An Efficient Multi-Cloud Service Composition Using a Distributed Multiagent-Based, Memory-Driven Approach

Philip Kendrick, Thar Baker, *Member, IEEE*, Zakaria Maamar, Abir Hussain, *Member, IEEE*, Rajkummar Buyya, *Fellow, IEEE*, and Dhiya Al-Jumeily, *Senior Member, IEEE*

Abstract—Cloud services are often distributed across several data centers requiring new scalable approaches to efficiently perform searching to reduce the energy and price cost of fulfilling requests. Multiagent-based systems have arisen as a powerful technique for improving distributed processing on a wide scale, which can operate in environments where partial observability is the norm and the cost of prolonged search can be exponential. In this paper, we present a multiagent-based service composition approach, using agent-matchmakers and agent-representatives, for the efficient retrieval of distributed services and propagation of information within the agent network to reduce the amount of brute-force search. Our extensive simulation results indicate that by introducing localized agent-based memory searches, the amount of actions (with their associated energy costs) can be reduced by over 50 percent which results in a lower energy cost per composition request.

Index Terms—Cloud data centers, energy efficiency, service composition, memory-driven solution, multiagent simulation

1 Introduction

In a short period of time, the use of cloud computing to benefit from features like *elastic* and *pay-per-use* resources (whether software, platform, or infrastructure) has dramatically increased. For instance, this use has generated £35 billion revenues in Europe by end of 2014 [1]. Elasticity permits to scale up/down resources according to changing users' (more/less) demands. Pay-per-use allows organizations to cut down operation costs by using resources whenever there is a need (like car rental). Along with these two features, cloud advocates regularly convey the message that cloud resources from one particular provider are sufficient for satisfying a user's demands. Unfortunately, this is not always the case; first, users do not like to be locked into one particular cloud provider;[1] and second, users' demands are more and more complex requiring the collaboration of several independent cloud providers [3], [4].

Also, Buyya et al. [5], [6] discuss the difficulty that the cloud application service Software-as-a-Servsice (SaaS) providers encounter to meet the Quality of Service (QoS) for all their customers, due to the fact that no single cloud provider is able to establish their data centers at all possible locations across the whole world. Hence, the use of services of multiple cloud service providers is deemed necessary as, together, they can provide better support for their specific consumer needs.

There is a consensus in the ICT community that any open environment like the Internet, requires a central authority that would, among other things, oversee all operations and guarantee fairness to all contributing parties. In our proposed multi-cloud environment, we refer to the central authority as matchmaker whose main role is to bridge the gap between cloud users and cloud providers despite their conflicting interests. Indeed, cloud users aim at minimizing expenditures along with securing high-quality services;[2] and, cloud providers aim at maximizing revenues along with consuming less energy that would be due to data processing, storage, and transfer between facilities (*aka* data centers) hosting cloud resources. Being energy efficient (*aka* green), in compliance with different regulations, has become of a paramount importance to all cloud stakeholders. The 2011 report of PBL Netherlands Environmental Assessment Agency and JRC European Commission [7] insist on reducing energy consumption in order to decrease CO_2 emission volume by 15-30 percent before 2020 [8]. In this paper, we examine how to achieve an energy efficient multi-cloud service collaboration. We advocate for softwareagents as potential

1. According to a Logicworks survey by Wakefield Research, "78 percent of IT decision makers believe that concerns about vendor lock-in preventing their organisation from maximising the benefits of cloud resources. This means that the majority of IT leaders consciously choose not to fully invest in cloud, because they value long-term vendor flexibility over long-term cloud success" [2].

2. Services wrap resources in compliance with Software-as-a-Service, Platform-as-a-Service, and Infrastructure-as-a-Service.

• P. Kendrick, T. Baker, A. Hussain, and D. Al-Jumeily are with the Faculty of Engineering and Technology, Department of Computer Science, Liverpool John Moores University, Liverpool L3 5UA, United Kingdom. E-mail: p.g.kendrick@2012.ljmu.ac.uk, {t.baker, a.hussain, d.aljumeily}@ljmu.ac.uk.
• Z. Maamar is with the College of Technological Innovation, Zayed University, Dubai 19282, UAE. E-mail: zakaria.maamar@zu.ac.ae.
• R. Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, University of Melbourne, Parkville, VIC 3010, Australia. E-mail: rbuyya@unimelb.edu.au.

Manuscript received 31 Dec. 2017; revised 2 Aug. 2018; accepted 12 Nov. 2018. Date of publication 18 Nov. 2018; date of current version 8 Sept. 2021.
(Corresponding author: Thar Baker.)
Recommended for acceptance by W. Shi and C. Jiang.
Digital Object Identifier no. 10.1109/TSUSC.2018.2881416

candidates for running this collaboration. They will (i) act on behalf of all stakeholders so their anonymity is maintained, (ii) be proactive when they respond to certain events like sudden increase in energy consumption, (iii) coordinate their activities with other peers, and (iv) memorize past behaviours and act accordingly.

A quick literature review reveals that existing service composition practices over the clouds overlook the energy aspect. This management is directly dependent on the locations and size of cloud data centers [9] along with the volume of data that needs exchange increasing network traffic. Some statistics indicate that cloud use increased from $16 billion in 2008 to $42 billion in 2012, and more rapidly thereafter [10]. This rapid growth in services over clouds (referred to as cloudservices in the rest of this paper) has generated £35 billion revenues just in Europe by end of 2014 [1]. This paper presents and evaluates an energy-conscious, distributed multi-agent based approach for composing cloud services. Agents are potential candidates for tackling the challenges of this composition. First, agents would act on behalf of composition's stakeholders by ensuring their anonymity. Second, agents would be proactive by taking preventive actions in response to certain events like sudden increase in energy consumption.

Our contributions are manifold including: (i) a novel multi-agent approach to performing Web services composition, (ii) a relaxable approach to fulfilling compositions based on the user's requirements of either an energy efficient or cost efficient search, and (iii) to reduce the number of energy footprint of performing Web service compositions by up to 50 percent through our agent-based memory-driven approach. Our agents can communicate with each other providing a memory-driven approach in which each agent is aware about its surrounding and the energy activities with the data centers.

The rest of the paper is organized as follows. Section 2 discusses related works on agents and service computing in brief. Section 3 presents the problem addressed along with the formal definition of service composition problem. Section 4 explains the model of processing Web service composition problem using intermediate agents, and introduces the proposed algorithms and a case study to illustrate the multiagent-based approach. Section 5 describes the implemented simulator, followed by the testing and evaluation of energy efficiency and price efficiency in Section 6. Finally, Section 7 concludes the paper and draws up some future work.

2 Agents and Services: A Brief Overview

Blending agent computing with service computing (with focus on Web services) has been around for many years as per the large number of scientific events that took place (e.g., ESAS2014,[3] ICWS2014,[4] IEEE/WIC/ACM ICWI,[5] IEEE/WIC/ACM'WIIAT,[6] and ICEBE2017[7]), some self-organized international venues such as Agent-Based Service Oriented Computing,[8] and Extent Web services technologies: the use of multi-agent approaches,[9] and some other references [11], [12], [13], [14]. Agents help address different issues such as how to agentify Web services, how to inject semantics into Web services, how to build robust Web services, how to develop communities of Web services, just to cite some. However, to the best of our knowledge, there are not serious efforts into addressing energy consumption in a multiple cloud-based service composition environment. Although competition is always healthy, multiple clouds would require criteria to select the best scheduling for optimized involvement, to address their (semantic) conflicts, to ensure their coordination, etc. All these criteria will have an impact on energy consumption.

