1-1-2015

Bio-inspired cost-aware optimization for data-intensive service provision

Lijuan Wang
Xidian University, lw840@uowmail.edu.au

Jun Shen
University of Wollongong, jshen@uow.edu.au

Junzhou Luo
Southeast University

Bio-inspired cost-aware optimization for data-intensive service provision

Abstract

The use of Big Data and the development of cloud computing have led to greater dependence on data-intensive services. Each service may actually request or create a large amount of data sets. The scope, number, and complexity of data-intensive services are all set to soar in the future. To compose these services will be more challenging. Issues of autonomy, scalability, adaptability, and robustness, become difficult to resolve. Bio-inspired algorithms can overcome the new challenging requirements of data-intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems and accordingly to add biological mechanisms to services. In this paper, we will discuss single-objective and multi-objective data-intensive service provision problems based on bio-inspired algorithms. Further, we will propose an ant-inspired negotiation approach. Finally, this paper points out future research topics.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Wang, L., Shen, J. & Luo, J. (2015). Bio-inspired cost-aware optimization for data-intensive service provision. Concurrency and Computation: Practice and Experience, 27 (18), 5662-5685.

Bio-Inspired Cost-Aware Optimization for Data-Intensive Service Provision

Lijuan Wang
School of Cyber Engineering,Xidian University, Xi'an, P. R. China
Jun Shen
School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia
Junzhou Luo
School of Computer Science and Engineering, Southeast University, Nanjing, P. R. China

Abstract

The use of Big Data and the development of cloud computing have led to greater dependence on data-intensive services. Each service may actually request or create a large amount of data sets. The scope, number, and complexity of data-intensive services are all set to soar in the future. To compose these services will be more challenging. Issues of autonomy, scalability, adaptability, and robustness, become difficult to resolve. Bio-inspired algorithms can overcome the new challenging requirements of data-intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems and accordingly to add biological mechanisms to services. In this paper, we will discuss single-objective and multi-objective data-intensive service provision problems based on bio-inspired algorithms. Further, we will propose an ant-inspired negotiation approach. Finally, this paper points out future research topics.

I. Introduction

Big Data has attracted much research attention. The Gartner Group listed Big Data in the "10 Critical Tech Trends for the Next Five Years" [1]. Big Data can generate value in every domain and sector. For example, if officials in our public health agency are to use Big Data creatively and effectively, they could get more useful information and the sector could save many lives. The data generated by scientific activities, social networking, social media, as well as commercial applications have increased exponentially. Data-intensive science is emerging as the fourth scientific paradigm, and new techniques and technologies for the new scientific paradigm are needed [2]. As a result, applications based on data-intensive services have become one of the most challenging applications in service oriented computing and cloud computing [3], [4]. A survey of the challenges, techniques, and technologies of data-intensive applications was presented in [5]. The scope, number, and complexity of data-intensive services are all set to soar in the future. On the one hand, Big Data provides opportunities and potential values. On the other hand, many challenges are arising with respect to the data capture, data storage, data analysis, data searching, data sharing, and data visualization [5]. The service provision, and in particular service composition, will face new challenges such as autonomy, scalability, adaptability, and robustness. Indeed, new mechanisms are needed to overcome those issues. In the following, we will briefly introduce the Big Data problem in scientific research fields.

One of the motivations of our work is the Alpha Magnetic Spectrometer (AMS) experiment, which uses cloud computing to process a huge amount of data. The AMS, also designated AMS-02, is a particle physics experiment module that is mounted on the International Space Station. The purpose of the AMS experiment is to advance knowledge of the universe and lead to the understanding of its origin by searching for antimatter and dark matter while performing precision measurements of cosmic rays composition and flux. The ground AMS lab is based at CERN[1] in Switzerland. The key technology for accessing the data collected from AMS relies on data services based on cloud computing. The AMS-02 SOC (Science Operation Center) at Southeast University in China (labeled as AMS-02 SOC@SEU) is supported by the IBM-sponsored Cloud Computing Center with 3500 CPU core and 500TB storage. The AMS-02 SOC@SEU typically receives 200GB of data from AMS and generates 700GB of data after processing them, on each day. Scientists and remote users deploy different processes, such as data mining, image processing, thematic map generation, or data query on a large amount of data at AMS-02 SOC. A set of operations is often necessary to provide an appropriate solution to complex scientific applications. The use of Web services technologies provides valuable solutions to speed up the scientific data analysis [6], [7]. A composition of a set of services as a composite service can be reused by other researchers. For science problems involving a large amount of data, cost-effective mechanisms for data-intensive service provision are needed.

The authors of [5] explained that bio-inspired computing was one of the underlying techniques to solve data-intensive problems. The authors stated that biological computing models were better appropriate for Big Data because they had

[1]http://home.web.cern.ch/about/experiments/ams

mechanisms with high-efficiency to organize, access, and process data. The authors of [8] already proved that it was useful for service management and discovery to add biological mechanisms to services. In this paper, we introduce the comprehensive applications of an ant colony system (ACS) and a genetic algorithm (GA) for cost-aware data-intensive service provision. We propose an economic model. Further, we investigate an ACS and a GA for the multi-objective data-intensive service provision. Taking further enhancements in the economic model, this paper proposes an ant-inspired negotiation approach. Finally, this paper presents an overview of some of remaining problems and opportunities for future work.

