University of Wollongong

Research Online

Faculty of Engineering and Information
Sciences - Papers: Part A

Faculty of Engineering and Information
Sciences

1-1-2014

A systematic review of bio-inspired service concretization

Lijuan Wang

Xidian University, lw840@uowmail.edu.au

Jun Shen

University of Wollongong, jshen@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/eispapers>

Part of the [Engineering Commons](#), and the [Science and Technology Studies Commons](#)

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

A systematic review of bio-inspired service concretization

Abstract

In service oriented computing, Web service selection is an important part of Web service composition. The Web service composition is achieved by solving the Web service concretization problem. The literature presents two types of Web service concretization approaches: local optimization approaches and global optimization approaches. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. The bio-inspired algorithms are sub-optimal methods. This paper will firstly present a hierarchical taxonomy of Web service concretization approaches. Then we conduct a systematic review on the current research of Web service concretization based on three bio-inspired algorithms, namely, ant colony optimization algorithms, genetic algorithms, and particle swarm optimization algorithms. Based on the findings from the systematic review, this paper also discusses the underlying applications of bio-inspired algorithms to the data-intensive service concretization problems.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Wang, L. & Shen, J. (2014). A systematic review of bio-inspired service concretization. IEEE Transactions on Services Computing, Online First 1-14.

A Systematic Review of Bio-Inspired Service Concretization

Lijuan Wang and Jun Shen, *Senior Member, IEEE*

Abstract—In service oriented computing, Web service selection is an important part of Web service composition. The Web service composition is achieved by solving the Web service concretization problem. The literature presents two types of Web service concretization approaches: local optimization approaches and global optimization approaches. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. The bio-inspired algorithms are sub-optimal methods. This paper will first present a hierarchical taxonomy of web service concretization approaches. Then we conduct a systematic review on the current research of Web service concretization based on three bio-inspired algorithms, namely, ant colony optimization algorithms, genetic algorithms, and particle swarm optimization algorithms. Based on the findings from the systematic review, this paper also discusses the underlying applications of bio-inspired algorithms to the data-intensive service concretization problems.

Index Terms—Ant colony optimization, genetic algorithm, particle swarm optimization, Web service concretization, Web service composition, quality of service

1 INTRODUCTION

As a major software framework for distributed applications, service oriented architecture [1] uses standard protocols to integrate existing component Web services into complex business processes and applications, which are referred to as Web service composition. These component Web services are developed independently by different service providers, so some services may have same functionality but differ in quality of service (QoS) attributes as well as other non-functional properties. In the context of Web service composition, abstract services are the functional descriptions of services, and concrete services represent the existing services available for potential invocation of their functionalities and capabilities. Given a request of composite service, which involves a set of abstract services and dependency relationships among them, there is a list of service candidate sets, which includes many concrete services for each abstract service. Web service selection refers to finding one service candidate to implement each abstract service according to users' requirements, which is an important part of Web service composition. For each abstract service of a composite service, the service concretization process is to bind one of its corresponding concrete services and meet the constraints specified for some of the QoS attributes [2], [3]. The final goal of the composite service construction is achieved by solving the well-known service

concretization problem. As most of the papers used the terms composition and selection interchangeably, we used the term "service concretization" in our paper to umbrella them. So far, many service concretization approaches have been designed. A hierarchical taxonomy of service concretization approaches will be given in this paper. The bio-inspired algorithms are one group of the main approaches.

One of our earlier studies [4] presented a survey on bio-inspired algorithms for Web service concretization, and it also proposed a bio-inspired cost minimization mechanism for data-intensive service provision. However, it did not describe the applications of the presented algorithms to Web service concretization specifically. There are also a few other overviews on service composition and bio-inspired optimization algorithms. The authors of [5] conducted an overview of automatic Web service composition from the perspective of workflow and AI planning. The study [6] discussed the urgent need for service composition, the required technologies to perform service composition, as well as several different composition strategies based on existing composition platforms and frameworks. The study [7] introduced a technical survey on automated service composition and discussed the whole automatic composition with respect to the service classification, the service combination, the service selection, the service description, and the service matchmaking.

However, to the best of our knowledge, no overview on service concretization problems based on bio-inspired algorithms has been published yet. This paper investigates the existing studies using bio-inspired algorithms to deal with QoS attributes, the optimality, and the dynamicity in Web service concretization. We will describe the applications of ant colony optimization (ACO) algorithms, genetic algorithms (GA), and particle swarm optimization (PSO) algorithms to Web service concretization, and also present a systematic review based on these three bio-inspired

• L. Wang is with the School of Cyber Engineering, Xidian University, Xi'an, China. E-mail: lw840@uowmail.edu.au.

• J. Shen is with the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia. E-mail: jshen@uow.edu.au.

Manuscript received 27 Feb. 2014; revised 5 Nov. 2015; accepted 12 Nov. 2015. Date of publication 17 Nov. 2015; date of current version 9 Aug. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2015.2501300

Fig. 1. A hierarchical taxonomy of Web service concretization approaches.

algorithms. The foundational investigations indicated that the bio-inspired algorithms could overcome the new challenging requirements of the data-intensive service concretization problem. The main contributions of this paper are four-fold: first, a hierarchical taxonomy of service concretization approaches is given; second, a detailed analysis of each method in the hierarchical taxonomy is made; third, a systematic survey and the analysis of Web service concretization based on bio-inspired algorithms are presented; fourth, the underlying applications of bio-inspired algorithms to the data-intensive service concretization problem are proposed. This paper gives an overview for people interested in bio-inspired service concretization. Moreover, through this paper the community can understand service concretization more comprehensively.

The paper is organized as follows. Section 2 introduces a hierarchical taxonomy of Web service concretization approaches. Section 3 describes our systematic review protocol. Section 4 shows the applications of the three types of bio-inspired algorithms to Web service concretization problems. Then Section 5 discusses the underlying applications of bio-inspired algorithms for tackling the data-intensive service concretization problems. Finally, Section 6 concludes this paper.

2 WEB SERVICE CONCRETIZATION APPROACHES

The Web service concretization approaches should be evaluated with respect to their optimality, their computational efficiency, and their dynamic complexity. The optimality is the extent to which the approach produces the best solution. The computational efficiency is measured by the time that the approach takes to produce a solution. The dynamic complexity is referred to the time complexity that the approach deals with the re-optimization. The literature presents two types of Web service concretization approaches: local optimization approaches and global optimization approaches. Fig. 1 shows a hierarchical taxonomy of Web service concretization approaches.

2.1 Local Optimization Approaches

In local optimization approaches, service selection is undertaken independently for each abstract service [8], [9], [10]. This type of approach is very efficient in terms of computation time, as the time complexity is $O(m)$ by using service broker,

peer-to-peer, or agent-based architecture to perform selection in parallel, where m is the number of concrete services for each abstract service. Even if local optimization approaches are useful in decentralized and dynamic environments, they are not suitable for service concretization with global QoS constraints, since they can only guarantee local QoS constraints. If there is no requirement specifying global constraints, then local optimization approaches are preferable.

From the viewpoint of computational time, the local optimization approaches can be appropriate when the global QoS constraints are transformed, i.e., decomposed, into local QoS constraints in order to overcome their drawbacks. A few decomposition methods have been proposed.

The study [11] used the concept of local quality levels and modeled the decomposition problem as an optimization problem, which was formulated as a mixed integer programming (MIP) model. The local quality levels were a set of discrete representative values, which were extracted from the quality properties of all candidates. The local constraints were the selected quality levels. The study [12] applied a genetic algorithm to solve the decomposition problem, and used the concept of quality degree. The quality degrees were a set of discrete quality values, which were extracted from partitioning each quality property. The local constraints were the selected quality degrees. The authors of [13] used the mean and standard deviation of the values of QoS attributes to decompose global QoS constraints.