Gutierrez-Garcia et al. [15], [16] propose a Multi-Agent System (MAS) for service composition in cloud computing. This composition is augmented in two orientation horizontal composition where integration of heterogeneous services (e.g., storage and compute) that satisfy a user request are scattered across several clouds; or vertical composition where homogenous services/resources are put together to expand the capacity of a given cloud node rather than satisfying an external request (e.g., augmenting storage capacity by adding new storage data centers [17]).

Parhi et al. [18], [19] use software reputation agents to analyze the popularity of Web services and rank them accordingly using user feedback and statistical information. In this case, the behavior of individual users is tracked and analyzed with focus on Web services' QoS properties. The model aims at reducing the amount of search for a service composition over the network of multiple clouds using a number of specialized agents.

Cloud service negotiation mechanisms and strategies, to establish Service Level Agreements (SLAs) among the cloud stakeholders (i.e., consumers, brokers, and providers) for service composition are discussed in [20], [21], [22], [23] using MASs. However, the proposed multiagent-based negotiation mechanisms do not allow the clients/consumers to break the contract, once set, if the service does not satisfy the consumer needs. Contrarily, a multiagent-based cloud commerce model [24], [25] devises a complex negotiation mechanism, along with its "parallel" negotiation activities among the cloud stakeholders in interrelated cloud service markets, that is breach-able by the consumers after paying a certain penalty fee.

Cloudle [26], [27], [28] proposes a new architecture for cloud service composition consisting of a discovery agent, a cloud ontology, a cloud services database, and multi-crawlers for cloud. Cloudle allows multi-crawlers/agents to update the cloud services database, also build a new one in certain cases (e.g., none of the pre-defined services in the database satisfy the request), with the new services composition after scanning/surveying all available services. If none of the available services serves the requested composition, the multi-crawlers traverse the Web components and extract relevant services, in which case it will need to build a new database for those services, which is deemed a time-consuming process.

3. https://www.computer.org/web/compsac/2014/esas
4. https://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=33102
5. https://grid.cs.gsu.edu/wic2013/wi
6. https://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=34076
7. http://conferences.computer.org/icebe/
8. http://www.springer.com/gp/book/9781849960403
9. http://www.springer.com/gp/book/9780387233437

To sum up, there are many references on Web services and software agents; however, to the best of our knowledge, none has addressed the energy aspect of service composition on the clouds.

3 Problem Statement

In a traditional cloud service request scenario, a user sends a request to a service provider directly, or via a matchmaker, stating the specifications of the requested service(s). The provider should then find the appropriate service(s) that satisfies the order from a set of available services. This scenario becomes more complicated as the number of requested cloud services increase alike. Currently, locating the best-fit service that matches the user needs and the matchmaker's aims is considered to be the most challenging task in multi-cloud environment due to many reasons ranging from (i) users' requirements (e.g., high performance service with less payment), (ii) providers' requirements (e.g., more income with less expenditure) to (iii) environmental requirements (e.g., less energy consumption and carbon emission). The current state-of-the-art solutions focus primarily on users' requirements and providers' requirements, as detailed in Section 2, whereas the primary aim of this paper is to propose and evaluate a high-end energy efficient service composition approach to address the overall amount of energy required by the appropriate composite services. In addition, Web Service Composition Problems (WSCP) over the clouds are often treated as classical search problems [29] with little attention given to the overhead of communicating over a network to perform service searching (refer to Definition 1).

In this paper, we identify the main actions that contribute to the overall energy cost of addressing WSCP: (1) sending and receiving information over a network, (2) brute-force searching cloud data center for matching Web services, and (3) cataloguing information about Web service locations in a central repository. We propose a distributed multiagent approach to addressing WSCP by reducing the amount of information sent over the network, using agent-based memory-driven approach to reduce the amount of service-ocation re-processing that must occur overtime and distributing the knowledge of services across several agents.

Definition 1. *The service composition problem over the clouds is defined as finding a subset of services that can fulfil some request:*

- *Let $q \in Q$ represents the composition request defined as a tuple $\langle u, \{i\}, \{o\}, \{p\} \rangle$ where u is the user's identity, i is a set of input information to be processed, o is a set of expected output information, and p is a set of user-preferences/restrictions governing how the data should be handled (e.g., users may specify that the data should not be processed out of some specified region).*
- *Let S represents the set of services in a cloud data center (s_C).*
- *The composition problem is to find a subset of services located across multiple cloud data centers to fulfil the request, such that: $\forall q \in Q | \exists s \in s_C$.*

4 Model Definition

The proposed model facilitates the processing of WSCP requests by using two intermediate agents (Definition 2): (1) the matchmaker that works on behalf of the user to process requests, and (2) the cloud representative that works on behalf of the cloud data center to make meta-information available to the matchmaker (e.g., cost of services, energy efficiency and availability). Together, the matchmaker and representatives fulfil composite cloud requests that may only be fulfilled by finding services located in different cloud data centers (refer to Fig. 1).

Proposals. The concept of a request proposal is introduced as a container for information that is passed between the matchmaker and representatives using a memory-driven approach. The output of the agent-representatives search of the cloud data center for suitable services is for zero-or-more proposals containing groups of services that can be used to transform the input data in some way. Incomplete proposals (i.e., a group of services that only partially transform the information) may be made complete by combining proposals or services located in other cloud data centers.

Regions. Regions are included within the model and the computational cost of interacting with entities far away. Any agent-matchmaker or agent-representative and cloud data centers that are based within the same area (defined manually or automatically based on relative network latency) are grouped as being within the same region and prioritized during the service composition search. Under circumstances where a user request cannot be complete within a particular region, we define a set of functions that allow matchmakers to transfer the user request and representative proposals to matchmakers in other regions for completion. To facilitate this, we assume the existence of an agent repository storing information about agent-matchmaker locations so that they may be contacted. Following the transfer of the request or proposal, the process of searching for services to fulfil the request is functionally similar to that previously described in Algorithms 1 and 2 (shown in Section 4.1 below). The purpose of geographical boundaries is to allow agents to build a historical database H of data centers that it can work with to encourage local optimization and to reduce the volume of requests that need to be outsourced to previously unused data centers.

Routes. In addition to considering the global distance between cloud data centers, agent-matchmakers and agent-representatives also consider the network routes available between each other. We assume the existence of many network routes available between users and matchmakers, matchmakers and representatives and representatives and cloud data centers. As agents make communications to other entities they also monitor the efficiency of the routes used to locally prioritize faster routes and responsive agents. Furthermore, we consider that routes are dynamic and so propose an update model that allows agents to periodically outside of normal operations traverse and measure the effectiveness of the route in terms of latency. In addition, as agents will be communicating and requesting information stored locally or available through other agents, we consider our solution a memory-driven approach.

4.1 Proposed Algorithms

Algorithms 1 and 2 describe the functions of the agent-matchmaker and agent-representative, respectively.