The remainder of this paper is organized as follows. Section II introduces background. Section III investigates how an ACS and a GA could be used to solve the cost-aware data-intensive service provision problems. Section IV investigates how an ACS and GA could be used to solve the multi-objective data-intensive service provision problem. Section V presents an ant-inspired negotiation approach for the problem. Section VI points out future research topics. Finally, section VII concludes this paper.

II. Background

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. The implementation of a basic service refers to the simple interactions between a client and a server. If the implementation of a service invokes other services, it is necessary to combine the functionalities of several services and this service is referred to as a composite service [9]. The process of developing a composite service in turn is called service composition [10]. 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. These concrete 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. 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 [11], [12]. 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.

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. The local optimization approaches are not suitable for QoS-based service selection with global QoS constraints, since they can only guarantee local QoS constraints and cannot satisfy the global QoS constraints. From the viewpoint of computational time, the local optimization approaches can be appropriate when the global QoS constraints are transformed into local QoS constraints. The global optimization approaches can solve service selection problem at both local and global service levels. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. These methods should be evaluated with respect to their optimality, their computational efficiency, and their dynamic complexity. The detailed analysis of each method indicated that the bio-inspired algorithms, belonging to the sub-optimal methods, could overcome the new challenging requirements of the data-intensive service provision problem. Then we conducted a systematic review of Web service composition and

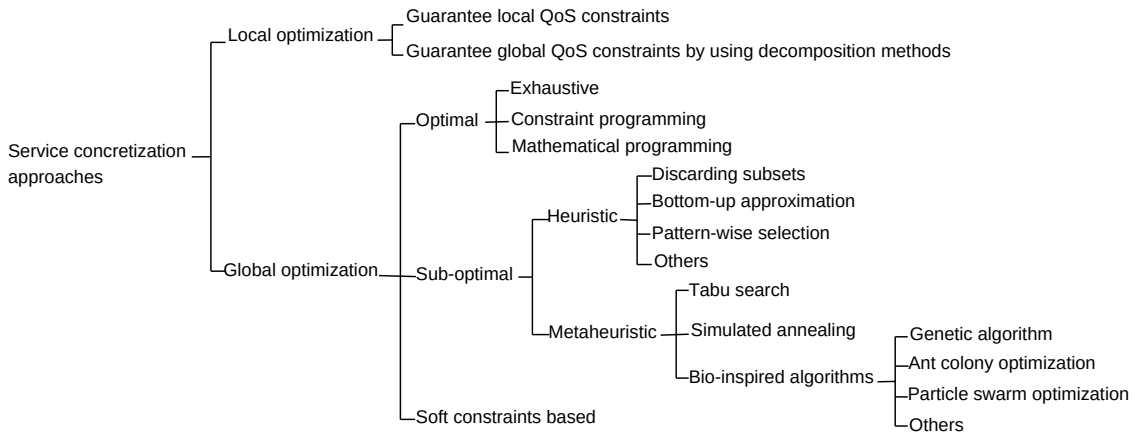


Fig. 1: A hierarchical taxonomy of Web service concretization approaches

selection based on three bio-inspired algorithms, namely, the ant colony optimization (ACO) algorithms, the genetic algorithms, and the particle swarm optimization (PSO) algorithms [13]. The systematic review based on the three bio-inspired algorithms helps us re-examine the models, the basic operational flow, the main issues, and the limits of each algorithm. In the following, we will briefly introduce the three bio-inspired algorithms.

A. Ant Colony Optimization Algorithms

The ant colony optimization algorithms are inspired by the foraging behavior of ant colonies, in which a set of artificial ants cooperate to find a solution of a problem by exchanging information via pheromone deposited on a graph edges. When ACO algorithms are used to solve a Web service concretization problem, the problem is modeled as a graph. The ACO algorithms iteratively perform a loop constitutes the ants' solution construction and the pheromone update. The ant colony system is an algorithm inspired by the ant system but differs from it in three main aspects [14]. First, the state transition rule provides a way to balance between the exploration of new edges and the exploitation of accumulated knowledge about the problem. Second, a local updating rule is applied while ants construct a path. Third, the global updating rule is applied only to edges which belong to the best ant path. Thus, we adopted the ACS in our paper.

Many studies have applied ACO algorithms to solve the Web service concretization problems. The study [15] 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. The study [16] 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 [17] 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.

B. 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 [18]. 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.

The authors of [19] conducted an initial population policy and a mutation policy to direct the evolution of genetic algorithms. The study [20] 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 study [21] designed a repair GA to address the Web service composition problem in the presence of domain constraints and inter service dependencies. The authors of [22] adopted an enhanced initial population policy and an evolution policy to improve the convergence of the GA. The performance was evaluated with regard to the coding scheme, the population diversity, the enhanced initial population, and the evolution policy. The study [23] considered the dependency constraint and conflict constraint between Web services in their hybrid GA.

C. 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 represents a candidate solution to the problem, and it has a position and a speed. In addition, the particles have a fitness function to evaluate their best positions.

The study [24] compared a multi-objective discrete PSO algorithm with an exhaustive method. The experimental results showed that the PSO algorithm could get lower computation cost and higher quality solutions. The authors of [25] 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 authors of [26] 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.

