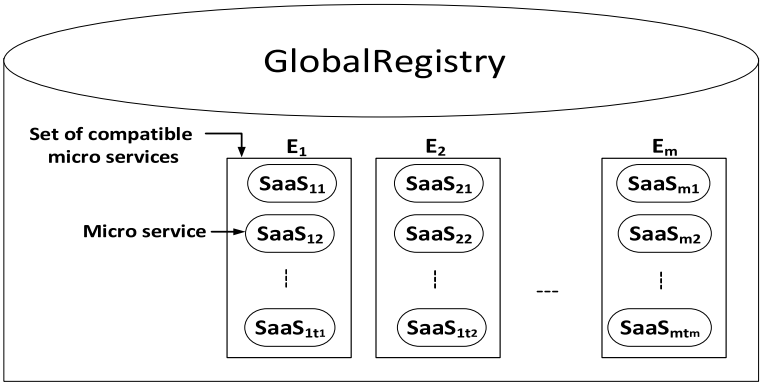

Fig. 4. SaaS management.

$\{1, ..., m\}$ in the data centers. Each registry E_k gathers a set of compatible micro SaaS(s) that have the same functionality $F_k,\ k \in \{1, ..., m\}$. The idea of grouping the micro SaaS(s) according to their functionalities is inspired by the work of Baridi [14]. Formally, we have the following sets:

- $GlobalRegistry = \{E_1, ..., E_m\}$ is the register of the whole micro services,
- $E_k = \{SaaS_{kj}, j \in \{1, ..., t_k\}\}$ is the set of micro services corresponding to the $k-th$ functionality, with t_k is the number of compatible micro services of E_k,
- $SaaS_{kj},\ j \in \{1, ..., t_k\}$, is the $j-th$ micro service belonging to the set $E_k,\ k \in \{1, ..., m\}$.

Figure 4 illustrates the organization of the global registry. An ontology OWL-S [15] is defined to allow a semantic description of the functional properties of a micro SaaS. Furthermore, to describe the non-functional properties of QoS, we add a new criterion 'Q' represents the values of the properties of QoS. Thus, a

microservice $SaaS$ in OWL-S is described as follows:

$$SaaS = (I, O, Pr, Ef, Q),$$

where:

- I is the set of inputs;
- O is the set of outputs;
- Pr is the set of pre-conditions;
- Ef is the set of the effects of the service;
- Q is the set of properties of QoS of the service,

where a given property of QoS $q \in Q$ is represented as follows: $q = (C, V, T)$, such that

- C is the concept of the proposed ontology of QoS (Financial (Cost), Performance (Response Time), etc.);
- V represents the value offered by the service $SaaS$. This value can be quantitative or qualitative;
- T represents the type of the value V (Integer, Real, String, etc.).

4.4 Selection Process

The selection process is illustrated by Fig. 5. The selection process receives two inputs: the set of plans $P = \{P_1, ..., P_n\}$ and the non-functional properties of the customer's request. In the first step, we apply the proposed selection algorithm based on the basic Genetic Algorithm (GA). The obtained result is used as input for the second step, corresponding to the domination selection. The latter aims to reduce the number of plans by eliminating the SaaS(s) that do not offer good performance. Note that the domination selection is based on the degree of similarity matching [16]. Finally, a refinement is applied in the third step, corresponding to the exhaustive research.

4.5 The Proposed Algorithm

The proposed algorithm EGA of SaaS Selection is constituted from three steps presented as follows:

(1) Generation of Best Composition Plans The objective of the first step is to find for each plan $P_i,\ i \in \{1, ..., n\}$ the best composition corresponding to this plan. For this, we apply a Genetic Algorithm (GA). It is known that basic GA is a powerful tool for solving combinatorial optimization problems. Furthermore, it is well suited for rapid and total exploration of a space of research of an important size, like a cloud environment. Note that the blue components are the same as those used in the basic GA. At the end of the EGA step, we obtain