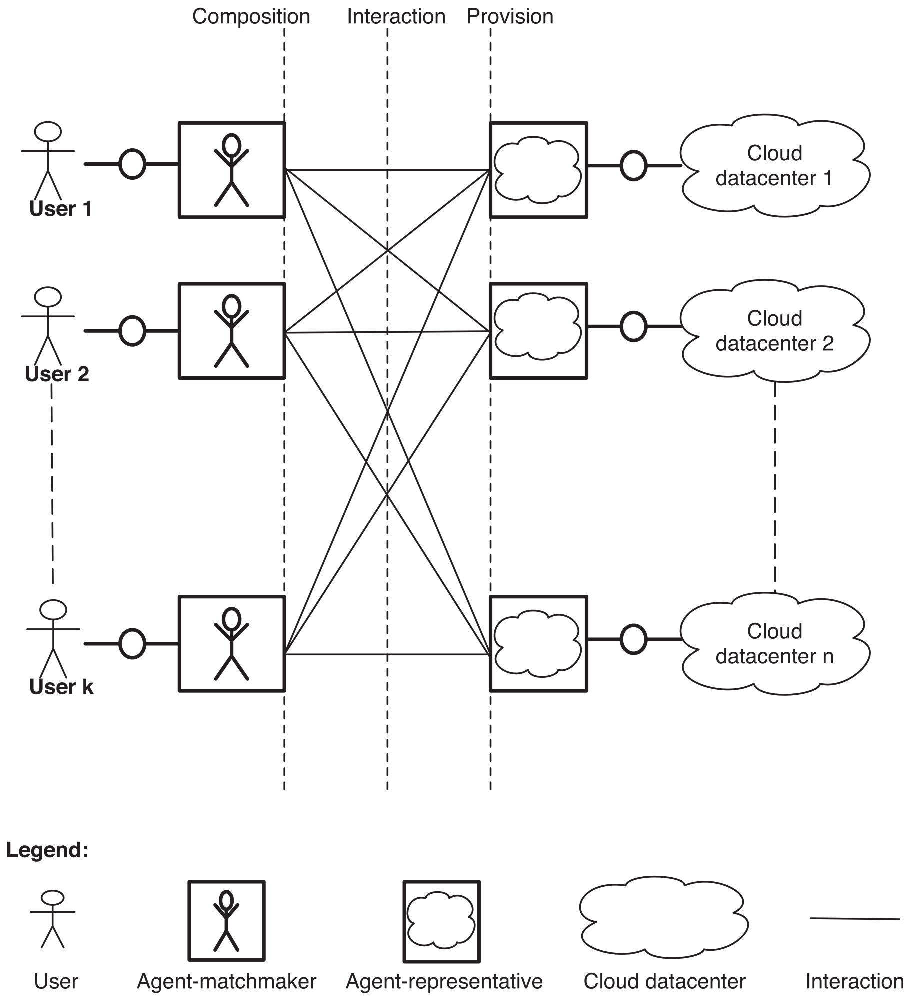

Fig. 1. High-level overview of the agent-matchmaker and agent-representatives interactions for finding services located in different cloud data centers using previously identified information and remembered in the agents memory.

Definition 2. *The proposed agent model is composed of four main entities to facilitate the handling and searching of service subsets to fulfil a composite cloud request.*

- *Let U represents the set of users that make requests.*
- *Let M represents the set of agent-matchmakers that receive the initial request from a user $u \in U$. Note that several users may use the same agent-matchmaker.*
- *Let C represents the set of cloud data centers containing services.*
- *Let R represents the set of agent-representatives belonging to an individual cloud data center (C_r) that process requests from a matchmaker ($m \in M$).*

Algorithm 1 lists the agent-matchmaker functions whose role is to interface with the user and cloud representatives. Matchmakers are geographically fixed agents within a region that receive requests $\langle u, \{i\}, \{o\}, \{p\} \rangle$ from users where u is the users identity, $\{i\}$ is a set of input information to be processed, $\{o\}$ is a set of expected output information and $\{p\}$ is a set of user preferences for how $\{i\}$ should be processed. Following the submission of a new request, the matchmaker first checks whether a "similar" request has been processed in the past by checking whether it exists in a log of past composition requests. H containing information about the location of services that can transform $\{i\}$ to $\{o\}$. The current events are dynamically appended to the historical log H after each successful composition request has been fulfilled to increase the speed at which services are located over time. If the request cannot be fulfilled by knowledge from H, a search of data centers within the agent's local region begins. Several pieces of information are combined to decide the order in which data centers are searched for composite services. First, a geographical region is defined for each matchmaker containing all of the available data centers and representative agents that can be contacted. For data centers within this region, two pieces of information are used:

1) $info_{dc}$ containing meta-data about the data center's services (e.g., service cost and energy efficiency). This information is periodically sent from the representative agent to the matchmaker during off-peak times.
2) $info_{route}$ contains information (e.g., congestion and latency) about possible physical routes that can be taken between the matchmaker and the representative agent. Using this information in conjunction with any user preferences (e.g., to prioritize service speed over cost), the matchmaker decides the order in which data center representatives are contacted.

Algorithm 1. Agent Matchmaker Functions

Require:
$\langle u, \{i\}, \{o\}, \{p\}\rangle \in q$ a request from user u where $\{i\}$ is a set of input information, $\{o\}$ is a set of expected output information and $\{p\}$ is a set of user preferences for how the request should be processed.
$C_{local} \leftarrow C_{local} \subseteq C$ a set of cloud data centers in the agents region (i.e., state, country or continent).
Define:
H is a database of past composition requests and how they were fulfilled.
$info_{route} \leftarrow \emptyset$ ordered information about the possible routes and their efficiency to the agent representatives (R) and data centers (C_{local}).
$info_{dc} \leftarrow \emptyset$ ordered information about the data centers (e.g., energy efficiency, average cost).
$info_{state} \leftarrow \emptyset$ ▷ Information obtained from agent representatives
$resp \leftarrow \emptyset$ the service composition response from $SearchDatacenter(\{i\}, \{o\})$.

for $\langle u, \{i\}, \{o\}, \{p\}\rangle \in q_u$ **do**
  if $q \subseteq H$ **then** ▷ If components of the request have been processed in the past, use that knowledge to directly contact the correct cloud data center.
    PollRepresentatives($\{i\}, \{o\}, H$)
  else
    PollRepresentatives($\{i\}, \{o\}, \emptyset$)
  end if
end for
procedure PollRepresentatives$\{i\}, \{o\}, \{p\}, H$ ▷ Contact a representative to search for services matching the input and output information.
  for $\langle \{i\}, \{o\}\rangle \in q$ **do** ▷ Begin recursive search of data centers for matching and missing services.
    $info_{state} \leftarrow C.ShareState$
    if p = cost saving **then** ▷ Prioritise searching cheaper data centers.
      for $cost \in info_{dc}$ **do**
        $resp \leftarrow SearchDatacenter(\{i\}, \{o\})$ ▷ Will return either a complete service composition, partial service composition or null.
      end for
    else if p = efficiency **then** ▷ Prioritise searching faster closer data centers.
      for $route \in info_{route}$ **do**
        $resp \leftarrow SearchDatacenter(\{i\}, \{o\})$
      end for
    else if p = region specific **then** ▷ Search for services in a specific region.
      for $\{C \mid C = region\}$ **do**
        $resp \leftarrow SearchDatacenter(\{i\}, \{o\})$
      end for
    else if p = offpeak **then** ▷ Search for services that are currently within the offpeak time.
      if $\{C \mid C.offpeak \in info_{state}\}$ **then**
        $resp \leftarrow SearchDatacenter(\{i\}, \{o\})$
      end if
    end if
  end for ▷ End once service composition is complete or searching is exhausted.
  $H \leftarrow resp$ ▷ Update the log with the service location.
  return $resp$ ▷ Return response to user.
end procedure

