Cloud computing is a pattern for delivering ubiquitous and on demand computing resources based on pay-as-you-use financial model. Typically, cloud providers advertise cloud service descriptions in various formats on the Internet. On the other hand, cloud consumers use available search engines (Google and Yahoo) to explore cloud service descriptions and find the adequate service. Unfortunately, general purpose search engines are not designed to provide a small and complete set of results, which makes the process a big challenge. This paper presents a generic-distrusted framework for cloud services marketplace to automate cloud services discovery and selection process, and remove the barriers between service providers and consumers. Additionally, this work implements two instances of generic framework by adopting two different matching algorithms; namely dominant and recessive attributes algorithm borrowed from gene science and semantic similarity algorithm based on unified cloud service ontology. Finally, this paper presents unified cloud services ontology and models the real-life cloud services according to the proposed ontology. To the best of the authors’ knowledge, this is the first attempt to build a cloud services marketplace where cloud providers and cloud consumers can trend cloud services as utilities. In comparison with existing work, semantic approach reduced the execution time by 20% and maintained the same values for all other parameters. On the other hand, dominant and recessive attributes approach reduced the execution time by 57% but showed lower value for recall.

© 2017 Production and hosting by Elsevier B.V., on behalf of Cairo University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Introduction

Cloud computing is considered the fifth utility [1] after water, electricity, telephone and gas based on pay-as-you-use model. There are three abstract delivery models for cloud services: (SaaS, PaaS, and IaaS) [2]. In Software as a Service (SaaS), consumers use applications running on providers’ infrastructure. In Platform as a Service (PaaS), consumers deploy applications onto providers’ infrastructure. Finally, in Infrastructure as a Service (IaaS), consumers deploy arbitrary software and have a full access to the operating system. Cloud service providers advertise cloud service descriptions on websites and portals. Advertisement contains flat text descriptions, images, tables and files. Cloud service discovery and selection process becomes a significant challenge because of exponential growth in the number of cloud service providers. Nowadays, finding the appropriate cloud service is a time-consuming and tedious task. Consumer uses the available search engines like (Google, Bing and Yahoo) with appropriate keywords to find all cloud provider websites, then they make a list of all available services with their features. Finally, the consumer selects the best appropriate service and uses it. Unfortunately, available search engines are not designed to give a small set of exactly matching cloud services. On the contrary, existing search engines show all websites that have the search keywords without any semantic matching like (ParkCloud, CurrencyCloud [3]). Buyya et al. [4] wrote in 2013 that “the discovery of cloud services is mostly done by human intervention: a person (or a team of people) looks over the Internet to identify offerings that meet his or her needs. We imagine that in the near future it will be possible to find the solution that matches our needs by simply entering our request in a global digital market that trades cloud computing services.” They added: “In this cloud marketplace, cloud service providers and consumers, trading cloud services as utilities”. Techniques used for web service discovery and selection [5] cannot be adopted for cloud services because of their special characteristics. This work presents a generic framework that serves as a template for cloud service marketplace. In this marketplace consumer can submit a request for cloud service and get a ranked list of the best matching services. The proposed framework is divided into six subsystems and thirteen components. Academic and industrial bodies can create instances of this framework by adopting different methods and approaches for each component. Additionally, this work presents a domain ontology for cloud services to create a shared understanding of the cloud environment and model the real-life cloud services according to the proposed ontology. Furthermore, this work implements two instances of generic framework by adopting two different matching algorithms. The first one is the dominant and recessive attributes algorithm borrowed from gene science, and the second one is the semantic similarity algorithm based on unified cloud service ontology. The contributions of this paper are:

- Presenting a generic framework for cloud service marketplace.
- Presenting cloud service domain ontology.
- Modeling the real-life cloud services according to domain ontology.
- Presenting percent distance similarity algorithm for cloud services matching.
- Building two instances of cloud services marketplace and compare them with existing work.

The rest of this paper is organized as follows: Section 2 surveys the existing researches in cloud service discovery and selection domain; Section 3 presents generic framework for cloud services marketplace; Section 4 presents cloud service domain ontology; Section 5 presents experiments and results; and Section 6 is a conclusion of the work.