The main drawbacks of these decomposition methods are two-fold: the local selection relies on a greedy method, and the decomposition deals with QoS parameters independently so does not take into account potential correlations and dependencies among them. Meanwhile, in these decomposition methods there is no analysis for cases when the requirements of the global QoS constraints are over-constrained. This often leads to a very restrictive decomposition of the global QoS constraints that cannot be satisfied by any of the service candidates, even if there could be feasible solutions otherwise.

2.2 Global Optimization Approaches

Global optimization approaches can solve Web service concretization problems at both local and global service levels [14], [15]. The problems have been modeled as a 0-1 knapsack problems [16], [17], constraint optimization

problems [18], [19], multi-dimensional and/or multi-objective optimization problems [20], [21], weighted directed acyclic graph problems [22], [23], and mathematical programming problems [24], [25]. Accordingly, many algorithms have been proposed. There are three types of algorithmic methods in the global optimization approaches: optimal method, sub-optimal method, and soft constraints-based method.

Optimal methods, such as the exhaustive algorithm [26], constraint programming-based algorithms [27], and mathematical programming-based algorithms (integer programming (IP) [24], linear programming (LP) [25], or mixed integer programming [2]) are designed to find optimal solutions. IP, LP and MIP-based algorithms need linear aggregation functions for the QoS attributes. Some standard attributes, such as the availability or reliability are not linear initially. While linearization can be adopted [10], it is quite difficult in other cases, such as computing the response time for the parallel structures. An alternative would be the use of non-linear IP, however, the scalability problems would still arise [14]. When using constraint programming-based algorithms, the constraints can be expressed without translating them into linear inequalities [27]. The study [27], [28] explained that constraint programming-based algorithms were essentially very simple and sometimes they could find a solution faster than more complex mathematical programming-based algorithms. In addition, optimal methods are often practical for small scale problems, but they need more computation time, especially in dynamic environments. This is exacerbated in a situation where the QoS-aware concretization has to be re-planned at runtime because the computation time of the concretization becomes crucial. This was confirmed by the existing studies [10], [14], [17], [26], [29].

Sub-optimal methods, such as the heuristic-based algorithms (discarding subsets [16], shortest path heuristic [30], and other heuristics [17], [31]), and the metaheuristic-based algorithms (tabu search [32], simulated annealing (SA) [29], GAs [14], [33], ACO algorithms [21], [34], PSO algorithms [29], [35] and others [36]) are designed to find an optimal or a near-optimal solution. Metaheuristics provide both a general structure and strategy guidelines for developing a heuristic to solve service concretization problems. They provide an efficient way to move towards a very good solution, though it might not necessarily be the best one. When the time requirements become excessive, sub-optimal methods must be employed. Continuing efforts have demonstrated that sub-optimal methods for QoS-aware service concretization usually require much less computation time than optimal methods, and they are consistently capable of achieving near-optimal solutions [16], [26], [30], [37], [38]. Meanwhile, there is a trade-off between computation cost and quality of the solution with regard to the sub-optimal methods. That is to say, how inferior the sub-optimal methods would be when compared to an optimal method that always finds the optimal solution.

Soft constraints-based algorithms are adopted in order to allow constraint violations in less likely feasible compositions when the requirements of the global QoS constraints are overly strict [19], [39], [40]. They are proposed to predict run-time service level agreement violations for composite services. They work around over-constrained QoS-aware service concretization problem and offer a

feasible solution. The choice of evaluation functions, the definition of penalties, and the relation between the assigned penalties and the violated guarantees, are key factors in this type of methods.

2.3 Bio-Inspired Algorithms for Service Concretization

In this paper, we are interested in how bio-inspired algorithms are used to solve Web service concretization problems. Although numerous research studies have addressed Web service concretization problems (including bio-inspired algorithms), service discovery and composition mechanisms will face new challenges as cloud computing, big data, and the next generation Internet change the whole scene. We believe that applying biologically inspired approaches to services can make them adapt to dynamic service environments. For example, the study [41] already proved that it was useful for service management and discovery to add biological mechanisms to services.

Biological systems present features such as autonomy, scalability, adaptability, and robustness. They are autonomous entities and often self-organized without a central controller. The biological environment provides a medium that allows biological organisms to interact and mobilize. For example, ants release chemical signals, creating chemical gradients in the environment after finding a good path from the nest to a food source, so that letting other ants sense the chemical and follow the path to food.

The study [42] pointed out four key attributes of biological systems, which can be applied to the design of biological inspired systems.

- 1) A large number of redundant components: a biological system is composed of massive numbers of redundant components. For example, an ant colony may contain millions of ants. A biological system is robust under perturbations or conditions of uncertainty because of the large number of redundant components.
- 2) Local interactions and collective behavior: an individual ant is not able to find the shortest path to food source, but a group of ants can find the shortest path via interactions among individual ants through the placement of pheromones.
- 3) Stochastic or probabilistic nature: the probabilistic nature of biological systems helps them explore large search spaces. For example, mutation in genetic evolution randomly changes genetic information to enable individuals to adapt to the environment.
- 4) Feedback-based control: biological systems frequently use positive and negative feedback control. For example, ants use positive feedback to deposit the pheromone trails and to recruit more ants toward a particular path.

In this paper, we will discuss these characteristics in three biological systems: insect colonies, genetic evolution, and swarms. We focus on ACO algorithms, GAs, and PSO algorithms. It is both important and interesting to know the extent of applications of bio-inspired algorithms to QoS-based Web service concretization, which is not covered by previous studies. This effort allows us to provide an overview on the existing studies.

TABLE 1
Distribution of Primary Studies

Bio-inspired algorithm	Articles	Year	Cost (C)	Time (T)	Availability (Av)	Reliability (Re)
ACO	[44]	2007	✓	✓	✓	
	[45]	2008	✓	✓	✓	✓
	[46]	2010	✓	✓		✓
	[21]	2010	✓	✓	✓	✓
	[34]	2011	✓	✓	✓	✓
GA	[14]	2005	✓	✓	✓	✓
	[47]	2005	✓			
	[48]	2006	✓	✓	✓	✓
	[49]	2007	✓	✓	✓	✓
	[50]	2008	✓	✓	✓	✓
	[32]	2008	✓	✓		✓
	[33]	2008	✓	✓	✓	✓
	[51]	2009	✓	✓	✓	✓
	[18]	2010	✓	✓	✓	✓
	[52]	2012		✓	✓	✓
PSO	[53]	2009				
	[54]	2010	✓	✓		
	[55]	2012	✓	✓	✓	
	[35]	2012	✓	✓	✓	✓
Combination of GA and PSO	[56]	2007				
Combination of ACO and GA	[23]	2010	✓	✓		
Combination of SPSO and SA	[29]	2011				

3 METHOD OF SYSTEMATIC REVIEW

This section describes our systematic review protocol, consisting of several steps outlined in [43].

3.1 Research Questions

In order to find principles for applying bio-inspired algorithms to solve Web service concretization problems, we have the following research questions:

- 1) How are the three types of bio-inspired algorithms applied to solve Web service concretization problems?
After knowing this, we have three additional research questions applicable in each type of algorithms:
- 2) What are the main issues that need to be addressed if the bio-inspired algorithms are to be applied?
- 3) What are the strategies proposed to address the issues?
- 4) What are the current challenges or limitations in the application of the bio-inspired algorithms to solve Web service concretization problems?

The domain of this study is the Web service composition and selection based on ACO algorithms, GAs, and PSO algorithms. Our research questions are not aiming at making a comparison among the three algorithms, but we discuss the comparisons within the scope of each primary study to support our argument from the obtained results, and we will present comparisons by experiments.

3.2 Specification for Literature Review Strategy

We use the following terms to search the literature:

- 1) Web service composition, Web service selection.
- 2) ant colony algorithm, ant colony optimization, ant-inspired, ant system, ant-based, genetic algorithm, particle swarm optimization, particle swarm algorithm.