Algorithm 2. Agent Representative Functions

Require:
$\langle m, \{i\}, \{o\}\rangle \in q$ a request from matchmaker m where $\{i\}$ is a set of input information and $\{o\}$ is a set of expected output information.
Define:
C is a set of cloud data centers.
$c' \in C$ the identity of the cloud data center the representative is assigned to.
S the set of services in c'.
$H \leftarrow \emptyset$ a log of past composition requests and how they were fulfilled.
$C_{local} \leftarrow C \subseteq C$ ▷ The subset of cloud data centers in the agents local region (i.e., state, country or continent).
$info_{route} \leftarrow \emptyset$ ordered information about the possible routes and their efficiency to the agent-matchmaker (m) and data centers (C_{local}).
$info_{dc} \leftarrow \emptyset$ ordered information about the data centers (e.g., energy efficiency and cost).

procedure SearchDatacenter$\{i\}, \{o\}$ ▷ Process incoming requests from matchmakers to search the data center for matching services.
  for $s \in S$ **do** ▷ Recursively find services with matching input or outputs.
    if $s_{c'} \in S = \{i\} \vee s_{c'} \in S = \{o\}$ **then**
      $S' \leftarrow s_{c'}$ ▷ Add any services that can work with the requested information to S'.
    end if
  end for
  if $\exists(\{i\} \in S' \wedge \{o\} \in S')$ **then** ▷ Return the services if the request can be completed in full.
    return S′
  else if $info_{route} >= route \in info_{dc}$ **then** ▷ If it's faster to contact other representatives directly than send data to the matchmaker.
    SearchDatacenter($\{i\} \in S', \{o\} \in S'$) ▷ Find missing the missing services.
  else
    return S′ ▷ Return the incomplete request to the matchmaker.
  end if
end procedure
procedure ShareState() ▷ Process incoming requests from matchmakers to share information.
  $info_{state} \leftarrow \emptyset$ ▷ Empty set of information to share with the matchmaker
  $\delta \leftarrow$ current time
  $offpeak \leftarrow$ current offpeak time range
  if $\delta \in offpeak$ **then**
    $info_{state} \leftarrow \langle isOffpeak, true\rangle$
  end if
  return $info_{state}$
end procedure

Algorithm 2 lists the cloud representative functions whose role is to search for the data center that it has been assigned to. Upon receipt of a request from a matchmaker $\langle m, \{i\}, \{o\}\rangle$ where m is the matchmaker's identity, $\{i\}$ is a set of input information to be processed, $\{o\}$ is a set of expected output information, the representative performs a recursive search of the data center by finding any services that match $\{i\}$ or $\{o\}$ as the respective inputs or outputs. In the case where a

TABLE 1
WSDL Services Consisting of Input and Output Information (Abstracted Using a Unique 11-Digit Code) Where Input Data is Required to Run the Service and Output Data is Produced As a Result of Running the Service

Service No.	Input Data	Ouput Data	Location
141	[P79a7296189, P37a4226984]	[P93a0686486, P11a9459124, P22a4008387]	data center 2
16	[P11a9459124, P93a0686486, P22a4008387]	[P62a7398547, P90a6939861, P71a7297795]	data center 2
100	[P22a4008387, P11a9459124]	[P90a6939861]	data center 2
76	[P37a4226984, P79a7296189]	[P11a9459124, P22a4008387]	data center 4
96	[P37a4226984, P79a7296189]	[P22a4008387, P11a9459124]	data center 4

request can be fulfilled by a single service, the information is processed and returned to the matchmaker. However, if the request is composite, the representative will recursively search for services that can be linked together to transform the input data $\{i\}$ into the output $\{o\}$. The outcome of the process is to either find a subset of services that can fulfil the request or produce proposals that can transform the input data in some way, but not fully produce the output. Partially complete proposals that are returned to the matchmaker may be fulfilled by performing the same searching process at different data centers. Under circumstances where the matchmaker has the option to choose from several proposals that can fulfil the user request, the user preference and meta-data about the cost and energy efficiency are used to decide which service composition should be used.

4.2 Case Study

To illustrate the agent-based model, a simulation using WSDL-defined services (Table 1) from the ICEBE05 dataset [30] was performed. ICEBE05 Web service datasets are originally auto-generated from software by the ICEBE05 organisation. It has been publicly available by Web service research community to solicit algorithms and software to discover pertinent Web services and compose them to make value-added functionality. Within the dataset composition services

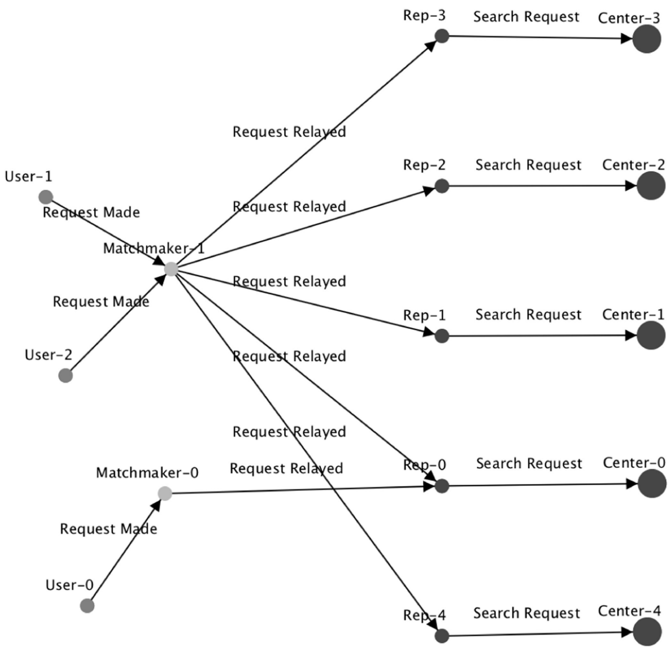

Fig. 2. The simulated cloud environment containing users, agent-matchmakers, agent-representatives, and cloud data centers.

are represented by the input and output data used to transform the input information into the output result.

The type of data is abstracted using unique 11-digit codes (e.g., [P37a4226984]) representing that information (refer to Table 1). Fig. 2 shows a snapshot of the simulated environment with three users (namely: User-0, User-1, and User-2) making composition requests, the process of which is described as follows: the simulation begins with a user (User-1) submitting a request to transform the input data [P37a4226984, P79a7296189] to the output data [P90a6939861]. This request is sent to an agent matchmaker (Matchmaker-1) which, using previously described performance metrics, selects a cloud representative to search first (Rep-3). The cloud representative, which is responsible for searching the cloud data center (Center-3), either identifies services which can be used to fulfil the request or communicates that no services satisfying the users requirements. In this example, three services, 16, 100, and 141 (refer to Table 1), were identified as being able to contribute to the user's request. While the identified services cannot complete the user's request directly, they can be used to partially transform the data and with the use of additional services, complete the request. Services 141 and 16 were returned as a partial proposal and service 100 was returned as a separate proposal. Details of the three services are then sent back to the agent matchmaker (Matchmaker-1) who contacts other representatives (Rep-1 and Rep-2) to search for composite services that can transform the data further. Services 76 and 96, found at a different data center (Rep-2), where found to be able to be combined with service 100 to produce the required data transformation and were both returned as proposals to the matchmaker. As new proposals are submitted to the matchmaker, it can select the best possible composition to fulfil the users requests based on their requirements. The matchmaker can, therefore, select the most efficient service to be used to process the data.