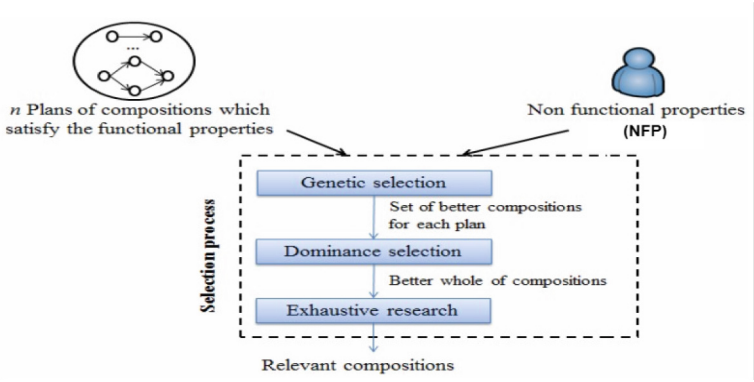

Fig. 5. Selection component.

for each plan $P_i,\ i \in \{1,\ldots,n\}$, a set of r_i compositions denoted by $BestP_i$ such that

$$BestP_i = \left\{\tilde{P}_{i1}, \tilde{P}_{i2}, \ldots, \tilde{P}_{ir_i}\right\},$$

where

$$\begin{aligned}
\tilde{P}_{i1} &= \left\{SaaS^1_{i1}, SaaS^2_{i1}, \ldots, SaaS^{n_{i_1}}_{i1}\right\},\\
\tilde{P}_{i2} &= \left\{SaaS^1_{i2}, SaaS^2_{i2}, \ldots, SaaS^{n_{i_2}}_{i2}\right\},\\
&\vdots \quad \vdots\\
\tilde{P}_{ir_i} &= \left\{SaaS^1_{ir_i}, SaaS^2_{ir_i}, \ldots, SaaS^{n_{ir_i}}_{ir_i}\right\}.
\end{aligned}$$

Note that r_i is the number of plans generated by EGA from the plan $P_i,\ i \in \{1,...,n\}$. In this way, EGA generates $\tilde{n} = r_1 + r_2 + ... + r_n$ new plans. These results are used as inputs for the following second step.

(2) Dominance Selection (Pareto Selection) This step is based on the concept of *dominance*, which is defined in the literature as follows. We consider a problem of maximization. Let x and y be two vectors of N dimensions. It is stated that x *dominates* y, if the following conditions are satisfied:

- $x_i \geq y_i, \forall\ i \in \{1,\ldots,N\}$;
- $\exists\ j \in \{1,\ldots,N\}$ such that $x_j > y_j$.

In the previous step, for each plan, $P_i,\ i \in \{1,...,n\}$ a set of compositions $BestP_i = \left\{\tilde{P}_{i1}, \tilde{P}_{i2}, \ldots, \tilde{P}_{ir_i}\right\}$ is generated by EGA. In the current step, for each set $BestP_i,\ i \in \{1,...,n\}$ is associated a set of vectors $V_i = \{V_{i1}, V_{i2}, ..., V_{ir_i}\}$ where for all $l \in \{1,...,r_i\},\ V_{il} = (FDM_{il}, NSC_{il})$ such that :

- FDM_{il}: the functional degree of matching between P_i and $\tilde{P}_{il}$;
- NSC_{il}: the non-functional degree of matching, which is the number of satisfied constraints between $\tilde{P}_{il}$ and the non-functional properties of the customer's request.

For all $i \in \{1, ..., n\}$, we keep only the vectors $V_{il},\ l \in \{1, ..., r_i\}$ that are not dominated in the set V_i. These results are used as inputs for the next final step.