We used five electronic databases, namely, ACM Digital Library, ScienceDirect, Google Scholar, IEEEExplore, and EI Compendex as data sources, and papers were selected for review if they had a key term (or synonym) in the title, abstract or keywords list. We selected 2005 as the starting year for the search since this year marked the first publication of the application of genetic algorithm to Web service composition, and the ending year is 2012. The search resulted in a total of 454 papers. After eliminating duplicates found by more than one electronic database, we were left with 278 papers.

The following exclusion criteria is applicable in this review to exclude studies that:

- 1) Do not relate to Web service selection and composition.
- 2) Do not report applications of bio-inspired algorithms.
- 3) Do not report experiments and simulations.
- 4) Are related to semantic Web service composition and selection.

The application of detailed exclusion criteria resulted in 120 remaining references, which were further filtered out by reading full-text. A final figure of 22 primary studies was reached after excluding similar studies that were published in different venues. Relevant information describing the distribution of primary studies within the four QoS attributes is shown in Table 1. The four QoS attributes which are the most commonly used are: response time, cost, reliability and availability, and according to [15], they represent a selection of the most relevant characteristics in the context of services. Response time, run time, and execution time might have the same or different definitions, and Table 1 places them into the single time column. In addition, the studies [29], [53], [56] did not provide details about whether the four QoS attributes were considered or not.

TABLE 2
A Summary of Results on Applying ACO Algorithms to Web Service Concretization Problems

Articles	Heuristic information used	Pheromone updating was applied to	Key parameters	Limitations and highlights
[44]	C, T, Av	the best solution	$\alpha = 1, \beta = 2, \rho = 0.05$	Effects of the number of abstract services are not taken into account. Further investigation is required to deal with all situations of clone ants.
[45]	C, T, Av, Re	the most satisfying solution	$\alpha = 2, \beta = 2$	The setting of parameter is not given and further performance evaluation is required.
[46]	C, T, Re	the solution in known Pareto Front	$\alpha = 1, \beta = 1, \rho = 0.3$	There is a need to use different metrics to measure the performance of multi-objective optimization.
[21]	C, T, Av, Re	all the solutions	$\alpha \in [2, 3, 4, 5], \beta \in [4, 8, 9], \rho = 0.7$	All the abstract execution paths are not consistent with the definition, especially those in the experimental part. The optimal of abstract execution paths cannot guarantee the global QoS constraints of a composite service.
[34]	C, T, RE, Av, and five domain QoS attributes	the top solutions and other remaining solutions used different updating rule	$\alpha = 1, \beta = 4, \rho = 0.5$	Effects of the number of abstract services and the number of concrete services are not taken into account. There is a need to devise different scenarios to test scalability.

4 FINDINGS FROM THE SYSTEMATIC REVIEW

4.1 Ant Colony Optimization Algorithms

The ant colony algorithm modeling the behavior of real ants is widely used for combinatorial optimization problems. When ACO algorithms are used to solve a Web service concretization problem, the problem is modeled as a weighted directed acyclic graph with a *start* vertex and an *end* vertex. The graph is denoted by $G = \langle V, E, \text{QoS} \rangle$, two vertices $i, j \in V$ are neighbors if there exists an arc $(i, j) \in E$. Then the *start* vertex is set as the ants' nest and the *end* vertex is set as the food source, and the QoS constraints are regarded as the weights of the edges. In the graph G , all ants are initially positioned at the *start* vertex and the task of each ant is to find a path from the *start* vertex to the *end* vertex. Thus, the problem is transformed from finding feasible solutions to the service concretization problem into selecting paths through the weighted graph. The basic operational flow in an ACO algorithm is as follows.

- 1) Initialize ACO parameters;
- 2) Generate random solutions from each ant's random walk;
- 3) Update pheromone intensities;
- 4) Steps 2 to 3 are repeated until a termination condition has been satisfied.

Beginning from the *start* vertex, each ant builds up a solution iteratively by always selecting the next vertex based on pheromone trail τ_{ij} and heuristic information η_{ij} . The pheromone trail and the heuristic information are indicators of tendency to move from vertex i to vertex j . In the process of searching, when located at a vertex i , each ant k chooses the next vertex j according to the probability distribution defined by (1),

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, & \text{if } j \in N_i^k; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The parameters α and β control the influence of pheromone value and heuristic value. The variable N_i^k is used to denote the set of unvisited vertices, which contains all the direct successors of vertex i in the graph G . Each ant changes the pheromone value τ_{ij} according to (2),

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau^k, \quad \forall (i, j) \in E. \quad (2)$$

The variable ρ is the pheromone evaporation coefficient, and $\Delta\tau^k$ is the amount of pheromone deposited by ant k . The choice of $\Delta\tau^k$ is an important aspect of the algorithm. It can be the same constant value for all the ants. Generally, it is required that the amount of pheromone deposited by an ant to be a non-increasing function of the path length [57].

A summary of results on applying ACO algorithms to Web service concretization problems is given in Table 2. Besides the listed items, the references in Table 2 differ from each other with respect to the single objective ACO algorithm which they were based on, such as ant colony system adopted by the authors of [44], ant system adopted by the authors of [21], [45], [46], and max-min ant system adopted by the authors of [34].

The authors of [44] presented a utility function to measure the user satisfactory degree. The method was compared with an exhaustive searching method, and a test scenario with nine abstract services was used in the evaluation. The simulation results showed that the execution time of the proposed algorithm was less than that of the exhaustive searching method.

In [45], different pheromones were used to denote different QoS attributes. The simulation used eight abstract services, and the number of concrete services corresponding to each abstract service was randomly selected from 0 to 25. The experimental results showed that the dynamic ACO algorithm had better performance than the typical ACO algorithm.

The study [46] modeled the Web service selection problem as a multi-objective optimization problem, and

proposed a multi-objective chaos ACO algorithm to solve it. The chaos variable was used to improve the efficiency of the ACO algorithm. To evaluate the proposed algorithm, the authors compared it with a multi-objective GA and a multi-objective ACO algorithm based on three test groups. The evaluation criteria were the number of optimal solutions and the running time. The simulation indicated that the multi-objective chaos ACO algorithm was able to find more optimal solutions and used less time than the multi-objective GA and the multi-objective ACO algorithm.

The study [21] used a k -tuple pheromone to represent k objectives. A strategy was adopted to decompose a composite service with a general composition structure into parallel execution paths. The experimental results showed that the proposed multi-objective ACO algorithm could find near-optimal solutions.

The authors of [34] integrated the max-min ant system into the framework of culture algorithm to solve the Web service selection problem. A comprehensive evaluation model based on generic QoS attributes and domain QoS attributes was designed. The generic QoS model was used to evaluate the QoS attributes of composite services, and the domain QoS model was used to conquer the over-constrained problem. A scenario with 10 abstract services and 50 concrete services for each abstract service was used to test the performance of the proposed algorithm. The experimental results showed that the solutions found by the proposed algorithm were much better than that of an ACO algorithm and a max-min ant system.

4.2 Genetic Algorithms

Genetic algorithms belong to the larger class of evolutionary algorithms, which generate approximate solutions to optimization and search problems by using techniques inspired by the principles of natural evolution: selection, crossover, and mutation [58]. In a GA, a population of chromosomes, which are encoded as individuals to the service concretization problem, evolves toward better solutions. Each individual is associated with a fitness value based on a fitness function that indicates how close it comes to meeting the overall specification, when compared to other individuals in the population. The fitness value of an individual is also an indication of its chances of survival and reproduction in the next generation.

When using GA to solve Web service concretization problems, the fitness function always corresponds to QoS attributes. An example of fitness function for an individual g is given by (3),

$$F(g) = \frac{w_1 * C(g) + w_2 * T(g)}{w_3 * Av(g) + w_4 * Re(g)} + w_5 * pf. \quad (3)$$

The QoS attributes ($C(g)$, $T(g)$, $Av(g)$, $Re(g)$) are normalized in the interval $[0, 1]$. The positive variables $w_1, \dots, w_5$ represent weighting factors. In particular, $w_1, \dots, w_4$ indicate the importance of each QoS attribute while w_5 weights the penalty factor pf . The penalty factor for an individual is defined as the distance between the QoS attributes of the individual and the constraints.