5 Simulation Environment

In this section, a discussion about the implementation of the proposed simulator, including how the model entities (e.g., agents and cloud data centers) are simulated, as well as an explanation of the underlying variables is provided. A new simulator was required to fully take advantage of the multiagent-based architecture and to allow measurements of individual actions taken by each agent. The simulated environment holds three dynamic entities: users, agent-matchmakers and agent-representatives. The cloud data center is treated as a static entity that may be queried by the agent representative to find corresponding Web services that match a request. Cloud services are extracted from the ICEBE05 dataset [30] and distributed randomly between the available

TABLE 2
Agent and Environment Variables

Variable	Value
No. Routes	3
No. Regions	3
No. Data centers	5
No. Users	3
No. Users Per Matchmaker	2
Request Size	5

cloud data centers. Cloud requests are made by the user to the agent-matchmaker for fulfilment.

Within the simulated environment, a number of variables control the number of entities and complexity of the WSCP. Table 2 lists several parameters, of which, *No. Routes* and *No. Regions* controls the operating landscape by defining the number of possible routes between the user and agent-matchmaker as well as the number of routes between the agent-matchmaker and agent-representative. Each cloud data center belongs to a particular region which simulates the geographical distance between clusters of cloud data centers. The variables *No. Users* and *No. Users Per Matchmaker* simulate the number of users that can make requests and the number of users that are assigned to a particular matchmaker. In this paper, we focus on one user that submits requests to evaluate the effectiveness of the architecture and leave discussion of request parallelism for future work. Finally, the variable *Request Size* controls the number of services required to transform the request input into the desired output.

The aforementioned simulated environment can be viewed graphically as a set of nodes representing the entities and edges representing the propagation of information and user requests (Fig. 2 showing the interaction of three users, two matchmaking agents and five representatives).

6 Performance Evaluation

To evaluate the proposed model the ICEBE05 dataset [30] was used to search for solutions to randomly generated WSCPs. Composition services were randomly generated by finding series of Web services that when ordered, would transform some initial input data into the requested output data. The complexity of the WSCPs are controlled by the number of services available that can fulfil the request as well as the length of the problem.

The baseline algorithm for searching services to fulfil the WSCP uses an exhaustive search of the environment, and is computationally similar to algorithms that do not make use of the proposed additional features. The baseline algorithm performs a randomised search of the available cloud data centers for the desired composition of services and represents how search would be performed without the memory of past composition requests and information about the likely location of composition parts. The evaluation metric is *Number of Actions*, which is a counter of actions performed such as sending and receiving requests and searching cloud data centers for matching services. Of the available actions that increment this metric, searching the cloud data center for matching services increases the value the most as it corresponds to the computation cost of iteratively searching for information. In this way, exhaustively searching cloud data centers for matching services is costly and is avoided in the improved algorithm. The improved algorithm uses agent memory of past searches to allow agents to find whole or partial services compositions without having to exhaustively search a cloud data center. The reduction in the number of actions needed to fulfil a request results in a less computationally expensive solution to the WSCP.

For the experimental setup, 5 iterations of 20 WSCPs were created by the user and tested under the baseline exhaustive algorithm and the improved memory-driven approach. For each of the 5 iterations, a new randomly generated environment was created from the ICEBE05 service files [30] and simulator parameters (Table 2). The number of actions used to fulfil the 20 requests are shown in Fig. 3. The number of actions needed to fulfil the request is largely governed by the complexity of the randomly generated WSCP, however, for each of the tested cases, the proposed improved algorithm outperformed the baseline. The degree of improvement made over the baseline is governed by the relevance of previously processed requests, such that, if the previously fulfilled requests contain services that can be used for the current WSCP, then the improvement is greater as the amount of exhaustive searching required is reduced. Additionally, Fig. 4 shows the results of 20,000 users

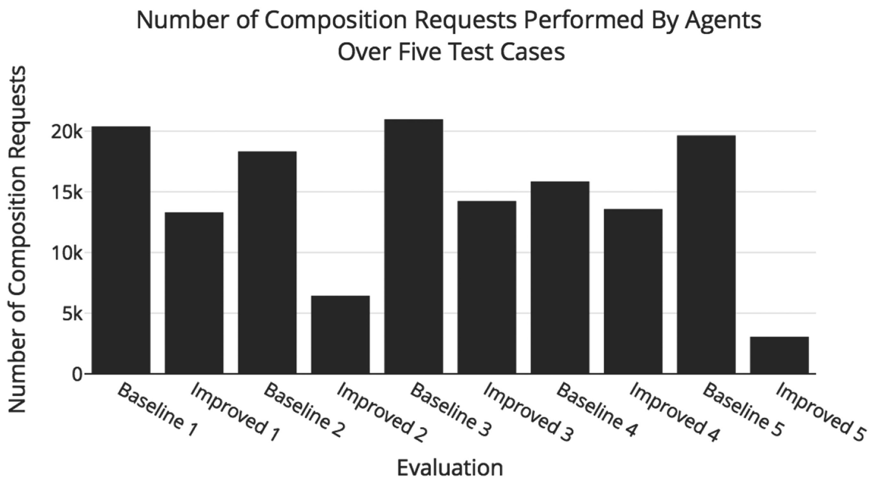

Fig. 3. Five simulation evaluations showing the baseline exhaustive search and improved agent algorithms.

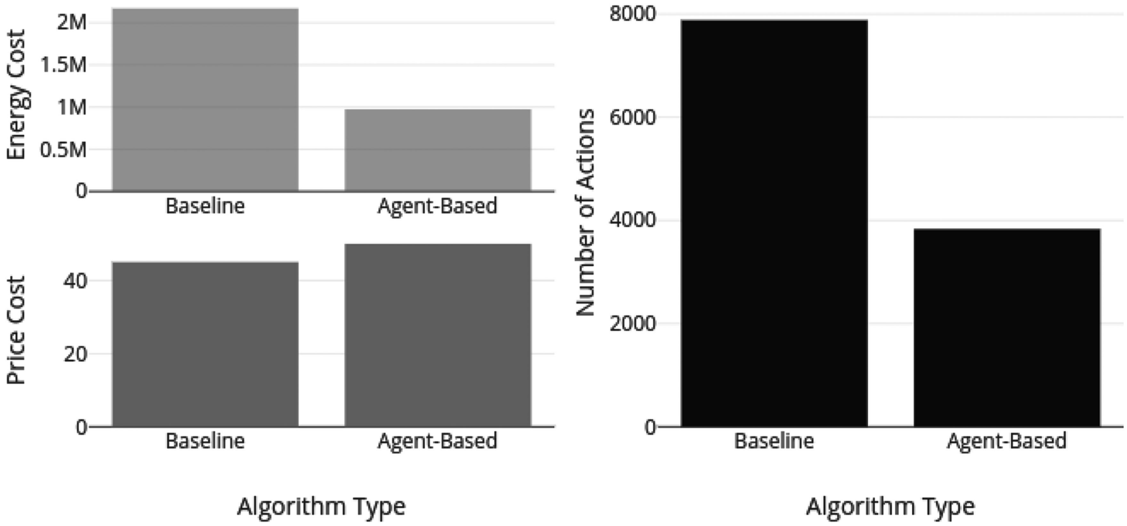

Fig. 4. An evaluation of 20,000 users interacting with the baseline and proposed agent-based system.