III. Cost-Aware Data-Intensive Service Provision

This section introduces the comprehensive applications of an ACS and a GA to the data-intensive service provision from the perspective of costs. Due to the explosion in digital data, the distributed nature of cloud computing, as well as the large increase of providers to the market, providing efficient cost models for composing data-intensive services will become central to this dynamic market. The location of users, service composers, service providers, and data providers will affect the total cost of service provision. Different providers need to make decisions about how to price and pay for resources. Each of them wants to have a good market position maximizing profit. An economic model is proposed for the data-intensive service provision.

A. An Economic Model

In general, data-intensive service provision will be cooperatively supported by four stakeholders: the data/service users, the service composers, the service providers, and the data providers. 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 four stakeholders. As shown in Fig. 2, in the downstream market, the data/service users require data-intensive services from the service composers, and the service composers seek optimal strategies to select elementary services provided by multiple service providers who compete on the basis of price and QoS attributes. In the upstream market, the service providers request the data from the data providers. The data/service users specify their requirements of the data-intensive services. From the service composers' point of view, it is important to be able to assess the value of the needed services and how much they want to pay for them to satisfy the data/service users' requirements as well as to maximize their profit. From the service providers' perspective, it is important to be able to analyze their competitors' position and improve their offers if they are to win contracts with the service composers. The price of the data may affect the total cost and the price of data-intensive services. Therefore, the prices of both service and data have a crucial impact on the service composers' and the service providers' profits.

Here we make a distinction between cost and other QoS attributes because cost is usually related to other quality attributes and it becomes more important in the data-intensive service provision. In the traditional service composition, executable services and their input or output data are usually on the same platform. Thus, the cost for data storage can be neglected or the cost is a constant determined before execution, and service selection algorithms need not consider it. However, in data-intensive service composition, service providers charge service requesters depending on the requesters' location and the amount of transferred data [27]. Each service requests data sets from the storage resources (or data servers). Each of these data sets may be replicated at several locations that are connected to each other and to the service platform through networks of varying capability [28]–[30]. When composing data-intensive services, optimizing the cost of data is a priority, as data plays the dominant role. In one of our earlier studies [31], we have introduced an extensible QoS model. In the model, we considered the price of data set, the size of data set, the access cost and the transfer cost of all data sets required by each service.

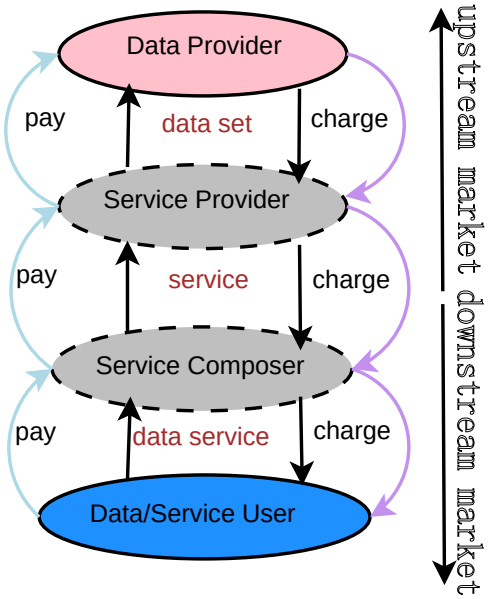


Fig. 2: Service and data set usage and charging relationship

B. Data-Intensive Service Provision Based on an ACS

As mentioned in subsection II-A, the key to ACS is how to determine the state transition rule, the local updating rule, and the global updating rule.

1) State Transition Rule: When ant k arrives at vertex i, it will choose successor j to move to by applying the rule given by (1).

$$j = \begin{cases} \arg\max_{j \in N_i^k} \{[\tau_{ij}][\eta_{ij}]^\beta\}, & \text{if } q \leq q_0; \\ \text{randomly selected from } N_i^k \text{ according to } p_{ij}^k, & \text{otherwise.} \end{cases} \tag{1}$$

The variable q is a random number uniformly distributed in $[0, 1]$, and q_0 $(0 \leq q_0 \leq 1)$ is a parameter. If $q > q_0$, j is randomly selected according to the probability distribution given by (2).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}][\eta_{ij}]^\beta}{\sum_{j \in N_i^k} [\tau_{ij}][\eta_{ij}]^\beta}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The set of unvisited vertices, N_i^k, contains all the direct successors of vertex i when ant k is at it. The variable τ_{ij} is the pheromone density on edge (i, j). The variable η_{ij} is the heuristic information and is set as the quality score of vertex j, which is computed according to the weighting phase described in [32]. The parameter β controls the influence of η_{ij}.

2) Local Updating Rule: When building up a solution to the problem, i.e., a path through the graph, the ants use a local pheromone updating rule that they apply immediately after having crossed an edge (i, j), which is shown by (3).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0, \forall (i, j) \in E. \tag{3}$$

The variable ξ $(0 < \xi < 1)$ is used to determine the local evaporation rate. At the beginning of the search process, a constant amount of pheromone is assigned to all edges, $\tau_{ij} = \tau_0$ $(\forall (i, j) \in E$, τ_0 is a constant). The local updating rule will reduce the pheromone trail on edge (i, j) after an ant has passed it. In other words, it allows an increase in the exploration of edges that have not been visited yet and, in practice, has the effect that the algorithm does not show stagnation behavior [33].

