Expert Systems with Applications 42 (2015) 135–145

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# Imperialist competitive algorithm with PROCLUS classifier for service time optimization in cloud computing service composition

Amin Jula [a,*], Zalinda Othman [a], Elankovan Sundararajan [b]

[a] Data Mining and Optimization Research Group, Centre for Artificial Intelligence, UKM Bangi, 43600 Selangor, Malaysia
[b] Centre of Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, UKM Bangi, 43600 Selangor, Malaysia

ARTICLE INFO

*Article history:*
Available online 4 August 2014

*Keywords:*
Cloud computing
Service composition
Service selection
Service time
Quality of service
QoS
Imperialist competition algorithm
Clustering
Proclus

ABSTRACT

Aiming to provide satisfying and value-added cloud composite services, suppliers put great effort into providing a large number of service providers. This goal, achieved by providing the "best" solutions, will not be guaranteed unless an efficient *composite service composer* is employed to choose an optimal set of required unique services (with respect to user-defined values for quality of service parameters) from the large number of provided services in the pool. Facing a wide service pool, user constraints, and a large number of required unique services in each request, the composer must solve an NP-hard problem. In this paper, *CSSICA* is proposed to make advances toward the lowest possible service time of composite service; in this approach, the PROCLUS classifier is used to divide cloud service providers into three categories based on total service time and assign a probability to each provider. An improved imperialist competitive algorithm is then employed to select more suitable service providers for the required unique services. Using a large real dataset, experimental and statistical studies are conducted to demonstrate that the use of clustering improved the results compared to other investigated approaches; thus, *CSSICA* should be considered by the *composer* as an efficient and scalable approach.



## 1. Introduction

In an era where computation complexity is growing dramatically and achieving desired results depends on the processing of big data, the computation world has no choice but to recognize the use of cloud computing (Armbrust et al., 2010; Hayes, 2008). Choosing each of the deployment models of cloud computing (public, community, private, and hybrid clouds (Dillon, Chen, & Chang, 2010; Peter Mell, 2011)) would provide required service models (Ellinger, 2013) with different security policies (Takabi, Joshi, & Gail-Joon, 2010; Wei et al., 2014; Zissis & Lekkas, 2012). According to a widely accepted classification, each service can belong to one of the three categories: software as a service (SaaS), platform as a service (PaaS), or infrastructure as a service (IaaS), which can provide more effective functionalities in cooperation and combination of other services.

Nevertheless, the increasing tendency of applicants to receive services from the cloud has led to an unprecedented increase in the number of providers who want to present their services in a cloud service pool. Hence, we are faced with a large number of unique services provided with similar functionality and different quality of service (QoS) (Jula, Sundararajan, & Othman, 2014).

On the other hand, due to the availability of complicated and varied services, a distinct simple service is unable to meet the prevailing functional prerequisites for several real-world cases. A set of simple atomic services that are able to work together is necessary to perform a complicated service. We also encounter hanging customer requirements from simple services into complicated services, along with a set of constraints, priorities, and QoS requirements (e.g., service time). Therefore, cloud suppliers must provide a package, referred to here as the *composite service composer (CSC)*, which is a set of components that searches for the best composition of pre-provided unique services in the service pool based on customer requirements and constraints. Because of the immense variety of unique services and large number of service providers, as well as the importance of customer-defined requirements and constraints, *CSC* is faced with an NP-hard problem referred to as cloud computing service composition *(CCSC)* (Fei, Yuanjun, Lida, & Lin, 2013; Li, Cheng, Ou, & Zhang, 2010; Wada, Suzuki, Yamano, & Oba, 2012) when seeking the optimal response to any request for a composite service.

Many studies have been conducted and many different heuristic and non-heuristic algorithms (Barney, 2012; Gutierrez-Garcia & Sim, 2010; Kofler, Haq, & Schikuta, 2010; Kofler, ul Haq, & Schikuta,

* Corresponding author.
*E-mail addresses:* amin.jula@gmail.com, aminjula@ftsm.ukm.my (A. Jula), zalinda@ftsm.ukm.my (Z. Othman), elan@ftsm.ukm.my (E. Sundararajan).

http://dx.doi.org/10.1016/j.eswa.2014.07.043
0957-4174/

2009; Zhou & Mao, 2012) have been proposed to promote the quality of functionality of *CSC* and achieve an optimal solution for *CCSC*. The application of particle swarm optimization (Kennedy & Eberhart, 1995), genetic algorithms (Holland, 1992), game theory (Scutari, Palomar, & Barbarossa, 2008), and memetic algorithms (Jula & Naseri, 2011) has led to significant improvements in the optimization of proper service selection for service composition (Jula, Othman, & Sundararajan, 2013; Ludwig, 2012; Qi & Bouguettaya, 2013; Wang, Sun, Zou, & Yang, 2013; Xia, Wan, Dai, Luo, & Sun, 2012; Xiao & Zhang, 2012; Yang, Mi, & Sun, 2012; Ye, Zhou, & Bouguettaya, 2011; Zhu, Li, Luo, & Zheng, 2012); however, none of these approaches use the *clustering* of the service providers to achieve improved performance (Jula et al., 2014).