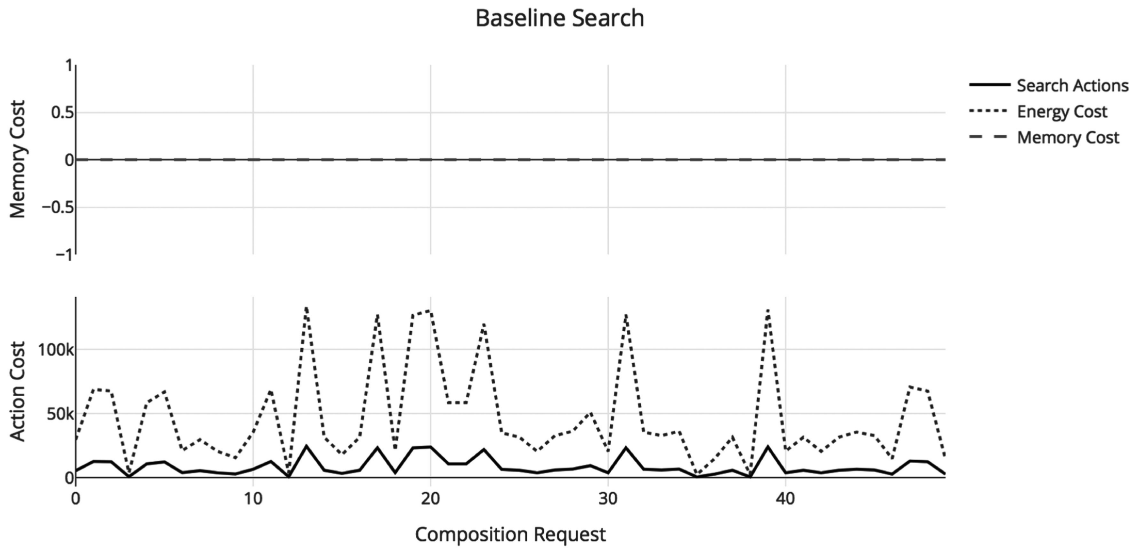

Fig. 5. Memory cost and action cost of the baseline search algorithm which uses no memory functions.

interacting with both the baseline and agent-based systems (using 50 agent matchmakers and 30 data centers). The results show a decrease in the average number of actions and a decrease in the overall energy cost resulting from the use of the agent approach. The trade-off in the system is an increase price cost due to the use of preferential web services to reduce the energy cost.

The benefit of using agent-based memory to distribute the burden of processing WSCPs between agents results in a less computationally expensive search of the environment.

6.1 Partial Observability

The proposed solution to solving WSCPs assumes the limitation of partial observability, e.g., the location of all services are not known at all times. Other works that use classical graph planning solutions, such as [29], assume a higher degree of observability and knowledge of the environment which is unrealistic given the amount of services, cloud data centers and the computational cost of maintaining the knowledge. Our proposed solution finds a middle-ground by logging information about complete WSCP solutions to increase the search performance.

6.2 Energy Efficiency

In addition to making improvements over traditional exhaustive search algorithms, the proposed system makes use of metrics such as service energy efficiency and the traversal time of routes between communicating entities. Where multiple services from different cloud data centers can be used to fulfil a request, the agents use the two aforementioned metrics to decide which service should be used to fulfil the request.

Figs. 5 and 6 show the results of 50 composition requests performed in succession with the same agent-matchmaker. The baseline search algorithm (Fig. 5), does not make use of the proposed agent functions and simply searches the available cloud data centers for corresponding services. Three variables are used in the analysis: (1) Search actions represents the number computational actions performed in the search for matching services (e.g., searching the data center and sending network messages to and from the agent

Fig. 6. Memory cost and action cost of the proposed memory-driven agent-based search algorithm.

TABLE 3
Performance Comparison of Related Works by Approach

Author	Approach	Evaluation Criteria	Evaluation Result	Considers Energy Efficiency
Zhang et al. [31]	Genetic Algorithm	Response Time, Reliability, Cost	1060.8 ms, 29.06%, $1060.8	×
Huang et al. [32]	Approximation Algorithm	Run Time Cost	16-453 ms depending on network structure	×
Karimi et al. [33]	Culture Genetic Algorithm	Composition Time	450-500 ms	×
Parhi et al. [34]	Agent Ontological Approach	Average Execution Time, Search Efficiency	0.0006-0.0007 ms, 55-60%	×
Akinwunmi et al. [35]	Trust Based	Round Trip Time	301 ms	×
Wang et al. [36]	NetMIP, WebCloudSim	Resource Consumption, QoS optimality, Computation Time	2,453 bytes on average, 1.0 on average, 80 ms	×
Wang et al. [37]	Particle Swarm Optimization Algorithm	Energy Consumption	Saving 35% of Energy of Active Servers	✓
Wang et al. [38]	Skyline Component Computation	Reliability, Time Cost	Saving 35% of Energy of Active Servers	×

representative). This feature is a general measure of how much work is performed to complete the composition request with lower values representing a more efficient search. (2) Energy cost represents the cost associated with searching for services within the cloud center. Each service has an associated energy cost which corresponds to the computation cost of using the service. (3) Memory cost is an agent-specific measure of the data remembered between composition requests (i.e., the location of past successful compositions). Over time the memory cost increases as more information about the locations and meta-data regarding statistics about the cloud data center are stored.

The benefit of having an agent-based memory function rather than a central system is that agents can in effect "specialise" their memory for a group local users that make use of the agent's services, for example, the agent may store information unique to a set of user requests that may not be useful for other user requests. In a central repository, all information for all users would be processed to find the relevant information, however, in a decentralised user-group system such as this, the relevant information can be found quicker as non-relevant information is stored elsewhere. For the baseline search that does not utilize the agent memory functions, this remains at a constant zero cost. The average action cost for the baseline approach is 46584.3584 and for the agent-based memory-driven approach is 24452.7752 making the agent system on average 52 percent more efficient.

In comparison to the baseline, the memory-driven agent search incurs a memory cost but due to the off-line meta-data gathering functions and memory of past compositions, the cost of search overall is reduced. While the memory cost is an additional burden on the agent-matchmaker, in future work we aim to consider ways to distribute this cost between the users so the burden of storing the data is reduced.

Energy efficiency is an important factor to consider for web services compositions, although many research papers (See Table 3) often overlook it for more traditional evaluation criteria such as the composition time or cost. However, other approaches such as [37] focus on energy consumption of the physical data centers' infrastructure.