3) Global Updating Rule: In the iteration, after all ants find their paths, a global pheromone updating rule is performed to the best path found so far, which is given by (4).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{4}$$

The variable ρ $(0 < \rho < 1)$ denotes the global pheromone evaporation rate, and

$$\Delta\tau_{ij} = \begin{cases} U, & \text{if } \forall (i, j) \in P^{bs} \ ; \\ 0, & \text{otherwise .} \end{cases}$$

U is the overall utility of the best path P^{bs}, which is given in [31]. This rule indicates that both the evaporation and the new pheromone deposit, only apply to every edge of the best path P^{bs}. There are two types of best paths: the best path in the current iteration of the trial, called iteration-best, and the best path from the beginning of the trial, called global-best. Experiments have shown that the global-best is slightly better, which is therefore used in this research.

C. Data-Intensive Service Provision Based on a GA

To use a genetic algorithm to search for a solution of our problem, we need to decide the coding scheme, the initial population, the selection operator, the crossover operator, and the mutation operator.

1) The integer array coding scheme is chosen, i.e., every chromosome is represented by an integer array with a number of items. Using the integer array coding scheme, any change in the number of concrete services would not influence the length of the genome. Also, this type of encoding scheme is human-readable and straightforward to represent the service composition solution.
2) The initial population is randomly created according to the service composition graph.
3) The selection operator is the combination of the elitism selection and the tournament selection. The elitism selection involves copying a few best individuals, unchanged, into the next generation. The tournament selection involves selecting a set of pairs of individuals as parents to breed the remainder of the next generation. Two separate tournaments were performed to choose father and mother, respectively.
4) The crossover operator is the single point crossover. Suppose there are $Npop$ individuals in the population and the crossover probability is PC, then there are $Npop * PC$ individuals which are replaced by the new offspring, and there are $Npop * (1 - PC)$ individuals which are able to survive to the next generation. The crossover point is created randomly, but must be checked as to whether it will create infeasible solutions.

5) The mutation policy is proposed as follows. The probability of mutation is PM, which is for the locus. The locus for each gene represents its own position in the chromosome. Every locus in each chromosome that was created by the crossover operation is checked for possible mutation by generating a random number between zero and one. If this random number is less than or equal to the given mutation probability, then the value of the gene will be replaced by the assignment of another concrete service in the service candidate set, according to the local selection approach. The local selection approach is based on the utilities of the concrete services. Prior to the mutation operation, the utility of each concrete service in each service candidate set is computed. Then all the concrete services in each service candidate set are sorted in descending order according to their utilities. When the mutation operation is applied, the replacement process will search another service candidate from the beginning of the service candidate set until the assignment is different from the old assignment, and will then replace it.

D. Experiments and Analysis

The aim of this evaluation is to analyze the performance of the proposed algorithms: 1) comparing the proposed GA with the proposed ACS; 2) comparing the proposed GA and ACS with the mixed integer programming (MIP) approach [34], [35]; and 3) comparing the proposed GA with the GA-based random selection approach [20], [23], [36], [37]. All the experiments are conducted on a computer with Inter Core i5 2500 CPU (3.3GHz and 8 GB RAM).

1) The Parameters: In the experiments, a trial testing method is adopted to determine the most suitable values for all parameters of ACS and GA, considering other researchers' earlier experiments. Finally, the parameters of ACS used in the experiment are: $\beta = 2$, $q_0 = 0.9$, $\tau_0 = 0.1$, $\rho = 0.1$, $\xi = 0.1$, and the number of ants is 20. The parameters of GA in the experiment are: $PC = 0.7$, $PM = 0.1$, and the number of individuals is 20. The weights for cost and response time are 0.8 and 0.2, respectively. The loop structure can be unfolded by cloning the vertices involved in the structure as many times as the maximal loop count [38]. In addition, two termination conditions were defined for both algorithms: iterate until a maximum iteration numbers ($MaxIt = 1000$) is reached, alternatively, iterate until the best utility remains unchanged for a given iteration numbers ($EIT = 50$).

2) Evaluation Methodology: The performance of the GA and the ACS is affiliated to the size of the problem. The size of the problem depends on the number of abstract services in the composite service, and the number of concrete services for each abstract service. Here, the influence of the number of data sets is not considered.

For the purpose of the evaluation, different scenarios are considered where a composite service consists of n abstract services, and n varies in the experiments between 10 and 50, in increments of 10. There are m concrete services in each service candidate set, and m varies in the experiments between 100 and 1000, in increments of 100. Each abstract service requires a set of k data sets, and k is fixed at 10 in the experiments. A scenario generation system is designed to generate all scenarios for the experiments. The system first determines a basic scenario, which includes sequence, conditional and parallel structures. With this basic scenario, other scenarios are generated by either placing an abstract service into it or adding another composition structure as substructure. This procedure continues until the scenario has the predefined number of abstract services.