A typical GA requires definition of two things: one is a genetic representation of the solution domain, and the other is a fitness function to evaluate the solution domain. Once the genetic representation and the fitness function are defined, a GA executes five steps:

- 1) *Initialization.* Traditionally, the population is formed by a group of randomly generated individual solutions.
- 2) *Evaluation.* The fitness of each individual in the population is evaluated.
- 3) *Selection.* Individuals are selected based on the fitness value to breed a new generation.
- 4) *Evolution.* New individuals are created through crossover and mutation operations. The new population is composed of the individuals in the new generation and a few individuals from the previous generation.
- 5) *Termination.* Steps 2 to 4 are repeated until a termination condition has been reached.

A summary of results on applying GAs to Web service concretization problems is given in Table 3.

The study [14] proposed a GA with static and dynamic penalty strategies in the fitness function. The experimental results showed that the GA was preferred when a large number of concrete services were available for each abstract service. When the number of concrete services was in a small scale, the integer programming algorithm would be preferred.

The authors of [47] presented a GA for cost-driven Web service selection, which only considered cost of concrete services. A scenario with 20 abstract services was used to test the performance of the GA. The experimental results showed that the GA could work and get better performance when compared with a local service selection method.

The authors of [48] conducted an initial population policy and a mutation policy to direct the evolution of genetic algorithms. The performance was evaluated by comparing a GA using the relation matrix coding scheme with a GA using the one-dimension coding scheme, comparing a GA using the initial population policy with a GA using the randomly created population, and comparing a GA using the proposed mutation policy with a GA using other policies. The experimental results showed that genetic algorithms using the proposed policies could get more excellent solutions than standard genetic algorithms.

The study [49] investigated a novel tree-coding GA for QoS-aware service composition. The tree-coding GA could simultaneously express multiple types of composite relationships and could re-plan process at runtime. The performance was evaluated by comparing the tree-coding GA with a one-dimensional coding GA. The experimental results showed that the tree-coding GA was effective and faster than the one-dimensional coding GA.

The study [50] designed a repair GA to address the Web service composition problem in the presence of domain constraints and inter service dependencies. The proposed GA used a repair operator to fix up the infeasible individuals in order to make all the individuals in the population be always feasible. The repair operator was based on an iterative heuristic approach, which randomly selected a gene and randomly chose a value from

TABLE 3
A Summary of Results on Applying GAs to Web Service Concretization Problems

Articles	Fitness function used	Encoding scheme	Selection operator	Crossover operator	Mutation operator	Limitations and highlights
[14]	C, T, Re, Av, the current generation and the maximum number of generations	integer array coding	roulette wheel selection and elitism selection	two points crossover, probability was 0.7	randomly selected a gene and randomly replaced with another one, probability was 0.01	The randomly replacement in mutation operator is not efficient.
[47]	C, and a constant	integer array coding	roulette wheel selection and elitism selection	single point crossover, probability was 0.8	The probability of mutation was for the locus of chromosome, mutation probability was 0.05	It is difficult to decide the value of the constant in the fitness function. Further investigation is required to design the fitness function and other QoS attributes should be considered. The experiments was too simple.
[48]	C, T, Re, Av	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	The crossover or mutation operators may generate illegal individuals frequently, and the proposed algorithm does not treat distributed service deployment.
[49]	C, T, Re, Av, and a penalty item	tree-coding	roulette wheel selection	single point crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.01	Each gene has to record the information of its children and father, which make the length of chromosomes longer and more complicated
[50]	C, T, Re, Av, and reputation	integer array coding	rank-based selection and elitism selection	single point crossover, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.08	The repairing operator randomly selects and assigns values, the efficiency is low and it cannot guarantee of finding the feasible solution in the maximal repairing times.
[32]	C, T, Re, reputation, security, and a penalty item	integer array coding	tournament selection	two points crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.2	The hybrid genetic algorithm is based on exploration of the neighborhood, and the percentage of the neighborhood is a key factor to determine.
[33]	C, T, Re, Av, and a penalty item	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	More scenarios are required to test the proposed algorithm. The investigation is required to escape from frequently created illegal individuals by the crossover and mutation operator.
[51]	C, T, Re, Av, and reputation	did not provide	did not provide	constrained single point crossover, probability was 0.8	the selection of mutation focused on one specific gene of the entire chromosome, probability was 0.001	The proposed algorithm is required to test more scenarios and further comparisons are needed.
[18]	C, T, Re, Av, total number of constraint violations and the maximal number of possible constraint violations	integer array coding	roulette wheel selection	a knowledge-based one, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.15	The performance is not stable, further improvement is required to modify the local optimizer.
[52]	C, T, Re, Av, the reversed index number of qualitative patterns and violation values with respect to quantitative constraints	binary string coding	according to a defined probability to select individuals	uniform crossover	randomly chose a single bit of a string to mutate	The structure of execution plan is focus on sequential flow. Large number of service candidates leads to long length of the gnome, and the readability of the chromosomes is fairly weak.

the set of minimal conflict values for the selected gene. The performance was evaluated by testing the effectiveness and scalability of the proposed repair GA, and compared it with a GA without the repair operator. The experimental results showed that the repair GA was scalable and effective.

The authors of [32] proposed a hybrid GA by using a local improvement procedure, which was based on an iterative neighborhood search. The hybrid GA was compared with a standard GA. The experimental results showed that the hybrid GA performed better than the standard GA for small and medium problem sizes.

The authors of [33] adopted an enhanced initial population policy and an evolution policy to improve the convergence of the GA. An expectation value was used to control the population diversity. The performance was evaluated with regard to the coding scheme, the population diversity, the enhanced initial population, and the evolution policy. The GA with the proposed policies was compared with a standard GA. The experimental results showed that the proposed GA could improve the fitness value and promote the convergence rate.

The study [51] incorporated a GA with the rough set theory, and exemplified by solving the Web service

TABLE 4
A Summary of Results on Applying PSO Algorithms to Web Service Concretization Problems

Articles	Fitness function used	The presentation of particle	Key parameter	Limitations and highlights
[53]	did not provide	N-tuple	ω was computed according to constraints	The experiments cannot show the efficiency of the proposed algorithm, and the fitness function and some parameters are not provided.
[54]	C, T	N-dimensional vector	$\omega = 0.0275$, $c_1 = 0.3$, $c_2 = 0.49$	Further performance evaluation is required, especially various performance metrics are needed to measure performance of multi-objective optimization algorithms.
[55]	C, T, Av, and reputation	N-dimensional vector	ω , c_1 and c_2 were self-adaptive	The adaptive termination of algorithms is not considered.
[35]	C, T, Re, Av	N-dimensional vector	ω , c_1 and c_2 were self-adaptive	Further experiments are required to make sure the results are not biased by the QoS attributes of concrete services.

composition problem. The proposed method was evaluated by comparing with a standard GA and an exhaustive enumeration method. The experimental results indicated that the execution time of the proposed GA was not limited to the number of abstract services, and its convergence rate and hit rate were much better than that of the other two methods.

The study [18] considered the dependency constraint and conflict constraint between Web services in their hybrid GA. A local optimizer was used to improve the individuals in the initial population. The proposed GA was compared with a penalty-based GA and a repairing-based GA. Three sets of scenarios with different number of abstract services, concrete services, and constraints were tested. The experimental results showed that the hybrid GA was as scalable as the other two genetic algorithms, and could find better solutions.

In [52], a service composition model considering quantitative and qualitative non-functional properties was designed. A global optimization with local selection algorithm and a GA were proposed to conduct the service composition. The performance was evaluated by using a public available dataset and a synthetic dataset. The experimental results showed that the quality of the GA was better than that of the global optimization in most cases.