6.3 Price Efficiency

The simulator was expanded to consider the cost of obtaining services from cloud data centers. Each data center

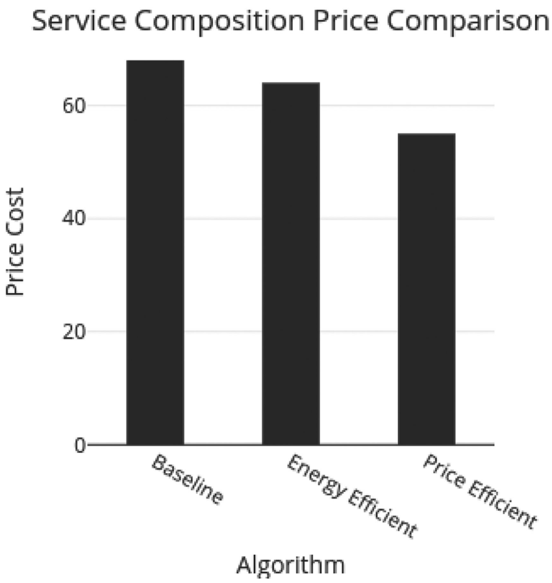

Fig. 7. A comparison of the cost of using the baseline algorithm, the agent-based energy efficient algorithm and the adapted agent-based cost efficient algorithm.

location has an associated timezone and several hours declared as being off-peak, during which the cost of using the services is reduced. The baseline and energy efficient algorithms do not consider the price in their compositions while the cost efficient algorithm, which functions similarly to the energy efficient algorithm, however prioritizes cost over energy. The off-line meta-data gathering function from the previous section is used to build a timetable of cloud data centers that are in their off-peak timezone and are given priority when cost efficiency is required. Experiments have shown that the energy efficient algorithm is on average more than 50 percent efficient in searching the cloud data centers compared to the baseline (owing to the agent memory and efficiency prioritization), however, the cost efficient algorithm is typically less efficient than the energy efficient approach, providing a 10-20 percent cost reduction, as shown in Fig. 7. The prioritization of these two features can be set by the user to reflect their individual needs.

6.4 Model Vulnerabilities

While distributed agent-based systems offer improvements over monolithic processing approaches, they necessarily incur costs that can limit the effectiveness of the system. The proposed model described in Section 4 relies on the use of distributed agents capable of storing information about past events. As a result, additional memory is required for each agent to be able to store past events which adds an additional cost and overhead to the system. This cost is managed by only storing fully complete past events (i.e., no partial or incomplete WSC proposals) to reduce the amount of information stored. Furthermore, as with any distributed agent-based approach, high availability is required to ensure that processing can take place in real-time without delay.

The benefit of employing multiple agents rather than handling all requests through a central system is the distribution of work that can be spread across the whole network. Rather than having one location that may suffer from localised problems such as routing errors or loss of power, the network of agents can provide a more available and robust system to handle requests. The disadvantage of this approach is that each agent is smaller in capacity than any central system and as such cannot handle as many concurrent requests. Within the simulator, multiple users can interact with a single matchmaking agent to represent concurrent compositions (see Fig. 2), however we leave a discussion on the actual capacity of those agents to future work in a live environment where the hardware capacity of an agent can be studied.

7 Conclusions and Future Work

In this paper, we have presented a Multiagent architecture for processing web service composition requests. Using a combination of agent-matchmakers that process the requests of the user and agent representatives that mediate communication between the matchmaker and cloud data center, our Multiagent-based approach, driven by localised memory, has shown to be an effective way to perform cost and energy efficient search of the cloud network. Evaluated on the ICEBE05 dataset, the agent-based memory-driven approach of remembering successful past service compositions for use in future events improved the action cost (a measure of how much work must be done to fulfil the request) by 52 percent making the system as a whole an efficient way to fulfil composition requests.

Further, as part of the future work, the proposed multiagent-based approach will be put in practice on a real world multi-clouds system in order to examine its viability and applicability on such complex real scenarios.

References

[1] T. A. Luxembourg, “Vat aspects of cloud computing in luxembourg,” Jul. 2013. [online]. Available https://www.vatglobal.com/countries/luxembourg
[2] Logicworks, “Vendor lock-in is big roadblock to cloud success, survey finds,” Dec. 2017. [Online]. Available: http://www.logicworks.com/blog/2016/08/vendor-lock-in-is-big-roadblock-to-cloud-success-survey-finds/
[3] B. Aldawsari, T. Baker, and D. England, “Towards a holistic brokerage system for multi-cloud environment,” in *Proc. 10th IEEE Int. Conf. Internet Technol. Secured Trans.*, 2015, pp. 249–255.
[4] T. Baker, B. Al-Dawsari, H. Tawfik, and Y. Ngoko, “GreeDi: An energy efficient routing algorithm for big data on cloud,” *Ad Hoc Netw.*, vol. 35, pp. 83–96, Dec. 2015.
[5] R. Buyya, R. Ranjan, and R. N. Calheiros, “InterCloud: Utility -oriented federation of cloud computing environments for scaling of application services,” in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2010, pp. 13–31.
[6] A. N. Toosi, R. O. Sinnott, and R. Buyya, “Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka,” *Future Generation Comput. Syst.*, vol. 79, no. 2, pp. 765–775, 2018.
[7] J. G. Olivier, G. Janssens-Maenhou, M. Muntean, and J. Peters, “Trend in global CO2 emissions,” PBL Netherlands environmental assessment agency and EC-JRC, 2016. [Online]. Available http://edgar.jrc.ec.europa.eu/news_docs/jrc-2016-trends-in-global-co2-emissions-2016-report-103425.pdf
[8] C. Jiang, Y. Wang, D. Ou, B. Luo, and W. Shi, “Energy proportional servers: Where are we in 2016?,” in *Proc. 37th IEEE Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1649–1660.
[9] W. Shi and T. F. Wenisch, “Energy-efficient data centers,” *IEEE Internet Comput.*, vol. 21, no. 4, pp. 6–7, Jul. 2017.
[10] E. Gleeson, “Computing industry set for a shocking change,” (2009, Apr.). [Online]. Available https://moneyweek.com/13297/computing-industry-set-for-a-shocking-change-43226/