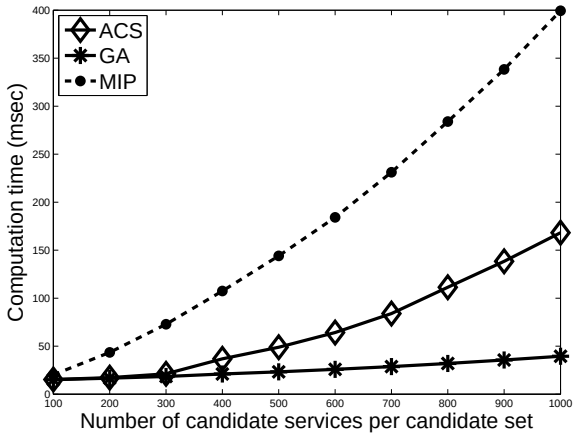
Three performance factors were evaluated: 1) the required computation time; 2) the quality of the solution; and 3) the value of FRIT, which is the number of the iterations when the best utility appeared, and from this iteration the best utility will not change until the termination condition has been reached. As ACS and GA are sub-optimal, the solutions obtained by these two bio-inspired algorithms have been evaluated through comparing them with the optimal solutions obtained by the MIP approach. The overall utility of the solution obtained by ACS or GA is denoted by U_{bio}, and the overall utility of the solution obtained by the MIP approach is denoted by U_{global}. Then the optimality of the solution created by ACS or GA is computed as $optimality = U_{bio}/U_{global}$. The open source integer programming system *lpsolve* version 5.5 [39] was used for the MIP approach.

3) The Dataset: The synthetic datasets were experimented. For each scenario, the price of a data set, the network bandwidth (Mbps) between each data server and the service platform, the storage media speed (Mbps), the size (MB) of a data set, and the number of data requests in the waiting queue were randomly generated from the following intervals: [1,100], [1,100], [1,100], [1000,10000] and [1,10]. Then every scenario was performed with 50 runs. All runs of the same scenario use the same data, and the average results over 50 independent runs are reported.

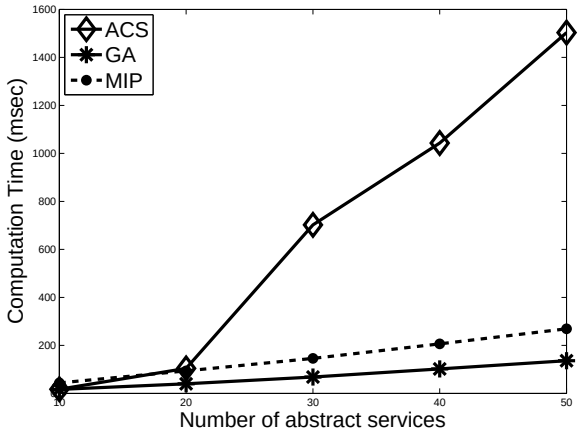
4) Results Analysis: In Fig. 3, the performance of ACS, GA, and MIP are compared with respect to the number of candidate services and the number of abstract services. In Fig. 3(a), the number of candidate services for abstract service varies from 100 to 1000, while the number of abstract service is set to 10. The results indicate that the proposed ACS and GA are faster than the MIP method. By increasing the number of candidate services, the required computation time of GA increases very slowly, this makes GA be more scalable. In Fig. 3(b), the number of abstract services varies from 10 to 50, while the number of candidate services for each abstract service is fixed at 200. The results of this experiment indicate that the performance of all three methods degrades as the number of abstract services increases. Again, we observe that GA outperforms ACS and MIP. The reason is that each ant needs to compute the probability of next vertex while finding the path, and such computation costs much time. Meanwhile, we observe that ACS outperforms MIP when the number of abstract services is small, in this case, less than 20.

Next, the quality of the solutions obtained by ACS and GA is presented. Table I shows that the optimality achieved by ACS and GA with respect to a varying number of candidate services and a varying number of abstract services. The results indicate that GA and ACS are able to achieve the optimal solutions, and GA slightly outperforms ACS.

Table II gives the means of FRIT obtained by ACS and GA with respect to a varying number of candidate services and a varying number of abstract services. The results show that both algorithms need more iterations to get the best utility value when the number of abstract services increases. When the number of concrete services increases, the values



(a) Performance vs. number of candidate services



(b) Performance vs. number of abstract services

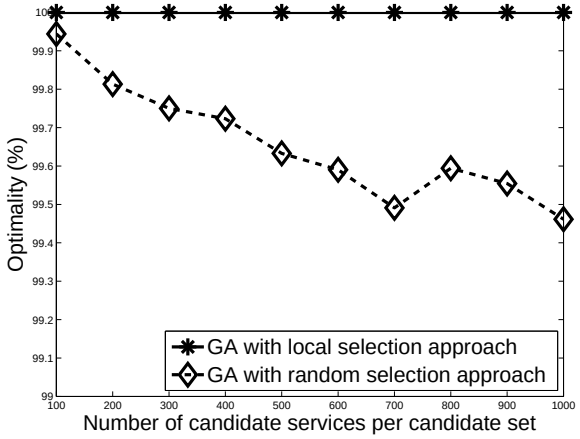
Fig. 3: The performance of ACS, GA, and MIP

TABLE I: Means of optimality (%)

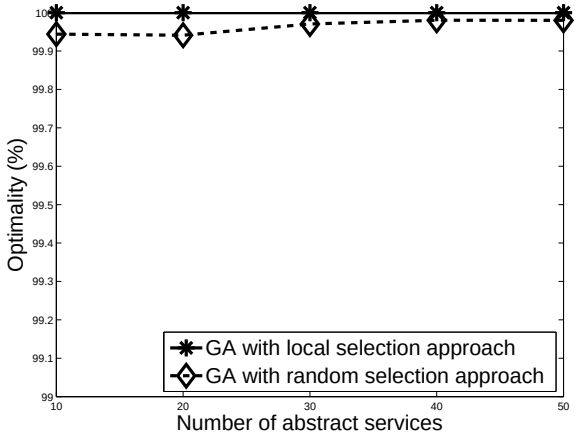
Scenarios	***ACS***	***GA***
n is fixed at 10, m varies between 100 and 1000, in increments of 100	100	100
	100	100
	100	100
	99.9908	100
	100	100
	100	100
	100	100
	99.9908	100
	99.9907	100
	100	100
m is fixed at 100, n varies between 10 and 50, in increments of 10	100	100
	99.9804	100
	99.9111	100
	99.8405	100
	99.7197	100