Related work

Researches in the area of cloud service discovery and selection process can be divided into the following categories:

Multi-criteria decision making approaches

Multi-Criteria Decision Making (MCDM) is a set of methodologies used to select the best matching in case of multiple alternatives with multiple attributes [6]. Park and Jeong [7] proposed a model for cloud service discovery based on MCDM approach with six criteria: Functionality, Reliability, Usability, Efficiency, Maintainability and Business. Godse and Mulik [8] presented an approach to select SaaS based on Analytic Hierarchy Process (AHP) and expert survey respondents. The problem with MCDM approach is completely ignoring the relationships between the different parameters.

Performance analysis approaches

Qu et al. [9] presented a cloud service selection system based-on user’s feedback and performance analysis. The proposed system aggregates the feedback from cloud users and the objective performance measurement from a third party. Rehman et al. [10] presented cloud services monitoring system based-on user experience feedback approach. System assumption is a cloud service that satisfies existing applications with specific usage profiles similar to new application, which is the best cloud services for new application. Unfortunately, Performance indicator may not be enough to judge the best matching cloud service and there is no way to check the reliability of users’ feedback.

Agent based approaches

Maheswari and Karpagam [11] presented an agent base and multilayered architecture to facilitate service discovery in cloud environment. Reshma and Balaji [12] proposed agents model for cloud service publication, discovery and selection, where clients can submit requests and matching attributes through user interface. There is no concrete approach for cloud service discovery and selection process in these two proposed works. Sim [13] developed cloud services discovery system based on Multi-Agents and search engine. This work doesn’t consider QoS parameters.

Semantic approaches

Tahamtan et al. [14] introduced a semantic framework that provides query capability based on unified cloud ontology and business service ontology. However, service matching is done based on SPARQL that need experienced users. Afify et al. [15] developed a unified ontology that serves as semantic based repository to facilitate SaaS publication, discovery and selection processes. This work focused on SaaS only and didn't consider PaaS and IaaS. Hasan et al. [16] proposed service discovery system based on hierarchal ontology. This work assumed the existence of local ontology in each cloud provider which is not applicable in the real world.

Other approaches

Zhang et al. [17] presented two-level cloud service directories for cloud services discovery. Unfortunately, this assumption is not applicable in the real world. Somu et al. [18] presented architecture for cloud services discovery based-on Hyper-graph Computational Model (HGCM) and Minimum Distance-Helly Property (MDHP) algorithm [19]. This work didn't provide a clear
architecture or framework for cloud services marketplace. Abour-ezq and Idrissi [20] presented cloud service search and selection system based-on Skyline algorithm [21] which is able to capture the numerical attributes only. Garg et al. [22] proposed a framework (SMICloud) for comparing and ranking cloud services based on Service Measurement Index (SMI). This work focused on QoS attributes only. Jia-jing et al. [23] presented two levels clustering model for cloud services discovery. This work assumed the existence of WSDL file for each cloud service which is not accepted from all cloud service providers. Barati and St-Denis [24] proposed formal approach for service matching based-on formal methods. This work focused on cloud services composition. Authors in [25,26] proposed an automated system for cloud services selection based on machine readable format XML. XML approach is based on predefined format that is not available on all cloud service advertisements. Lu and Xu [27] proposed a system for cloud service composition based on SPARQL. Unfortunately, SPARQL language need experienced users. Rekik et al. [28] proposed an end to end Business Process Outsourcing (BPO) framework for cloud services.

Summary of key findings

In spite of the considerable amount of research that was done in the field of cloud services discovery and selection, there is no unified understanding or shared concepts through these studies. Almost all studies proposed virtual solutions without considering the current status of cloud service providers. In addition to that, there is no complete framework or architecture that covers the total cloud services discovery and selection process. Hence, this work presents a cloud service marketplace to: (i) automate cloud services discovery and selection process; (ii) reduce the time and effort of finding cloud services; (iii) make service providers more visible to all consumers; (iv) create a shared understanding of cloud service domain and (v) improve the overall user experience.