(3) Exhaustive Research In this step, we construct other plans from the non-dominated plans by using the concept of compatibility of microservices. More specifically, for each non-dominated plan $\tilde{P}_{il} = \left\{SaaS^1_{il}, SaaS^2_{il}, \ldots, SaaS^{n_{x_i}}_{il}\right\},\ i \in \{1, ..., n\},\ l \in \{1, ..., r_i\}$, we construct other compatible plans by using for all microservice $SaaS^s_{il},\ s \in \{1, ..., n_{x_i}\}$, of $\tilde{P}_{il}$ its set of compatible microservices A^s_{il} from $E_k,\ k \in \{1, ..., m\}$ (see Figs. 4 and 6). In this way, we obtain a total number of compatible plans which is smaller than the number of plans of the initial population $\bigcup_{i=1}^{n} BestP_i$ because we have eliminated all the dominated plans in the step of dominance.

An exhaustive research is carried out on the sets $A^s_{il},\ i \in \{1, ..., n\},\ l \in \{1, ..., r_i\},\ s \in \{1, ..., n_{x_i}\}$ as depicted in Fig. 6. Thus, we test all possible combinations of services to find the best compositions.

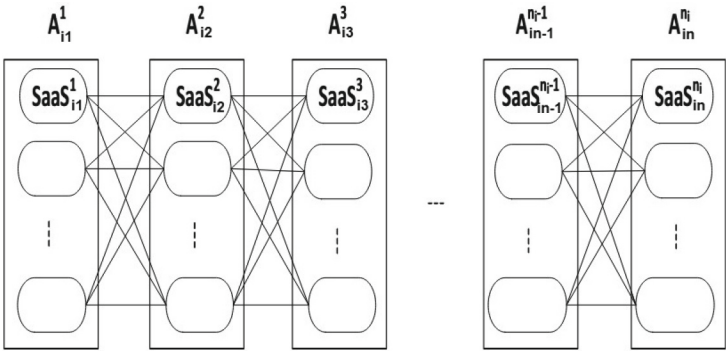

Fig. 6. Exhaustive research step.

Hence, the result of this last step is a set of compositions that will be returned to the customer. The latter has to choose the best composition that satisfies their non-functional needs.

5 Performance Evaluation

To validate the feasibility of the proposed approach (EGA), we construct a simulation experiment system based on the Cloud Computing experiment platform via CloudSim simulator. All experiments were performed in the environment of Microsoft Windows 10 (64-bit) system, Intel R© core i7 CPU 2.90 GHz, and 12 GB RAM, and all algorithms were implemented in JAVA.

Furthermore, we realize some experiment simulations, such as comparing the proposed approach, the basic Genetic Algorithm, and the Exhaustive Search. For

that, we suppose that each micro-published SaaS is available with its expected QoS; quantitative or qualitative criteria.

For the experiments, we assume that the Broker, especially the components "Discovery and Composition", returns three plans of composition for a customer's request about a trip plan service, which is composed of microservices, including booking a flight, booking a hotel, renting a car, etc. An optimal trip achieves some objectives, e.g., minimizing the travel duration and minimizing the cost. In the series of experiments, we varied the size of the research space to prove that the proposed solution is suited for a high number of cloud services compared to the Exhaustive Research and the basic Genetic Algorithm.

Scenario 1: Case of three criteria

Let a customer's request contain three criteria: Financial (cost of invocation), Performance (execution time), and Composition to prove the front of the Pareto. The customer wants to minimize the execution time and the cost of invocation simultaneously, with the highest degree of matching similarity for functional properties and a high number of constraints for non-functional properties. Figure 7 and Fig. 8 respectively illustrates the results obtained with 5 and 20 Compatible services per component service. One notices on Fig. 7 that our proposal finds the border of Pareto contrary to the basic Genetic Algorithm.

With the increase of compatible services, our proposal finds optimal solutions that belong to the front of Pareto, but does not manage to find the totality of the front of Pareto. The obtained results are shown in Fig. 8.