TABLE II: Means of FRIT

Scenarios	*ACS*	*GA*
n is fixed at 10, m varies between 100 and 1000, in increments of 100	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
m is fixed at 100, n varies between 10 and 50, in increments of 10	1	7
	7	10
	11	12
	18	13
	26	15



(a) Optimality vs. number of concrete services



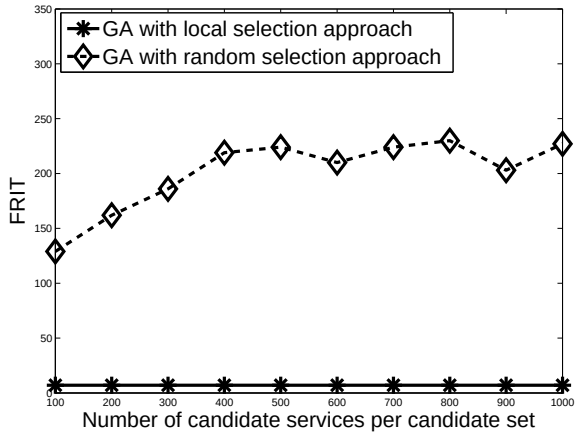
(b) Optimality vs. number of abstract services

Fig. 4: The optimality of the two types of genetic algorithms

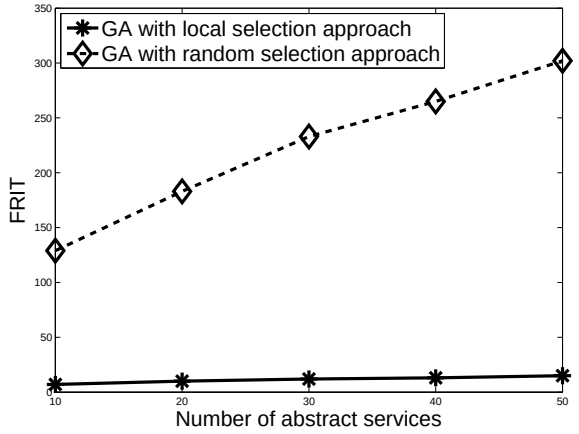
of FRIT of both algorithms does not change and ACS can get the best utility value almost in the first iteration.

The following presents the quality of the solutions obtained by the GA with local selection approach and the GA with random selection approach. Fig. 4 shows the optimality achieved by the two types of GAs in terms of a varying number of candidate services and a varying number of abstract services. The results indicate that GA with local selection approach outperforms GA with random selection approach.

Fig. 5 gives the mean values of FRIT of the two types of GAs in terms of a varying number of candidate services



(a) FRIT vs. number of concrete services



(b) FRIT vs. number of abstract services

Fig. 5: The values of FRIT of the two types of genetic algorithms

and a varying number of abstract services. The results show that when the number of concrete services and the number of abstract services increase, the values of FRIT of the GA with random selection approach increases more quickly than the GA with local selection approach. Again, we observe that the GA with local selection approach outperforms the GA with random selection approach.

IV. Multi-Objective Data-Intensive Service Provision

This section will evaluate ACS and GA for the data-intensive service provision, which is modeled as a multi-objective optimization problem. Both the algorithms for a multi-objective context will get a set of Pareto-optimal solutions by considering two objectives at the same time, the total cost and the total execution time of a composite service. Last section describes two algorithms for the single-objective optimization problem. In the multi-objective algorithms there are two goals, one is to converge to the Pareto-optimal solutions and second is to maintain the diversity of the Pareto-optimal solutions. These two goals cannot be measured by one single performance metric. Also, it needs to define all the algorithmic components in the multi-objective ant colony system (MOACS), and to explore the constraints-handling approach in the multi-objective genetic algorithm (MOGA).

A. Problem Statement

The goal of the majority of existing multi-objective optimization algorithms is to find Pareto-optimal solutions. The concept of dominance is used to relate the solutions found in these algorithms.

Definition 4.1 (Dominance): In a minimization problem for all objectives, a solution x_1 dominates another solution x_2 (denotes as $x_1 \prec x_2$) if and only if the two following conditions are true: 1) x_1 is no worse than x_2 in all objectives, namely $F_i(x_1) \leq F_i(x_2)$ ($\forall i \in \{1, 2, \ldots, N\}$, N is the number of objective functions), and 2) x_1 is strictly better than x_2 in at least one objective, namely $F_j(x_1) < F_j(x_2)$ ($\exists j \in \{1, 2, \ldots, N\}$).

Definition 4.2 (Cover): In a minimization problem for all objectives, a solution x_1 is said to cover another solution x_2 (denotes as $x_1 \preceq x_2$) if one of the two following conditions is true: 1) x_1 dominates x_2, namely $x_1 \prec x_2$, or 2) x_1 is equal to x_2 in all objectives, namely $F_i(x_1) = F_i(x_2)$ $(\forall i \in \{1, 2, \ldots, N\})$.