4.3 Particle Swarm Optimization Algorithms

Particle swarm optimization is one of the evolutionary computational techniques. It has been widely used to solve service concretization problems because it has strong robustness, a small number of parameters, and it is simple and easy to implement. In a PSO algorithm, a group of particles flying through d -dimensional search space, evolves toward optimal solutions. Each particle i represents a candidate solution to the problem, and it has a position x_{id} and a speed v_{id} . In addition, the particles have a fitness function to evaluate their best positions, which is similar to (3). The particle updates its speed and position through individual best position P_{id} and global best position P_{gd} according to (4),

$$\begin{aligned} x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1) \\ v_{id}(t+1) &= \omega v_{id}(t) + c_1 \gamma_1 (P_{id}(t) - x_{id}(t)) \\ &\quad + c_2 \gamma_2 (P_{gd}(t) - x_{id}(t)). \end{aligned} \quad (4)$$

The variable t means the t -th generation. The variable ω is the inertia weight that controls the exploration and exploitation of the search space. γ_1 and γ_2 are two mutually independent random functions. The variables c_1 (cognition coefficient) and c_2 (social coefficient) are the acceleration constants, which change the velocity of particle i towards P_{id} and P_{gd} [59]. The basic operation of a PSO algorithm is as follows.

- 1) *Initialization.* The swarm population is formed.
- 2) *Evaluation.* The fitness of each individual particle is evaluated.
- 3) *Modification.* The best position of each particle, the best position of the whole swarm and each particle's velocity are modified.
- 4) *Update.* Move each particle to a new position.
- 5) *Termination.* Steps 2 to 4 are repeated until a termination condition has been satisfied.

A summary of results on applying PSO algorithms to Web service concretization problems is given in Table 4.

The study [53] showed that the proposed multi-objective discrete PSO algorithm could get lower computation cost and higher quality solutions than an exhaustive method. The experimental results of [54] showed that their PSO algorithm used more execution time and created more Pareto solutions than the multi-objective GA.

The authors of [55] designed a non-uniform mutation strategy, an adaptive weight adjustment strategy, and a local best first strategy for a PSO algorithm. These strategies were used to enhance the population diversity and to improve the convergence rate. The performance of each strategy was evaluated. The experimental results indicated that the proposed strategies could improve the convergence rate and fitness values.

The authors of [35] proposed an immune optimization algorithm based on a PSO algorithm. An improved local best strategy and a perturbing global best strategy were also discussed. The proposed algorithm was compared with a standard PSO algorithm and a genetic algorithm. The mean and standard deviation of the optimal fitness values were computed. The experimental results showed that the proposed algorithm was better than the other two algorithms in terms of the search ability, the convergence rate, and the stability.

TABLE 5
A Summary of Results on Applying Combined Algorithms to Web Service Concretization Problems

Articles	Fitness function used	Reason(s) for combining	Limitations and highlights
[56]	did not provide details	The GA was used to search throughout the problem space, and the PSO was used to enhance local search ability.	There is a need to show more details of the experiments, and find most appropriate and robust parameters for testing.
[23]	C, T	The GA was used to set the key parameters of ACO	The experiment cannot indicate the performance of the algorithms. The QoS dataset is not provided. There is a need to design more scenarios for performance testing.
[29]	did not provide details	The swarm in stochastic PSO was often not convergent to the global optimal position but a local best position, and SA was convergent to the optimum but its convergent speed was slow.	The parameters of the proposed algorithm and the QoS dataset are not provided. Further investigations are required to test the performance of the proposed algorithms in terms of the changing number of abstract services and concrete services.

4.4 Combined Bio-Inspired Algorithms

A summary of results on applying the combined algorithms to Web service concretization problems is given in Table 5.

The study [56] designed a hybrid genetic particle swarm algorithm to balance the iterations of the global optimization and the local optimization. The proposed algorithm was compared with a balanced multi-objective GA. The experimental results indicated that the hybrid algorithm improved the solutions and enhanced the execution efficiency.

The authors of [23] presented a dynamic algorithm by combining an ACO algorithm and a GA. The proposed algorithm was compared with a GA and an immune algorithm. The experimental results showed that the proposed algorithm had better performance than the other two algorithms in terms of the optimal solutions and the iteration numbers.

The study [29] proposed a cooperative evolution algorithm that consisted of a stochastic PSO and a simulated annealing. The stochastic PSO was used to rapidly search the local optimum, while the SA was adopted to make the individual jump off the local optimum. The cooperative algorithm was compared with a stochastic PSO and a SA in terms of the quality of solution, the computation complexity, and the convergence rate. The experimental results indicated that the cooperative algorithm was better than the other two.

4.5 Summary

The ACO algorithm focuses on theoretical problems, such as parameter initialization and information updating. It can be run continuously and is generally capable of adapting to changes of an optimization problem. It may have some advantages over the PSO algorithm and the GA in dynamic environments. On the other hand, the efficiency of the ACO algorithm is somewhat low. In a GA, the mutation operator and the crossover operator are relatively fixed, and the random search without guide will cause degeneration. The ACO and PSO algorithms share many common characteristics with the GA, such as a randomly generated population, and a fitness function to evaluate individuals. Unlike a GA, however, ACO and PSO algorithms have no evolution operators such as the crossover operator and the mutation operator. All these bio-inspired algorithms have the problem of premature convergence, and they tend to be easily trapped into the local optimality.

So many efforts have been made to improve their performance by combining them.

A variety of studies suggest that the parameters of bio-inspired algorithms have a strong impact on the performance of the service concretization. They argue that inept setting of parameters may result in bad performance, but only a few of the existing studies [21], [34], [35], [55] gave an explicit investigation about the influence of parameters. For example, the design of a GA has a great influence on its behavior and performance [62]. It is necessary for a GA to accord with the characteristic of the service concretization in order to get better performance. By reviewing a number of studies on bio-inspired algorithms-based Web service concretization, we summarize the main control parameters involved in each type of algorithms in Table 6. The Table also indicates the effects of these parameters to the performance of the algorithms. It should be pointed out that the effects of the parameters to the performance of the bio-inspired algorithms are not yet well studied. And different problems and contexts may set different values. The settings of the parameters of the bio-inspired algorithms result in varying degrees of performance, and all the factors contribute to the bio-inspired algorithms' performance. This makes it difficult to choose the operators and set the values in order to yield optimum performance.

Normally, the Web service concretization needs to be performed and completed in a short time period, and the quality of solutions needs to fully satisfy service requesters' requirements. Many studies into Web service concretization based on bio-inspired algorithms are attempting to reduce processing time for a near optimum solution, to improve the quality of solutions, and in particular, to avoid being trapped into local optimization. Some studies have carried out comparisons among different versions of the same type of algorithms, such as comparing an ACO algorithm with a chaos ACO algorithm [46], or comparing an improved GA with a standard GA [18], [32], [33], [48], [49], [50], [51]. Other studies have compared different types of algorithms, such as comparing the combination of an ACO algorithm and a GA with a standard GA [23].