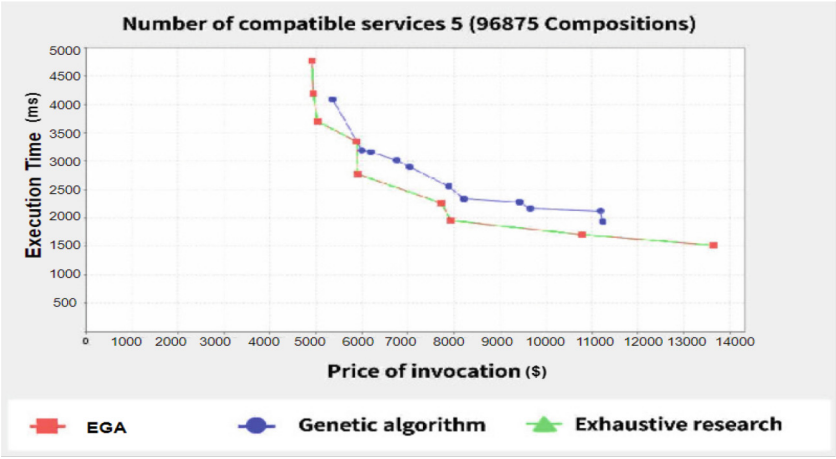

Fig. 7. Results (5 compatible SaaS(s)).

Scenario 2: case of five criteria

In this experiment, we check the validity of the proposed approach to satisfy five constraints in the customer's request: Performance (execution time),

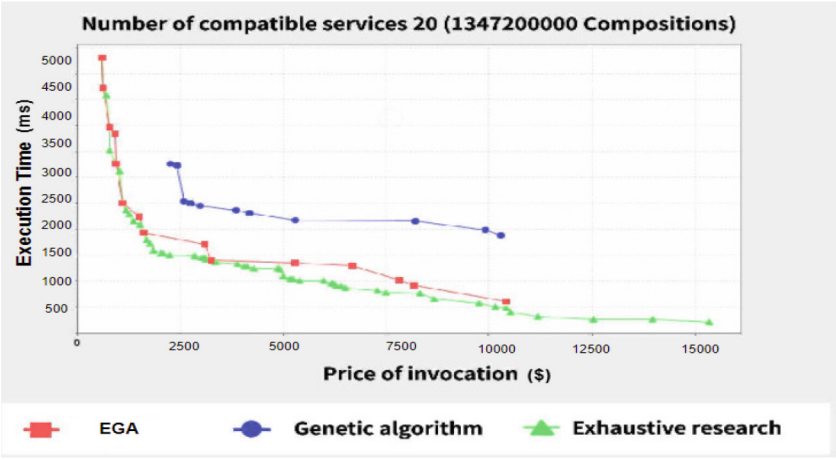

Fig. 8. Results (20 compatible SaaS(s)).

Financial (cost), Availability, Reliability, Composition. The customer wants to minimize the execution time to maximize the Reliability and maximize the degree of similarity matching under two constraints, such that $Cost \leq 14\$$ and $Availability > 85\%$. Table 2 illustrates the results obtained by our proposal on a set of 10 compatible services per component service.

Table 2. The results of the application of the proposed algorithm EGA

N	Financial	Performance	Availability	Reliability
1	$12.0	390.0 ms	85.138091%	61.99854%
2	$11.0	409.0 ms	87.801845%	64.99589%
3	$10.0	359.0 ms	86.281206%	60.00177%
4	$13.0	300.0 ms	90.916292%	53.34019%
5	$14.0	334.0 ms	85.188091%	63.01589%
6	$13.0	361.0 ms	86.088604%	40.03999%
7	$14.0	219.0 ms	92.078371%	29.63348%
8	$11.0	320.0 ms	90.008433%	61.60209%
9	$14.0	392.0 ms	95.594545%	28.84362%

The performance of our EGA algorithm is significantly reflected in the results obtained. If we carefully examine Table 2, which presents the selected services, it is clear that our approach ensures a precise match between the SaaS selection and the customer's specific needs.

Notably, the selected services are characterized by a cost not exceeding 14$. At the same time, these services show high availability, reaching or exceeding

85%, thus fully meeting the customer's needs. These results provide convincing evidence of the outstanding performance of our system in the selection and composition of SaaS(s), ensuring successful optimization between different constraints to meet customer requirements.