Framework for cloud services marketplace

Cloud services marketplace facilitates the process of finding the appropriate cloud service that meets cloud consumer requirements. Cloud services marketplace collects cloud service advertisements from provider websites automatically or semi-automatically. On the other hand, it receives consumer queries to find the best matching services and displays the results in ordered list. Generic framework presents a template to formalize cloud services discovery and selection process. As shown in Fig. 1 generic framework is divided into six subsystems and thirteen components as follows:

User interface subsystem

It is a graphical interface that facilitates the communication between end user and CSDS. User Interface contains three components as follows:

- Query Receiver receives user queries in different formats. User enters plain text queries or uses some predefined lists, checkboxes or radio buttons to enter the queries.
- Results Viewer displays the matching results for user as a ranked list. User can change ranking preferences to see the different order of matching services.
- User Profile monitors user query to predict the user behavior and give recommendations.
Query handler subsystem

Query Handler Subsystem receives user queries and returns a ranked list of matching services. It contains three components as follows:

- Query Translator receives the user queries to extract the semantic query based on cloud ontology. Query Translator could use Natural Language Processing (NLP) approach to convert plain text query or predefined query into semantic query.
- Service Matching component contains the algorithm and approach that CSDS will implement to find the best service for cloud consumer.
- Service Ranking component ranks the matching services based on user preferences.

Cloud ontology subsystem

Cloud ontology facilitates the semantic reasoning between user request and available cloud services by providing a shared understanding of the cloud services domain. Cloud ontology Subsystem contains two components as follows:

- Domain ontology contains cloud service concepts into hierarchical taxonomy structure to provide a shared language in cloud services domain.
- Relationship ontology contains individual of domain ontology and relationship among them. It provides a common understanding of cloud services domain between all CSDS components and actors.

Service collector subsystem

Service collector collects cloud service advertisements published by cloud service providers in different formats. Service Collector Subsystem consists of two components as follows:

- Service Detector collects cloud service descriptions with different formats. Generally there are two approaches for service detection: the first one is the crawler search engine where cloud providers advertise their services on websites without any communication with CSDS, and the second is the registry approach where cloud service providers need to register their services by direct communication with CSDS.
- Service Identifier classifies and categorizes discovered cloud services based on different methods and techniques.

Services repository subsystem

Service Collector Subsystem maintains an up-to-date services repository. This repository contains all available cloud service descriptions with their semantic representation.

Service monitoring subsystem

Service Monitoring Subsystem ensures that cloud service meets the Service Level Agreement (SLA) and provides the feedback to Query Handler Subsystem. Service Monitor consists of two components as follows

- User feedback collects consumer feedbacks about cloud service performance.
- Third party monitoring monitors cloud service providers to ensure that cloud provider meets the SLAs.

Cloud services ontology

Cloud service providers publish service advertisements on the internet using various formats and only 1.8% of available cloud services have a semantic description [29]. Some cloud providers do not mention any word related to cloud in their names like (drop-box). On the other hand, some other organizations, which are not related to cloud services, may use the word “Cloud” in their names like (ParkCloud, CurrencyCloud). Furthermore, cloud providers use different words to advertise the same concept like: “Amazon Workspaces” [30], “Desktop as a Service” [31], which makes the cloud services discovery and selection process more complicated. Lack of standards for cloud service advertisements is considered to be a big challenge for cloud services discovery and selection [32]. To overcome this challenge, this work built a cloud service domain ontology based on NIST [2] Cloud Computing Reference Architecture, other standards [33–35] and information collected from cloud service provider websites as shown in Fig. 2. Cloud service domain ontology describes all the concepts and relationships between concepts to create a shared understanding in cloud service domain. Unified cloud services ontology enables data and application interoperability amongst different cloud services. Additionally, unified cloud services ontology facilitates cloud service portability between different cloud providers. Furthermore, cloud services ontology enables automatic discovery and composition of cloud services. Finally, cloud services ontology eases the Service Levels Agreement (SLA) management.