Furthermore, it is necessary to establish a standard framework for designing and testing bio-inspired Web service concretization. Currently, there is no such framework available. The existing studies presenting comparisons

TABLE 6
Main Control Parameters in the Three Types of Bio-Inspired Algorithms

Algorithm	Main control parameters	Effects to the performance of the algorithm
ACO	number of ants	The larger the number of ants, the better the convergence rate, and this leads to longer simulation times.
	influence of heuristic (β)	If $\beta = 0$, only the pheromone information is used, which generally leads to rather poor results [57]. The larger values lead to suboptimal solutions.
	influence of pheromone (α)	If $\alpha = 0$, the services with best QoS attributes are more likely to be selected: this corresponds to a stochastic greedy algorithm. The smaller values of α lead to the slow convergence and local optimum, and the larger values lead to a strong emphasis of initial, random fluctuations and bad algorithm behavior [57].
	pheromone evaporation coefficient (ρ)	The larger values affect the global search ability, and the smaller values slow the convergence speed.
	population size	The increase of the population size will increase the number of generations to convergence, improve the solution accuracy, and increase the computational time. Moreover, there is an optimal population size for each given problem, and further increase of the population size has no benefit [60].
GA	selection operator	The three common selection operators are the roulette wheel selection, the rank-based selection, and the tournament selection. On the average, the tournament selection is the most effective and efficient operator both in quality and computation time, and the rank-based selection is the worst one. When the problem size increases, the solution quality of the roulette wheel selection and the tournament selection becomes worse.
	crossover operator	The three common crossover operators are one-point, two-point, and uniform crossover. It is difficult to decide which form of crossover will give optimum performance. Usually, it is decided by an empirical study.
	crossover probability	The smaller the crossover probability, the more individuals survive in the next generation unchanged. Usually, the probability of crossover is set by means of trial and error for a given problem.
	mutation operator	The mutation operator can be operated on a randomly selected gene in each chromosome, or each gene in a randomly selected chromosome, or one specific gene of the entire chromosome. These three methods were compared in the study [48].
	mutation probability	The smaller mutation probability may lead to local optima, and the larger mutation probability increases the search areas, however, prevents the GA to find any optima solution.
PSO	population size	The literature suggested that the population size in PSO as $2n$ to $5n$, where n is the number of decision variables [61]. And it depends on the particular version of PSO used, the number of variables, and the complexity of the search space.
	inertia weight (ω)	The inertia weight is added to the standard PSO algorithm in order to improve the convergence rate and balance the exploration and exploitation of search space.
	acceleration factor (c_1 and c_2)	The smaller values of the acceleration factors, the more generations are needed to reach the optima. The larger values will prevent the algorithm from finding the optima.

among bio-inspired algorithms had limited themselves to their own simulated environment and test data set. Their experimental results represented only a small number of specific scenarios. It is difficult to tell which algorithm performs better in a certain scenario, and which value is better for the key parameter. Guidelines are also needed for service developers to choose the optimal algorithm to offer composite services, which could enable the service requesters to get better performance based on their own requirements and preferences.

5 UNDERLYING APPLICATIONS

Due to the deluge of enormous sources of data and the development of cloud computing, applications based on data-intensive services have become one of the most challenging applications in service oriented computing. Data access, data analysis, and data manipulation may require a

nontrivial composition of a series of data retrieval and process executions. The service-based strategy provides maximal flexibility when designing data-intensive applications. These data services are different from traditional services because they handle huge amounts of data, and their QoS attributes might be quite different. Bio-inspired computing is one of the ongoing or underlying techniques and technologies to harness the data-intensive service concretization problems. The authors of [63] stated that biological inspired systems were better appropriate for the data-intensive problems because they were efficient to organize, access, and process data.

5.1 Current Work

The data-intensive service concretization problem with global QoS constraints is an extension of the traditional service concretization problem, in which data sets, as the inputs and outputs of services, are incorporated. Services

use data from data providers, and they also use data from other services. In general, data-intensive service concretization will be cooperatively supported by three stakeholders: the service composers, the service providers, and the data providers [64]. Providers need an approach to regulate and price their resources, either services or data or their combination. They all want to have a good market position maximizing their profits. It assumes that an economic model is an accurate representation of the reality, and it will offer a suitable way to regulate the interactions among the three stakeholders.

We have been applying bio-inspired algorithms to tackle the data-intensive service concretization problems [65], [66], [67]. A graph for the data-intensive service concretization problem was given in [64]. The lifetime of the problem framework was described in [68]. In the lifetime, the first step is that the service composer tries to select a set of service candidates while the data provider provides data sets, and the second step is that if a feasible solution which satisfies the service composer's local and global QoS constraints does not exist, negotiations are performed in order to determine new quality values for each involved service. An extensible QoS model was proposed in [67]. In the model, the cost for a data-intensive service includes the access cost and the transfer cost of all required data sets, and the estimated execution time for a data-intensive service includes the time for processing data sets and the time for accessing data sets. An enhanced ACO algorithm and a modified GA were designed to solve the data-intensive service concretization problem [69], [70]. We also designed strategies to adapt an ant colony system to handle the dynamic scenarios [71]. Meanwhile, data-intensive services are used in a dynamic and changing environment, and different providers typically have conflicting objectives [72], [73]. In order to automate the process of reaching an agreement in the data-intensive service concretization problem, we proposed an ant-inspired negotiation approach [68], [74]. In addition, we proposed a multi-objective ant colony system for data-intensive service concretization [75], where we focused on the scalability and adaptability of service concretization, and particular attention is paid to multi-objective optimization related to cost and QoS attributes. Summary of our current work can be found in [76].

5.2 Remaining Problems

The data-intensive service provision problem will become more intense and more complex, as a result of the enormous proliferation of data-intensive services, for example, in critical areas such as disaster management and health care. To solve the problem, it is necessary to investigate the approaches to the traditional service concretization problems. By analyzing methods in the two types of Web service concretization approaches, we found bio-inspired algorithms for QoS-aware service composition usually required much less computation time for large scale problems, especially in the dynamic environments. The findings from the systematic review had been the basis for solving the problems. There are also a few other studies on data-intensive service composition and data placement [77], [78], [79]. Our current work and other studies indicate that we should

attempt to find approaches to supporting economic data-intensive services provision from holistic perspectives.

One of the remaining problems is how to place the data. As we mentioned in the above section, the data provider sells data sets to multiple service providers in order to maximize the data usage and the profit. The data provider may improve the utility of data by changing data placement policies. The data placement policies are about how data can be organized physically in order to minimize the cost and access latency of data sets. For example, the data providers may place data near the services or partition a large amount of data sets into several portions (or replicas if necessary). The data placement problem is very similar to the minimum K-median problem, especially from a facility location perspective [80]. By investigating the data placement policies, the data providers could lower the total cost of the composite service.

Another remaining problem is how to create an economic model. The economic model described in our earlier work [64] may be extended to include the network providers, since the cost of transferring data and the response time of accessing data depend on the network bandwidth. Various providers need a standardized yet adaptive way to regulate and price their resources, which are specified by the utility of the stakeholders. Hence, the composite service should be constructed by achieving the Pareto optimum under the Nash equilibrium for the data-provider utility, the network-provider utility, the service-provider utility, and the service-composer utility.

6 CONCLUSION

In this paper, the Web service concretization based on bio-inspired algorithms is discussed. The existing Web service concretization approaches can be categorized into two groups: local optimization approaches and global optimization approaches. The local optimization approaches cannot satisfy the global QoS constraints, or they can be used by exploiting decomposition methods. The global optimization approaches construct composite services from a global level to satisfy the QoS constraints. Accordingly, several optimal and sub-optimal algorithms have been proposed, and in particular, metaheuristics-based algorithms intend to find near-optimal solutions. The sub-optimal algorithms demonstrate good scalability and can decrease the computational complexity when compared with the optimal algorithms.

The applications of ant colony optimization algorithms, genetic algorithms, and particle swarm optimization algorithms to Web service concretization have been described in detail. We also presented many studies that had applied bio-inspired algorithms to solve data-intensive service concretization problems. While some progresses have been made in recent years in the area of bio-inspired Web service concretization, there remains much potential for further efforts to be made in this area, especially for the data-intensive problems.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the help of Dr. Madeleine Cincotta in the final language editing of this paper.