Definition 4.3 (Non-dominated set): Among a set of solutions, the non-dominated set of solutions are those that are not dominated by any member of the set.

A solution is said to be Pareto-optimal if it is not dominated by any other possible solution. Thus, the Pareto-optimal solutions to a multi-objective optimization problem form the Pareto front or Pareto-optimal set [40]. Pareto-optimal sets are the solutions that cannot be improved in one objective function without deteriorating their performance in at least one of the remaining objective functions.

In the service composition, a graph is used to represent the dependencies between services. Fig. 6 presents an example of a graph in which data sets, as the inputs and outputs of services, are incorporated. The data-intensive service composition problem with global QoS constraints is an extension of the service composition problem, denoted as $G = \{V, E, D, start, end\}$ and is mathematically stated as:

Minimize an objective function F, given:

1) $V = \{AS_1, AS_2, \ldots, AS_n\}$ represents the set of n abstract services, and *start* and *end* represent two virtual tasks;
2) $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$ is the service candidate set of AS_i, which includes all concrete services to implement AS_i;
3) $q_{cs_{i,j}} = [q^1_{cs_{i,j}}, q^2_{cs_{i,j}}, \ldots, q^r_{cs_{i,j}}]$ with r QoS parameters is the QoS vector of concrete service $cs_{i,j}$;
4) E represents the edges of the graph, which includes all links between concrete services of any two connected service candidate sets;
5) $D = \{d_1, d_2, \ldots, d_z\}$ represents a set of z data servers;
6) $DT^i = \{dt^1, dt^2, \ldots, dt^k\}$ represents a set of k data sets which are required by abstract service AS_i, and these data sets are distributed on a subset of D;
7) $Q_c = [Q^1_c, Q^2_c, \ldots, Q^u_c]$ $(1 \leq u \leq r)$ represents a set of global QoS constraints, which define requirements regarding the aggregated QoS values of the requested composite service.

In a traditional service composition problem, a single objective function F may be chosen from any of the following ones:

1) *Inverse of overall utility*

$$F_1 = 1/\sum_{i=1}^{n}\sum_{j=1}^{m}\left(U(cs_{i,j})x_{i,j}\right)$$

where $U(cs_{i,j})$ is the utility of concrete service $cs_{i,j}$. The variable $x_{i,j}$ $(\sum_{j=1}^{m} x_{i,j} = 1)$ represents only one concrete service is selected to implement each abstract service during the process of service composition, where $x_{i,j}$ is set to 1 if $cs_{i,j}$ is selected to implement abstract service AS_i and 0 otherwise.

2) *Overall Cost*

$$F_2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(Cost(cs_{i,j})x_{i,j}\right)$$

where $Cost(cs_{i,j})$ is the cost of concrete service $cs_{i,j}$.

3) *Overall execution time*

$$F_3 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(T_{et}(cs_{i,j})x_{i,j}\right)$$

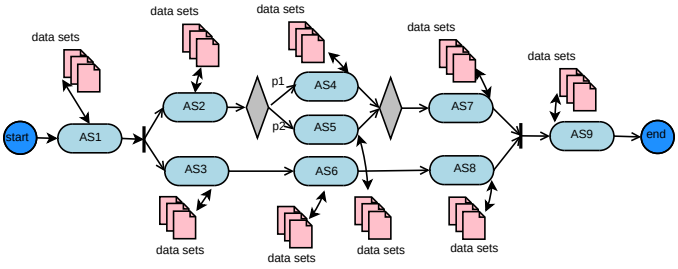


Fig. 6: A graph for the data-intensive service composition

TABLE III: MOACO algorithmic components and values (Adapted from [41])

Components	*Values*	*Illustration*
Ant colony	Single	Multiple colonies can cooperate with each other by exchanging solutions or sharing solutions.
	Multiple	
Pheromone information	Single matrix	The pheromone information associated with each objective are combined.
	Multiple matrices	Each matrix corresponds to one objective.
Heuristic information	Single matrix	The heuristic information associated with each objective is combined.
	Multiple matrices	Each matrix corresponds to one objective.
Pheromone and heuristic aggregation	Weighted sum	$\sum_{f=1}^{N} \lambda_f \eta_{ij}^f$, $\sum_{f=1}^{N} \lambda_f = 1$, where N is the number of objectives.
	Weighted product	$\prod_{f=1}^{N} (\eta_{ij}^f)^{\lambda_f}$, $\sum_{f=1}^{N} \lambda_f = 1$, where N is the number of objectives.
	Random	a random objective is selected to be optimized.
Weight setting	Dynamic	Each ant may be assigned a different weight from the other ants in the iteration.
	Fixed	The weights can be set a priori and each objective has the same importance during the entire algorithm run.
Pheromone update	Elite	The iteration-best or best-so-far solution is used to update the pheromone.
	Best-of-objective	The iteration-best or best-so-far solutions with respect to each objective is used to update the pheromone.
	Non-dominated set	The solutions in the non-dominated set are allowed to update the pheromone.
	All	All ants are allowed to update the pheromone.
Pareto archive	Off-line	The non-dominated solutions can be stored in an external set used to update the pheromone.
	On-line	The Pareto set is connected to the pheromone update procedure at any time.
	No-archive	The Pareto set is not used to update pheromone but it is used as a final solution.