6 Conclusion

Quality of Service (QoS) is crucial for customers when selecting SaaS services. This paper introduced a novel approach to SaaS selection that centers on using ontologies. Furthermore, we mathematically formalized the SaaS selection problem and presented a multi-criteria selection methodology based on an extended genetic algorithm. Moreover, the quality criteria defined above in the selection context can be used to evaluate the quality of micro and virtual SaaS services. Thus, we can conclude that the proposed approach contributes to solving some problems with SaaS selection, which is a complex topic. However, future work could extend this approach by considering the selection of SaaS services in heterogeneous cloud environments, focusing mainly on reliability, elasticity, and security criteria.

References

1. Ayachi, M., Nacer, H., Slimani, H.: Cooperative game approach to form overlapping cloud federation based on inter-cloud architecture. Clust. Comput. **24**, 1551–1577 (2021)
2. Ayachi, M., Nacer, H., Slimani, H.: Correction to: cooperative game approach to form overlapping cloud federation based on inter-cloud architecture. Clust. Comput. **24**, 1579–1582 (2021)
3. Ayachi, M., Nacer, H., Slimani, H.: Cloud computing interoperability: an overview. In: International Conference on New Technologies of Information and Communication (NTIC 2022), Mila, Algeria (2022)
4. Tsai, W., Sun, X., Balasooriya, J.: Service-oriented cloud computing architecture. In: Proceeding of the Seventh International Conference on Information Technology, pp. 684–689 (2010)
5. Kouchi, S., Nacer, H., Slimani, H.: Service selection in cloud computing environment by using cuckoo search. In: International Conference Advances in Information, Communication and Cybersecurity(ICI2C), pp. 219–228 (2021)
6. Nacer, H., Djebari, N., Slimani, H., Aissani, D.: A distributed authentication model for composite web services. J. Comput. Secur. **70**, 144–178 (2017)
7. Nacer, H., Aissani, D.: Semantic web services: standards, applications, challenges and solutions. J. Netw. Comput. Appl. (JNCA) **44**, 134–151 (2014)
8. Dhan, F., Alwabel, A.: Artificial bee colony with cuckoo search for solving service composition. Intell. Autom. Soft Comput. **35**, 3385–3402 (2023)
9. Khatoonabadi, S., Lotfi, S., Isazadeh, A.: GAP2WSS: a genetic algorithm based on the pareto principle for web service selection. Neural Evol. Comput. (2021)
10. Sefati, S.S., Halung, S.: A hybrid service selection and composition for cloud computing using the adaptive penalty function in genetic and artificial bee colony algorithm. Sensors **22**, 4873 (2022)

11. Kouchi, S., Nacer, H., Slimani, H.: Multi-criteria based virtual SAAS selection in cloud environment. In: Global Congress on Emerging Technologies (GCET-2024), pp. 125–131. IEEE (2024)
12. Kouchi, S., Nacer, H., Beghdad-bey, K.: Towards a reference architecture for interoperable clouds. In: 8th International Conference on Electrical and Electronics Engineering (ICEEE), pp. 229–233 (2021)
13. Siegel, J., Perdue, J.: Cloud services measures for global use: the service measurement index (SMI). In: Proceeding of Annual SRII Global Conference, pp. 411–415, San Jose, USA (2012)
14. Badidi, E.: A framework for software as a service selection and provisioning. Int. J. Comput. Netw. Commun. (IJCNC) **5**(3), 189–200 (2013)
15. Ouchaou, L., Nacer, H., Slimani,H., Boukria, S.: Semantic networks based approach for SAAS management in cloud computing. In: Proceeding IEEE International Conference on Smart Communications in Network Technologies (SaCoNeT), pp. 2255–260 (2018)
16. Chen, H., Dou, Z., Hao, X., Tao, Y., Song, S., Sheng, Z.: Enhancing multi-field B2B cloud solution matching via contrastive pre-training (2024)