REFERENCES

- [1] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.
- [2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, May 2007.
- [3] G. Canfora, M. Penta, R. Esposito, and M. Villani, "A lightweight approach for qos aware service composition," in *Proc. 2nd Int. Conf. Serv. Oriented Comput.*, 2004, pp. 36–47.
- [4] L. Wang and J. Shen, "Towards bio-inspired cost minimisation for data-intensive service provision," in *Proc. IEEE Int. Conf. Serv. Econ.*, 2012, pp. 16–23.
- [5] J. Rao and X. Su, "A survey of automated web service composition methods," in *Proc. Semantic Web Serv. Web Process Composition*, 2005, vol. 3387, pp. 43–54.
- [6] S. Dustdar and W. Schreiner, "A survey on web services composition," *Int. J. Web Grid Serv.*, vol. 1, no. 1, pp. 1–30, Aug. 2005.
- [7] Y. Syu, S. Ma, J. Kuo, and Y. FanJiang, "A survey on automated service composition methods and related techniques," in *Proc. IEEE 9th Int. Conf. Serv. Comput.*, 2012, pp. 290–297.
- [8] F. Li, F. Yang, K. Shuang, and S. Su, "Q-peer: A decentralized qos registry architecture for web services," in *Proc. Serv.-Oriented Comput.*, 2007, vol. 4749, pp. 145–156.
- [9] C. Patel, K. Supekar, and Y. Lee, "A qos oriented framework for adaptive management of web service based workflows," in *Proc. Database Expert Syst. Appl.*, 2003, vol. 2736, pp. 826–835.
- [10] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.
- [11] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end qos constraints," *ACM Trans. Web*, vol. 6, no. 2, pp. 7:1–7:31, 2012.
- [12] F. Mardukhi, N. Nematbakhsh, K. Zamanifar, and A. Barati, "Qos decomposition for service composition using genetic algorithm," *Appl. Soft Comput.*, vol. 13, no. 7, pp. 3409–3421, Feb. 2013.
- [13] S. Sun and J. Zhao, "A decomposition-based approach for service composition with global qos guarantees," *Inf. Sci.*, vol. 199, pp. 138–153, Mar. 2012.
- [14] G. Canfora, M. Penta, R. Esposito, and M. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proc. Conf. Genetic Evol. Comput.*, 2005, pp. 1069–1075.
- [15] M. Jaeger and G. Muehl, "Qos-based selection of services: The implementation of a genetic algorithm," in *Proc. KiVS 2007 Workshop: Serv.-Oriented Archit. Serv.-Oriented Comput.*, 2007, pp. 359–370.
- [16] M. Jaeger, G. Muehl, and S. Golze, "Qos-aware composition of web services: An evaluation of selection algorithms," in *Proc. Move Meaningful Internet Syst. 2005: CoopIS, DOA, and ODBASE*, 2005, vol. 3760, pp. 646–661.
- [17] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. Web*, vol. 1, no. 1, pp. 1–26, May 2007.
- [18] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in *Proc. World Congress Comput. Intell.*, 2010, pp. 268–275.
- [19] M. Zemni, S. Benbernou, and M. Carro, "A soft constraint-based approach to qos-aware service selection," in *Proc. Serv.-Oriented Comput.*, 2010, vol. 6470, pp. 596–602.
- [20] L. Li, P. Yang, L. Ou, Z. Zhang, and P. Cheng, "Genetic algorithm-based multi-objective optimization for qos-aware web services composition," in *Proc. Knowl. Sci., Eng. Manage.*, 2010, vol. 6291, pp. 549–554.
- [21] W. Zhang, C. Chang, T. Feng, and H. Jiang, "Qos-based dynamic web service composition with ant colony optimization," in *Proc. 34th Annu. IEEE Int. Comput. Softw. Appl.*, 2010, pp. 493–502.
- [22] C. Pop, V. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoae, "Ant-inspired technique for automatic web service composition and selection," in *Proc. 12th Int. Symp. Symbolic Numeric Algorithms Scientific Comput.*, 2010, pp. 449–455.
- [23] Z. Yang, C. Shang, Q. Liu, and C. Zhao, "A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm," *Comput. Inf. Syst.*, vol. 6, no. 8, pp. 2617–2622, Aug. 2010.
- [24] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven web service composition in meteor-s," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2004, pp. 23–30.
- [25] V. Cardellini, E. Casalicchio, V. Grassi, and F. Presti, "Efficient provisioning of service level agreements for service oriented applications," in *Proc. 2nd Int. Workshop Serv. Oriented Softw. Eng.: Conjunction with 6th ESEC/FSE Joint Meeting*, 2007, pp. 29–35.
- [26] I. Estévez-Ayres, P. Basanta-Val, M. Garcia-Valls, J. Fisteus, and L. Almeida, "Qos-aware real-time composition algorithms for service-based applications," *IEEE Trans. Indus. Informat.*, vol. 5, no. 3, pp. 278–288, Aug. 2009.
- [27] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán, and M. Toro, "Improving the automatic procurement of web services using constraint programming," *Int. J. Cooperative Inf. Syst.*, vol. 14, no. 4, pp. 439–467, Dec. 2005.
- [28] D. Ivanović, M. Carro, and M. Hermenegildo, "A constraint-based approach to quality assurance in service choreographies," in *Proc. 10th Int. Conf. Serv.-Oriented Comput.*, 2012, pp. 252–267.
- [29] X. Fan, X. Fang, and C. Jiang, "Research on web service selection based on cooperative evolution," *Expert Syst. with Appl.*, vol. 38, no. 8, pp. 9736–9743, Aug. 2011.
- [30] P. Bless, D. Klabjan, and S. Chang, "Heuristics for automated knowledge source integration and service composition," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1292–1314, Apr. 2008.
- [31] N. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "Qos-aware service composition in dynamic service oriented environments," in *Proc. Middleware*, 2009, vol. 5896, pp. 123–142.
- [32] J. Parejo, P. Fernandez, and A. Cortes, "Qos-aware services composition using tabu search and hybrid genetic algorithms," *Actas de Talleres de Ingeniera Del Software y Bases de Datos*, vol. 2, no. 1, pp. 55–66, 2008.
- [33] Y. Ma and C. Zhang, "Quick convergence of genetic algorithm for qos-driven web service selection," *Comput. Netw.*, vol. 52, no. 5, pp. 1093–1104, 2008.
- [34] Z. Wang, Z. Liu, X. Zhou, and Y. Lou, "An approach for composite web service selection based on dgqos," *Int. J. Adv. Manufacturing Technol.*, vol. 56, no. 9–12, pp. 1167–1179, Oct. 2011.
- [35] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan, "An improved discrete immune optimization algorithm based on pso for qos-driven web service composition," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2208–2216, Aug. 2012.
- [36] Z. Kobti and Z. Wang, "An adaptive approach for qos-aware web service composition using cultural algorithms," in *Proc. Adv. Artif. Intell.*, 2007, vol. 4830, pp. 140–149.
- [37] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in *Proc. Int. Conf. Web Serv.*, 2006, pp. 72–82.
- [38] M. Jäger, "Optimising quality-of-service for the composition of electronic services," Ph.D. dissertation, Berlin Univ. of Technology, Berlin, Germany, Jan. 2007.
- [39] Y. Guan, A. K. Ghose, and Z. Lu, "Using constraint hierarchies to support qos-guided service composition," in *Proc. IEEE Int. Conf. Web Serv.*, 2006, pp. 743–752.
- [40] B. Pernici, S. Siadat, S. Benbernou, and M. Ouziri, "A penalty-based approach for qos dissatisfaction using fuzzy rules," in *Proc. Serv.-Oriented Comput.*, 2011, vol. 7084, pp. 574–581.
- [41] S. Balasubramaniam, D. Botvich, R. Carroll, J. Mineraud, T. Nakano, T. Suda, and W. Donnelly, "Biologically inspired future service environment," *Comput. Netw.*, vol. 55, no. 15, pp. 3423–3440, 2011.
- [42] T. Nakano, "Biologically inspired network systems: A review and future prospects," *IEEE Trans. Syst., Man, Cybern.-Part C: Appl. Rev.*, vol. 41, no. 5, pp. 630–643, Sep. 2011.
- [43] B. Kitchenham. (2007). Guidelines for performing systematic literature reviews in software engineering, Keele University and Durham University Joint Report, Tech. Rep. EBSE-2007-001 [Online]. Available: <http://www.dur.ac.uk/ebse/resources/>
- [44] X. Zheng, J. Luo, and A. Song, "Ant colony system based algorithm for qos-aware web service selection," in *Proc. 4th Int. Conf. Grid Serv. Eng. Manage.*, 2007, pp. 39–50.
- [45] Y. Xia, J. Chen, and X. Meng, "On the dynamic ant colony algorithm optimization based on multi-pheromones," in *Proc. 7th IEEEACIS Int. Conf. Comput. Inf. Sci.*, 2008, pp. 630–635.
- [46] L. Wang and Y. He, "A web service composition algorithm based on global QOS optimizing with mocao," in *Proc. Int. Conf. Inform. Cybern. Comput. Eng.*, 2010, vol. 111, pp. 79–86.