Semantic similarity based on cloud ontology

Semantic similarity determines how much a concept A is related to the concept B. Researchers use different factors to calculate the semantic similarity based on ontology, such as, the path length and depth [36] and information content in each node [37]. Andreasen [38] considered generative nature of the ontology and presented a semantic similarity algorithm based on shared nodes between concepts as follows:

![Fig. 2. Classes of cloud services domain ontology.](image)
\[ \text{Sim}_{\text{And}}(x, y) = \rho \frac{|\alpha(x) \cap \beta(y)|}{|\alpha(x)|} + (1 - \rho) \frac{|\alpha(x) \cap \beta(y)|}{|\alpha(y)|} \]  
\[ (1) \]

\( \alpha(x) \) and \( \alpha(y) \) are the set of upwards nodes reachable from \( x \) and \( y \) respectively. \( \alpha(x) \cap \beta(y) \) represents the number of shared reachable nodes between \( x \) and \( y \). The value of \( \rho \in [0, 1] \) represents the degree of influence. Based on the ontology in Fig. 3, the calculation of semantic similarity between two operating systems like Windows 8 and Mac was conducted as follows: \( \alpha(\text{Windows8}) = 4 \), \( \alpha(\text{Mac}) = 2 \) and \( \rho = 0.5 \) then \( \text{SSim}(\text{Windows8, Mac}) = 0.5 \).

**Numerical similarity**

Semantic similarity is responsible for similarity between concepts. On the other hand, Numerical similarity is responsible for similarity between the attribute values of these concepts. As an example, assume that cloud consumer is looking for a solution with 10 GB RAM and EC2 offers a solution with 12 GB RAM while GoGrid offers a solution with 1 GB RAM. It is very clear that EC2 solution is more similar to consumer request than GoGrid solution. Cloud service marketplace need to retrieve all alternatives with matching score bigger than threshold. Numerical similarity gained less interest from researchers than Semantic similarity. Kang and Sim [39] present a numerical similarity algorithm based on user requested attribute value and the min and max value of this attribute as follows:

\[ \text{Sim}_{\text{SimAnd}}(x, y, a) = 1 - \frac{|x - y|}{\max(\max a - x, x - \min a)} \]  
\[ (2) \]

\( |x - y| \) represents the distance between user request \( x \) and alternative service \( y \). max \( a \) and min \( a \) represent the max and min value in attribute \( a \) for all available cloud services respectively. To overcome the limitations and disadvantages of the previous algorithms, this paper presents percent distance similarity (PDSim) algorithm. Proposed algorithm was based only on the requested attribute value and independent of max and min value of these attribute as follows:

\[ \text{PDSim}(x, y, a) = \begin{cases} 1 - \frac{x - y}{\max a}, & y < 2x \\ 0, & y \geq 2x \end{cases} \]  
\[ (3) \]

If \( y < 2x \) then \( y \) is similar to \( x \) and the similarity value is \( \text{PDSim}(x, y) \).

If \( y \geq 2x \) then the distance between attributes is big and similarity is zero, so cloud consumer need to change the query to get different results.

**Experiments**

Two instances of generic framework are implemented as follows: the first one was based on dominant and recessive attributes approach borrowed from gene science and the second one was based on ontology semantic similarity approach.

**Dominant and recessive attributes approach**

Based on the concept of dominant and recessive attributes borrowed from gene science, the cloud service attributes are divided into master attributes (dominant) and slave attributes (recessive). The existence of all dominant attributes is necessary to accept the cloud service as an alternative and the absence of only one dominant attribute is enough to reject the cloud service. On the other hand, the existence or absence of the recessive attributes is only affecting the matching score of cloud service alternative. Recessive attributes similarity is calculated based on Eq. (3). Similarity between two dominant attribute values \( y, z \) is calculated as follows:

\[ \text{DSim}(y, z) = y \wedge z \]  
\[ (4) \]

\( y, z \in [0, 1] \): 0 and 1 represent the presence or existence of the attribute respectively.

Matching score between user request and cloud service alternative is a product of the total similarity of all dominant attributes (DSim) and the total percent distance similarity of all recessive attributes (PDSim) as follows:

\[ \text{ms} = \frac{\sum_{i=1}^{u} \text{PDSim}(cs^{ai}, ur^{ai}) + \sum_{j=1}^{v} \text{DSim}(cs^{aj}, ur^{aj})}{u} \]  
\[ (1) \]

\( v \) is the number of dominant attributes and \( u \) is the number of recessive attributes. If \( \text{ms} > \text{th} \) then cloud service is accepted as alternative. \( \text{Th} \) is matching threshold.