Despite all the research conducted in the domain, one of the main but almost neglected issues in facing very large search space problems is the large number of available options for being selected, and the limited number of solutions in the initialization phase of the algorithm. In other words, a very small percentage of potential solutions can be chosen as the first generation of solutions. Because selection of members of the initial generation of solutions is usually randomized, it is obvious that chance of selecting appropriate solutions is extremely low. Lack of providing proper initial solutions will lead to improper evolution process in execution of algorithm. Hence, it can be concluded that applying non-blind selection techniques can significantly enhance the performance of the algorithms. To reach this aim, two different approaches can be followed, including proposing intelligent initialization methods and operators, and search space clustering techniques. What previously has been employed to address this problem is restricted to the use of skyline operator to limit the number of service providers (Qi & Bouguettaya, 2013; Wang et al., 2013). What is the main challenge of using Skyline operator in real-world *CCSC* is high time-complexity that increases exponentially with the increase in the number of QoS parameters.

In this paper, PROCLUS (Aggarwal, Wolf, Yu, Procopiuc, & Park, 1999; Kurniawan et al., 1999) (as an appropriate clustering algorithm for high-dimensional data) is used to categorize service providers into three classes based on the time of their provided services. Afterwards, the probability of selection of single services from each category is calculated based on the average service time of all assigned service providers in each category. Using selection probability of the categories in initialization of first generation of solutions has altered the blind production of the first set of solutions and thus reduced the service time of the final solution.

But then imperialist competitive algorithm (ICA) as a very young evolutionary algorithm (Atashpaz-Gargari & Lucas, 2007) presents three attractive features to be used in solving *CCSC*. One of the most important concerns in solving optimization problems with very large search spaces is falling into trap of local optima. ICA searching process is designed in such a way that the probability of falling into the trap effectively reduced by frequent relocation of solutions in the search space using different well-designed operators. Novelty of utilizing sociopolitical behavior of countries in designing the algorithm, and extensive areas of innovation because of being newer than most of proposed evolutionary algorithms, are other two advantages of applying ICA.

In this study, for addressing Service Time Optimization in Cloud Computing Service Composition (STOCCSC), Classified Search Space Imperialist Competitive Algorithm (*CSSICA*) is proposed. In *CSSICA*, along with applying classified search space and using achieved probabilities in selection procedure, improvements in the structure and functioning methods of the ICA are also adopted to improve the algorithm's performance.

The remainder of this paper is organized as follows. After the introduction, descriptions of Service Time optimization in Cloud Computing Service Composition (*CCSC*) and Imperialist Competitive Algorithm (ICA) are presented in Section 2. A complete description of the proposed algorithm, *CSSICA*, is provided in Section 3. A detailed discussion of the experimental results obtained using *CSSICA* and two other algorithms, ICA and Niching PSO, including numerical and statistical investigations, is provided in Section 4. Finally, our conclusions and potential topics for future research are presented in Section 5.

## 2. Problem and algorithm description

### 2.1. Service time optimization in cloud computing service composition (STOCCSC)

An increase in the number of available services will lead to an increase in the number of similar operating services on various servers. Because these similar services are to be found in various places and have definite QoS parametric values, the *CSC* should employ suitable methods to select a unique service among the various similar services that are available on distinct servers. Applying the appropriate method will enable the best quality of service to be attained based on the requirements and priorities of the end user. In view of fundamental changes in cloud environments, accessible services, and end-user requirements, the *CSC* must be powerfully designed with automatic functional capabilities.

Thus, one of the most significant problems in service composition is the selection of suitable simple services to be merged together to produce an ideal combination that is able to meet the functional and QoS requirements of the end user.

This paper assumes that every *composite service (CS)* within a cloud is comprised of *n unique services (USs)*, all of which have *service time (ST)* as a QoS parameter. A combination of exclusive services must correspondingly act in an ordinal *workflow (wf)* to terminate a *CS*. However, if $wf_k$ is the workflow of $CS_k$, then $ST(wf_k)$ can be defined to denote the *ST* value of the workflow *k*, whereupon the *ST* vector of the workflow can be expressed as (1).

$$ST(wf_k) = (ST_1(wf_k), \quad ST_2(wf_k), \ldots, ST_n(wf_k)) \tag{1}$$

The competency value (CV) of $wf_k$ is the total service time of $wf_k$ and is obtained using (2). The optimal solution for *STOCCSC* is the solution with the lowest *ST* and thus with the lowest *CV*.

$$CV(wf_k) = \sum_{i=1}^{n} ST_i(wf) \tag{2}$$

### 2.2. Imperialist competitive algorithm

In the field of evolutionary computation, the novel ICA is founded on human social and political advancements (Atashpaz-Gargari & Lucas, 2007; Bahrami, Faez, & Abdechiri, 2010; Zarandi, Zarinbal, Ghanbari, & Turksen, 2013), unlike other evolutionary algorithms, which are based on the natural behaviors of animals or physical events. ICA starts with an initial randomly generated population, in which the individuals are known as countries. Some of the best countries are considered imperialists, whereas the other countries represent the imperialist colonies. To solve an optimization problem with *n* dimensions, a country is formed as an $1 \times n$ array as follows:

$$Country = [p_1, p_2, \ldots, p_n] \tag{3}$$

The power of a country *i* is calculated using the objective function *f*, which is a function of the variables $(p_1, p_2, \ldots, p_n)$, yielding the following equation:

$$Power(Country_i) = f(Country_i) = f(p_{i1}, p_{i2}, \ldots, p_{in}) \tag{4}$$