[11] R. Liu, F. Chen, H. Yang, W. Chu, and Y.-B. Lai, "Agent-based web services evolution for pervasive computing," in *Proc. 11th IEEE Asia-Pacific Softw. Eng. Conf.*, 2004, pp. 726–731.
[12] G. Zou, Y. Xiang, Y. Gan, D. Wang, and Z. Liu, "An agent-based web service selection and ranking framework with QoS," in *Proc. 2nd IEEE Int. Conf. Comput. Sci. Inf. Technol.*, 2009, pp. 37–42.
[13] O. Kwona, G. P. Im, and K. C. Lee, "An agent-based web service approach for supply chain collaboration," *Scientia Iranica*, vol. 18, no. 6, pp. 1545–1552, 2011.
[14] J. Bentahar, Z. Maamar, W. Wan, D. Benslimane, P. Thiran, and S. Subramanian, "Agent-based communities of web services: An argumentation-driven approach," *Service Oriented Comput. Appl.*, vol. 2, no. 4, pp. 219–238, 2008.
[15] J. Gutierrez-Garcia and K. Sim, "Agent-based cloud workflow execution," *Integr. Comput.-Aided Eng.*, vol. 19, no. 1, pp. 39–56, 2012.
[16] J. Gutierrez-Garcia, F. Ramos-Corchado, and J. Koning, "An obligation-based framework for web service composition via agent conversations," *Int. J. Web Intell. Agent Syst.*, vol. 10, no. 2, pp. 135–150, 2012.
[17] J. O. Gutierrez-Garcia and K. M. Sim, "Agent-based cloud service composition," *Appl. Intell.*, vol. 38, no. 3, pp. 436–464, 2013.
[18] M. Parhi, B. Pattanayak, and M. Patra, "A multi-agent-based framework for cloud service description and discovery using ontology a multi-agent-based framework for cloud service description and discovery using ontology," in *Intelligent Computing, Communication and Devices*. Berlin, Germany: Springer, Jan. 2015, pp. 337–348.
[19] M. Parhi and B. K. Pattanayak, "A multi-agent-based framework for cloud service description and discovery using ontology," *ARPN J. Eng. Appl. Sci.*, vol. 9, no. 4, pp. 542–553, 2014.
[20] K. M. Sim and B. Shi, "Concurrent negotiation and coordination for grid resource coallocation," *IEEE Trans. Syst. Man Cybern.*, vol. 40, no. 3, pp. 753–766, Jun. 2010.
[21] K. M. Sim, "Grid resource negotiation: Survey and new directions," *IEEE Trans. Syst. Man Cybern.*, vol. 40, no. 3, pp. 245–257, May 2010.
[22] K. M. Sim, "Evolving fuzzy rules for relaxed-criteria negotiation," *IEEE Trans. Syst. Man Cybern.*, vol. 38, no. 6, pp. 1486–1500, Dec. 2008.
[23] D. Yoo and K. M. Sim, "A multilateral negotiation model for cloud service market," in *Proc. Grid Distrib. Comput. Control Autom. Commun. Comput. Inf. Sci.*, 2010, pp. 54–63.
[24] K. M. Sim, "Towards complex negotiation for cloud economy," in *Proc. Int. Conf. Advances Grid Pervasive Comput.*, 2010, pp. 395–406.
[25] A. More, S. Vij, and D. Mukhopadhyay, "Agent based negotiation using cloud – An approach in e-commerce," in *Proc. 48th Annu. Conv. Comput. Soc. India ICT Critical Infrastructure*, 2014, pp. 489–496.
[26] J. Kang and K. M. Sim, "Cloudle: A multi-criteria cloud service search engine," in *Proc. IEEE Asia-Pacific Serv. Comput. Conf.*, 2011, pp. 312–318.
[27] J. Kang and K. M. Sim, "Cloudle: An ontology-enhanced cloud service search engine," in *Proc. 1st Int. Workshop Cloud Inf. Syst. Eng. Collocated 11th Int. Conf. Web Inf. Syst. Eng.*, 2010, pp. 416–427.
[28] J. Kang and K. M. Sim, "Cloudle: An agent-based cloud search engine that consults a cloud ontology," in *Proc. Int. Conf. Cloud Comput. Virtualization*, 2010, pp. 312–318.
[29] S. C. Oh and S. R. T. Kumara, "Effective web services composition in diverse and large-scale services networks," *IEEE Trans. Services Comput.*, vol. 1, no. 1, pp. 15–32, Jan.–Mar. 2008.
[30] H. K. B. University, "*IEEE Int. Conf. e-Business Engineering (ICEBE05) Dataset*," China. (2005). [Online]. Available: http://www.cs.hku.hk/icebe2005/
[31] M. Zhang, L. Liu, and S. Liu, "Genetic algorithm based QoS-aware service composition in multi-cloud," in *Proc. IEEE Conf. Collaboration Internet Comput.*, 2016, pp. 113–118.
[32] J. Huang, Y. Liu, R. Yu, Q. Duan, and Y. Tanaka, "Modeling and algorithms for QoS-aware service composition in virtualization-based cloud computing," *IEICE Trans. Commun.*, vol. E96-B, no. 1, pp. 10–19, 2013.
[33] M. B. Karimi, A. Isazadeh, and A. M. Rahmani, "QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm," *J. Supercomput.*, vol. 73, no. 4, pp. 1387–1415, 2017.
[34] M. Parhi, B. K. Pattanayak, and M. R. Patra, "A multi-agent-based framework for cloud service discovery and selection using ontology," *Service Oriented Comput. Appl.*, vol. 12, no. 2, pp. 137–154, 2018.
[35] A. Akinwunmi, E. Olajubu, and G. Aderounmu, "A multi-agent system approach for trustworthy cloud service discovery," *Cogent Eng.*, vol. 3, no. 1, pp. 1–17, 2016.
[36] S. Wang, A. Zhou, F. Yang, and R. N. Chang, "Towards network-aware service composition in the cloud," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2603504.
[37] S. Wang, A. Zhou, C.-H. Hsu, X. Xiao, and F. Yang, "Provision of data-intensive services through energy- and QoS-aware virtual machine placement in national cloud data centers," *IEEE Trans. Emerging Topics Comput.*, vol. 4, no. 2, pp. 290–300, Apr.–Jun. 2016.
[38] S. Wang, A. Zhou, M. Yang, L. Sun, C.-H. Hsu, and F. Yang, "Service composition in cyber-physical-social systems," *IEEE Trans. Emerging Topics Comput.*, to be published, doi: 10.1109/TETC.2017.2675479.

Philip Kendrick is working toward the PhD degree in the Department of Computer Science, Liverpool John Moores University, United Kingdom. He is interested in the areas of multi-agent systems, machine learning, and cyber security. More specifically, his work examines how non-linear algorithms can be used to improve the performance of networked systems.

Thar Baker is a senior lecturer with software systems engineering, head of the Computer Science Research Group and member of the Applied Computing Research Group in the Department of Computer Science, Liverpool John Moores University, United Kingdom. He has published numerous referred research papers in multidisciplinary research areas including: cloud computing, algorithm design, energy efficiency, and Web service computing. He has been involved as a member of the editorial board and review committees for a number of international journals and conferences. He is a member of the IEEE.

Zakaria Maamar received the PhD degree in computer science from Laval University, Canada. He is a professor with the College of Technological Innovation, Zayed University, Dubai campus. His research interests include mobile computing, business process management, and social computing. He has published several papers in peer-reviewed journals and conference proceedings.

Abir Hussain is a professor and the head of the Applied Computing Research Group, Faculty of Engineering and Technology, Liverpool John Moores University, United Kingdom. She has published numerous referred research papers in conferences and Journal in the research areas of Neural Networks and Telecommunication Fraud Detection. Her research has been published in a number of high esteemed and high impact journals such as the *Expert Systems with Applications*, the *PloS ONE*, the *Electronic Letters*, the *Neurocomputing*, and the *Neural Networks and Applications*. She is one of the initiators and chairs of the Development in e-Systems Engineering (DeSE) series, most notably illustrated by the IEEE technically sponsored DeSE International Conference Series. She is a member of the IEEE.

Rajkummar Buyya is a Redmond Barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the university, commercializing its innovations in cloud computing. He has authored more than 625 publications and seven text books including *Mastering Cloud Computing* published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese, and international markets, respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=118, g-index=245, 73,200+ citations). He is recognized as a "Web of Science Highly Cited Researcher" in 2016 and 2017 by Thomson Reuters, a fellow of the IEEE, and scopus researcher of the Year 2017 with the Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. For further information please visit his cyberhome: www.buyya.com.

Dhiya Al-Jumeily is a professor of artificial intelligence and the associate dean of external engagement for the Faculty of Engineering and Technology. He has extensive research interests covering a wide variety of interdisciplinary perspectives concerning the theory and practice of applied computing in medicine, human biology, and health care. He has published more than 170 peer reviewed scientific publications, six books, and five book chapters, in multidisciplinary research areas. He is a senior member of the IEEE.