**Semantic similarity based on ontology approach**

This approach divides cloud service attributes into two types: numerical and non-numerical. Numerical attributes similarity was calculated based on percent distance similarity algorithm (Eq. (3)), while non-numerical attributes similarity was calculated based on ontological semantic similarity (Eq. (1)). Matching score between user request and cloud service alternative is average of the total percent distance similarity (PDSim) of all numerical attributes and the total semantic similarity (SQ) of all non-numerical attributes as follows:

\[ \text{ms} = \left( \frac{\sum_{i=1}^{u} \text{PDSim}(cs^{ai}, ur^{ai})}{u} + \frac{\sum_{j=1}^{v} \text{SQSim}(cs^{aj}, ur^{aj})}{v} \right) / 2 \]  
\[ (2) \]

\( u \) is the number of numerical attributes and \( v \) is the number of non-numerical attributes. If \( \text{ms} > \text{th} \) then cloud service is accepted as alternative. \( \text{Th} \) is matching threshold.

**Results and discussion**

One more contribution of this paper is collecting cloud service advertisements from providers’ websites and modeling them according to the proposed unified ontology. Cloud service ontology is built using protegé. The following paragraphs compare between two approaches for cloud service matching. The first one is dominant and recessive attributes approach (non-semantic query NSQ) presented in 5.1 and the second one is ontology semantic similarity approach (semantic query SQ) presented in 5.2. Work
proposed by Kang and Sim [39] is considered as a reference point and it will be referred as SimQ. Comparison based on four parameters: number of matching services, execution time, average score, recall and precision was carried out. Fig. 4 shows ontology cloud service marketplace results (SQ) for user requesting DaaS service with following parameters (Vcpu = 4, Ram = 10 GB, Storage = 75 GB, Availability = 99%, Price = 30 USD/month, Location = India, OS = Win, Backup = yes and \(\text{th} = 0.9\)).

Number of matching services

Fig. 5 shows the number of matching services for each query based on different values of threshold. Semantic query (SQ) and SimQ showed almost the same number of matching services for all threshold values. Semantic query uses cloud ontology to determine all equivalent class and retrieve the matching cloud services. On the other hand, non-semantic query (dominant and recessive attribute approach) showed lower number of matching services, because it will consider cloud service as an alternative only if all dominant attributes are available. The bigger number of matching services requires more effort from user to find the best one. On the other hand, the smaller number of matching services means less chance for user to find the appropriate service.

Average matching score

Average matching score is an important parameter for cloud services marketplace matching algorithms. It is playing the main role in determining the best matching threshold for cloud services marketplace to satisfy cloud consumer requirements with lowest time and effort. Low value of matching threshold results in huge number of matching services and will increase the work load for the cloud consumer. On the other hand, higher value of matching threshold will result in very low number of matching services and may lose the opportunity to find the appropriate service. As shown in Fig. 6 semantic query (SQ) and SimQ have almost the same number of average matching score for all threshold values because of adding the semantic similarity to matching score. On the other hand, non-semantic query maintain a lower values of average matching score for all threshold values.

Execution time

Execution time is an important factor for any information retrieval system. The success of cloud services marketplace is
depending on the time that it takes to retrieve the matching cloud services. Execution time is the period between submitting the query and displaying the results on the screen for cloud consumer. It does not include the time needed to collect cloud service advertisements, identify and classify those cloud services or update cloud ontology with new concepts. As shown in Fig. 7, semantic query (SQ) takes less execution time than SimQ for all threshold values, because of its numerical algorithm as shown in paragraph (Numerical similarity). On the other hand, non-semantic query (NSQ) shows the lowest execution time for all threshold values because it does not consult the cloud ontology and the absence of only one dominant attribute is enough to reject the cloud service.

**Recall and precision**

Number of matching services does not show the accuracy and efficiency of information retrieval system. Recall and precision evaluate the completeness and effectiveness of information retrieval system. As shown in Fig. 8, semantic query (SQ) and SimQ showed almost the same recall percentage for all threshold values. On the other hand, non-semantic queries (NSQ) showed low percent of recall for all threshold values, because the absence of only one dominant attribute is enough to reject the cloud service. The precision value was 1 for all queries, since there were no false retrieved services.