- [47] L. Cao, M. Li, and J. Cao, "Cost-driven web service selection using genetic algorithm," in *Proc. Internet Netw. Econ.*, 2005, vol. 3828, pp. 906–915.
- [48] C. Zhang, S. Su, and J. Chen, "A novel genetic algorithm for qos-aware web services selection," in *Proc. Data Eng. Issues E-Commerce Serv.*, 2006, vol. 4055, pp. 224–235.
- [49] C. Gao, M. Cai, and H. Chen, "Qos-aware service composition based on tree-coded genetic algorithm," in *Proc. 31st Annu. Int. Comput. Softw. Appl. Conf.*, 2007, pp. 361–367.
- [50] L. Ai and M. Tang, "Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm," in *Proc. IEEE 4th Int. Conf. eScience*, 2008, pp. 119–126.
- [51] W. Liang and C. Huang, "The generic genetic algorithm incorporates with rough set theory - an application of the web services composition," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5549–5556, Apr. 2009.
- [52] H. Wang, P. Ma, and X. Zhou, "A quantitative and qualitative approach for nfp-aware web service composition," in *Proc. IEEE 9th Int. Conf. Serv. Comput.*, 2012, pp. 202–209.
- [53] X. Fan, C. Jiang, and X. Fang, "An efficient approach to web service selection," in *Proc. Web Inf. Syst. Mining*, 2009, vol. 5854, pp. 271–280.
- [54] H. Rezaie, N. NematBaksh, and F. Mardukhi, "A multi-objective particle swarm optimization for web service composition," in *Proc. Netw. Digital Technol.*, 2010, vol. 88, pp. 112–122.
- [55] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication," *Int. J. Comput. Intell. Syst.*, vol. 3, no. 01, pp. 18–30, Mar. 2012.
- [56] J. Liu, J. Li, K. Liu, and W. Wei, "A hybrid genetic and particle swarm algorithm for service composition," in *Proc. 6th Int. Conf. Adv. Lang. Process. Web Inf. Technol.*, 2007, pp. 564–567.
- [57] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.
- [58] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.
- [59] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.
- [60] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, 2013, pp. 371–376.
- [61] M. Pant, R. Thangaraj, and A. Abraham. (2009). Particle swarm optimization: Performance tuning and empirical analysis [Online]. available: <http://www.softcomputing.net/psf-foci-chapter.pdf>
- [62] I. Rojas, J. Gonzalez, H. Pomares, J. J. Merelo, P. A. Castillo, and G. Romero, "Statistical analysis of the main parameters involved in the design of a genetic algorithm," *IEEE Trans. Syst., Man, Cybern.-Part. C: Appl. Rev.*, vol. 32, no. 1, pp. 31–37, Feb. 2002.
- [63] C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Inf. Sci.*, vol. 275, pp. 314–347, 2014.
- [64] L. Wang, J. Shen, J. Luo, and F. Dong, "An improved genetic algorithm for cost-effective data-intensive service composition," in *Proc. 9th Int. Conf. Semantics, Knowl. Grids*, 2013, pp. 105–112.
- [65] J. Shen, G. Beydoun, S. Yuan, and G. Low, "Comparison of bio-inspired algorithms for peer selection in services composition," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2011, pp. 250–257.
- [66] J. Shen and S. Yuan, "Qos-aware peer services selection using ant colony optimisation," in *Proc. Bus. Inf. Syst. Workshops*, 2009, vol. 37, pp. 362–374.
- [67] L. Wang, J. Shen, C. Di, Y. Li, and Q. Zhou, "Towards minimizing cost for composite data intensive services," in *Proc. IEEE 17th Int. Conf. Comput. Supported Cooperative Work Des.*, 2013, pp. 293–298.
- [68] L. Wang and J. Shen, "Multi-phase ant colony system for multi-party data-intensive service provision," *IEEE Trans. Serv. Comput.*, 2014, Doi: 10.1109/TSC.2014.2358213.
- [69] L. Wang, J. Shen, and G. Beydoun, "Enhanced ant colony algorithm for cost-aware data-intensive service provision," in *Proc. IEEE 9th World Congress Serv.*, 2013, pp. 227–234.
- [70] L. Wang and J. Shen, "Economical data-intensive service provision supported with a modified genetic algorithm," in *Proc. IEEE 2nd Int. Congress Big Data*, 2013, pp. 358–365.
- [71] L. Wang, J. Shen, and J. Luo, "Impacts of pheromone modification strategies in ant colony for data-intensive service provision," in *Proc. IEEE 21th Int. Conf. Web Serv.*, 2014, pp. 177–184.
- [72] A. Ismail, J. Yan, and J. Shen, "Dynamic service selection for service composition with time constraints," in *Proc. Australian Softw. Eng. Conf.*, 2009, pp. 183–190.
- [73] A. Ismail, J. Yan, and J. Shen, "An offer oeneration approach to sla negotiation support in service oriented computing," *Serv. Oriented Comput. Appl.*, vol. 4, no. 4, pp. 277–289, 2010.
- [74] L. Wang, J. Shen, Q. Zhou, and G. Beydoun, "Ant-inspired multi-phase and multi-party negotiation in the data-intensive service provision," in *Proc. IEEE 11th Int. Conf. Serv. Comput.*, 2014, pp. 211–218.
- [75] L. Wang, J. Shen, and J. Luo, "Facilitating an ant colony algorithm for multi-objective data-intensive service provision," *J. Comput. Syst. Sci.*, vol. 81, pp. 734–746, 2014.
- [76] L. Wang, J. Shen, and J. Luo, "Bio-inspired cost-aware optimization for data-intensive service provision," *Concurrency Comput.: Practice Exp.*, vol. 27, no. 18, pp. 5662–5685, 2015.
- [77] S. Deng, L. Huang, Y. Li, and J. Yin, "Deploying data-intensive service composition with a negative selection algorithm," *Int. J. Web Serv. Res.*, vol. 11, no. 1, pp. 76–93, 2014.
- [78] X. Li, Y. Wu, F. Ma, E. Zhu, F. Wang, L. Wu, and Y. Yang, "A new particle swarm optimization-based strategy for cost-effective data placement in scientific cloud workflows," in *Proc. Future Inf. Technol.*, 2014, vol. 309, pp. 115–120.
- [79] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A data placement strategy in scientific cloud workflows," *Future Gener. Comput. Syst.*, vol. 26, pp. 1200–1214, 2010.
- [80] J. Byrka, "An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem," in *Proc. Approximation, Randomization, Combinatorial Optimization. Algorithms Techn.*, 2007, vol. 4627, pp. 29–43.

Lijuan Wang received the PhD degree in 2014 from the University of Wollongong. She is a lecturer at Xidian University in Xi'an, China. She has published 16 papers in journals and conferences in computer science. Her interests are on neural network, bio-inspired computing, and data security.

Jun Shen received the PhD degree in 2001 from Southeast University, China. He is an associate professor at the University of Wollongong in Wollongong, NSW, Australia. He has published more than 100 papers in journals and conferences in computer science and information system areas. His expertise include Web services and e-learning systems as well as big data. He is a senior member of the IEEE, ACM, and ACS. He is a member of IS curriculum review task force sponsored by ACM/AIS.