**Conclusions**

Cloud services marketplace is urgently needed to remove the barriers between service providers and consumers and to automate cloud services discovery and selection process. This work is the first attempt to build cloud services marketplace where cloud service providers and consumers can trade cloud services as utilities. This paper presented a generic framework for cloud services marketplace based on unified cloud ontology. Additionally, this paper implemented two instances of this framework, one is based on the dominant and recessive attributes approach and other is based on the ontological semantic similarity approach. Comparison between these two approaches was conducted based on four parameters (number of matching services, execution time, average score, recall and precision). Semantic approach based on cloud ontology reduced the execution time by 20% and maintained the same values for all other parameters as SimQ. On the other hand, non-semantic approach (dominant and recessive attributes approach) reduced the execution time by 57%, but showed lower value for recall. As a future work, collecting more cloud services will improve the unified cloud services ontology.

**Conflict of Interest**

The authors have declared no conflict of interest.

**Compliance with Ethics Requirements**

This article does not contain any studies with human or animal subjects.

**References**

[1] UCLA History. A crash into the future. Available from: <https://alumni.ucla.edu/ucla-history/ucla-history-45/>.
[2] SP 800-145. The NIST definition of cloud computing (September 2011) Available from: http://dx.doi.org/10.6028/NIST.SP.800-145.
[3] ParkCloud. Available from: <https://www.parkcloud.com>. CurrencyCloud. Available from: <https://www.currencycloud.com>.
[4] ParkCloud. Available from: <https://www.parkcloud.com>.
[5] ParkCloud. Available from: <https://www.currencycloud.com>.
[6] ParkCloud. Available from: <https://www.currencycloud.com>.
[7] ParkCloud. Available from: <https://www.currencycloud.com>.
[8] ParkCloud. Available from: <https://www.currencycloud.com>.
[9] ParkCloud. Available from: <https://www.currencycloud.com>.
[10] ParkCloud. Available from: <https://www.currencycloud.com>.
[11] ParkCloud. Available from: <https://www.currencycloud.com>.
[12] ParkCloud. Available from: <https://www.currencycloud.com>.
[13] ParkCloud. Available from: <https://www.currencycloud.com>.
[14] ParkCloud. Available from: <https://www.currencycloud.com>.
[15] ParkCloud. Available from: <https://www.currencycloud.com>.
[16] ParkCloud. Available from: <https://www.currencycloud.com>.
[17] ParkCloud. Available from: <https://www.currencycloud.com>.
[18] ParkCloud. Available from: <https://www.currencycloud.com>.
[19] ParkCloud. Available from: <https://www.currencycloud.com>.
[20] ParkCloud. Available from: <https://www.currencycloud.com>.
[21] ParkCloud. Available from: <https://www.currencycloud.com>.
[22] ParkCloud. Available from: <https://www.currencycloud.com>.
[23] ParkCloud. Available from: <https://www.currencycloud.com>.
[24] ParkCloud. Available from: <https://www.currencycloud.com>.
[25] ParkCloud. Available from: <https://www.currencycloud.com>.
[26] ParkCloud. Available from: <https://www.currencycloud.com>.
[27] ParkCloud. Available from: <https://www.currencycloud.com>.
[28] ParkCloud. Available from: <https://www.currencycloud.com>.
[29] ParkCloud. Available from: <https://www.currencycloud.com>.
[30] ParkCloud. Available from: <https://www.currencycloud.com>.
[31] ParkCloud. Available from: <https://www.currencycloud.com>.
[32] ParkCloud. Available from: <https://www.currencycloud.com>.
[33] ParkCloud. Available from: <https://www.currencycloud.com>.
[34] ParkCloud. Available from: <https://www.currencycloud.com>.
[35] ParkCloud. Available from: <https://www.currencycloud.com>.
[36] ParkCloud. Available from: <https://www.currencycloud.com>.
[37] ParkCloud. Available from: <https://www.currencycloud.com>.
[38] ParkCloud. Available from: <https://www.currencycloud.com>.
[39] ParkCloud. Available from: <https://www.currencycloud.com>.
[40] ParkCloud. Available from: <https://www.currencycloud.com>.

**Fig. 6.** Average matching score based on threshold.

**Fig. 7.** Execution time based on matching threshold.

**Fig. 8.** Recall values for each query.
Sim KM. Agent-based cloud computing. IEEE Trans Serv Comput 2012;5(4):564–77.

Tahamtan A, Beheshti SA, Anjomshoaa A, Tjoa AM. A cloud repository and discovery framework based on a unified business and cloud service ontology. IEEE Eighth World Congress 2012:203–10.

Affy Y, Moawad I, Badr N, Tolba M. A semantic-based software-as-a-service (saas) discovery and selection system. In: 8th International conference on IEEE computer engineering and systems (IC-CES) 2013; 2013. p. 57–63.

Hasan N, Shu G, Malekzadeh A, Zi-Long J. An efficient approach based on hierarchical ontology for service discovery in cloud computing. TELKOMNIKA Indonesian J Electr Eng 2014;12(4):2995–13.

Zhang J, He L, Huang F, Liud B. Service discovery architecture applied in cloud computing environments. Appl Mech Mater 2013;241–244:3177–83.

Somu N, Kirthivasan K, Sriram V. Computational model for ranking cloud service Providers using hypergraph based techniques. Future Gen Comput Syst 2017;68:14–30.

Alain B, Cherifi H, Aboutajdine D. Hypergraph imaging: an overview. Pattern Recogn 2002;35(3):651–8.

Abourezq M, IdriSSI A. A cloud services research and selection system. Int Conf Multimedia Comput Syst 2014:1195–9.

Borzsonyi S, Kossmann D, Stocker D. The Skyline operator. Int Conf Data Eng (ICDE) 2001:147–54.

Garg SK, Versteeg S, Buyya R. SMICloud a framework for comparing and ranking cloud services. Fourth IEEE Int Conf Utility Cloud Comput 2011:210–8.

Jia-jing H, Jin-dong W, Nan W, Kan N. Service discovery method based on two-step clustering. In: 4th International conference on computer science and network technology (ICCSNT) 2015. p. 220–4.

Barati M, St-Denis R. An architecture for semantic service discovery and realizability in cloud computing. In: 2015 6th International conference on the network of the Future (NOF) 2015; 2015. p. 1–6.

Kang J, Sim K. Towards agents and ontology for cloud service discovery. Int Conf Cyber-Enabled Distrib Comput Knowl Discov 2011:483–90. doi:http://dx.doi.org/10.1109/CyberC.2011.84.

Lu Y, Xu X. A semantic web based framework for service composition in a cloud manufacturing environment. J Manuf Syst 2017;42:69–81.

Rekik M, Boukadi K, Ben-Abdallah H. An end-to-end framework for context-aware business process outsourcing to the cloud. Comput Electr Eng 2017;63:308–19.

Noor TH, Sheng QZ, Alzazi A, Ngu AH, Law J. Csc e crawler engine for cloud services discovery on the world wide web. In: Web Services (ICWS) IEEE 20th international conference 2013; 2013. p. 443–50.

Amazon. Available from: <https://aws.amazon.com/workspaces/>.

Deasktop as a service: Available from: <http://www.desktopasservice.com/>.

Gracia J, Mena E. Semantic heterogeneity issues on the web. IEEE Internet Comput 2012;16(5):60–7.

Cloud Computing Reference Architecture (ISO/IEC 17789:2014). Available from: <https://www.iso.org/standard/60545.html>.

IBM Cloud Computing Reference Architecture 4.0. Available from: <https://www.ibm.com/developerworks/community/groups/community/ibm-cloud-ra/>.

The Open Cloud Manifesto. Available from: <https://gevaperry.typepad.com/Open%20Cloud%20Manifesto%20v1.0.5.pdf>.

Bulskov H, Knappe R, Andreasen T. “On measuring similarity for conceptual querying. In: Proceedings of the 5th international conference on flexible query answering systems 2002;2002. p. 100–11.

Sánchez D, Batet M, Isern D. Ontology-based information content computation. Know – Based Syst 2011;24(2):297–303.

Andreasen T, Bulskov H, Knappe R. From ontology over similarity to query evaluation. In: Bernardi R, Moortgat M, editors. 2nd CoLogNET-ElsNET symposium – questions and answers: theoretical and applied perspectives. Elsevier Science 2003;2003. p. 39–50.

Raghavan V, Bollmann P, Jung G. A critical investigation of recall and precision as measures of retrieval system performance. ACM Trans Inform Syst (TOIS) 1989;7(3):205–29.