Algorithm 1 Multi-objective data-intensive service composition based on MOACS

$step = 0$; // iteration counter
$GP = \emptyset$; // the global non-dominated set
while 1 **do**
  $step = step + 1$;
  $P = \emptyset$; // The solutions found by ants in each iteration
  set all ants at the $start$ vertex;
  for each ant k **do**
    while ant k is not at the end vertex **do**
      construct a solution according to the state transition rule described in [42];
      record the solution to P;
      apply the local updating rule described in [42];
    end while
  end for
  when all ants arrive at the end vertex, find the non-dominated set from P;
  update the global non-dominated set GP;
  apply the global updating rule described in [42] to GP;
  if $step > MaxIt$ **then**
    break; // $MaxIt$ is the maximum number of iterations
  end if
end while
output all solutions in the global non-dominated set.

where $T_{et}(cs_{i,j})$ is the execution time of concrete service $cs_{i,j}$.

It should be noted that other composition structures and their aggregation functions can also be used in the objective functions. Here, we only list the sequential structure. For the multi-objective context of the present work, the objective function F is considered as a two-dimensional vector, considering the overall cost and execution time, with no objective considered as more important than the other. In this case, a set of Pareto-optimal solutions may be found.

B. Multi-Objective Ant Colony Optimization Algorithms

Various alternatives to the implementation of multi-objective ant colony optimization (MOACO) algorithms have been proposed in the literature. They usually differ from each other with respect to the single-objective ACO algorithms on which they are based, such as the ant system, the ant colony system, and the max-min ant system, as well as the variations in their algorithmic components. The authors of [41] provided a comprehensive review of the use of ACO algorithms in the realm of multiple-objective problems, which is illustrated in Table III.

In this paper, the proposed MOACS used a unique ant colony to simultaneously minimize all functions. All objectives share the same pheromone trails. We need to redesign the state transition rule (1), the local updating rule (3), and the

Algorithm 2 Multi-objective data-intensive service composition based on MOGA

1: $iga = 0$; //generation counter
2: randomly create an initial *Population* with *Npop* individuals;
3: apply the fast non-dominated sort procedure to *Population*; //get the non-dominated fronts and the fitness value of each individual
4: calculate the crowding-distance value of each individual in *Population*;
5: **while** 1 **do**
6: $iga = iga + 1$;
7: select parents from *Population* using binary tournament selection strategy;
8: perform the single-point crossover operator and the mutation operator to the selected parents to create an *Offspring Population*;
9: combine *Population* and *Offspring Population* to get a *Combined Population*;
10: apply the fast non-dominated sort procedure to the *Combined Population*;
11: calculate the crowding-distance value of each individual in the *Combined Population*;
12: use the elitism mechanism to select *Npop* individuals from the *Combined Population* according to the fitness value and the crowding-distance value of each individual;
13: the new selected individuals create a new *Population*;
14: **if** $iga > MaxIg$ **then**
15: break; //$MaxIg$ is the maximum number of generations
16: **end if**
17: **end while**
18: output all individuals after removing the duplication from $Population$.

global updating rule (4) in order to solve the multi-objective data-intensive service provision problem. The details of the new rules for MOACS were described in [42]. The implementation of our proposed MOACS is given in Algorithm 1.

C. Multi-Objective Genetic Algorithms

Multi-objective genetic algorithms (MOGAs) belong to the class of multi-objective evolutionary algorithms, which are stochastic optimization methods and usually use a population-based approach to find Pareto-optimal solutions [43]. The non-dominated sorting genetic algorithm (NSGA) was one of the first multi-objective evolutionary algorithms [44]. The NSGA was not an efficient algorithm because of 1) the high computational complexity of non-dominated sorting, 2) its lack of elitism, and 3) the need for specifying the sharing parameter. Several years later, an improved version of NSGA, called NSGA-II, was proposed to solve the problems of NSGA [45]. The three new mechanisms of NSGA-II are the fast non-dominated sorting procedure, the crowded-comparison procedure, and the elitism mechanism. The authors of [45] also proposed a constraint-handling method for NSGA-II, which was based on the tournament selection algorithm and it separated the constraints and objectives. The tournament selection algorithm is much better than a number of other existing constraint-handling approaches, as confirmed in other studies [46], [47]. In the MOGA of this paper, this constraint-handling method was adopted.

In the proposed MOGA, the integer array coding scheme is used to encode chromosomes. The initial population is randomly created. The binary tournament selection operator and the single-point crossover operator are adopted in the proposed MOGA. The mutation operator for each chromosome replaces the value of the gene with the assignment of another concrete service in the service candidate set randomly. The implementation of the proposed MOGA is given in Algorithm 2. At the end of the algorithm, it needs to remove the duplication and then output the remainder of the population. This is because it is possible that the crossover operator is applied to an individual and to itself.

D. Experiments and analysis

To evaluate the proposed MOACS and MOGA, different scenarios were considered where a composite application comprises services from n abstract services, and n varies in our experiments between 10 and 50, in increments of 10. There are m concrete services in each service candidate set, and m varies in our experiments between 10 and 100, in increments of 10. Each abstract service requires a set of k data sets, and k is fixed at 10 in our experiments.

The following five performance metrics were chosen: 1) the computation time, 2) the overall non-dominated vector generation (ONVG), 3) the comparison metric (C metric), 4) the size of the dominated space, and 5) the summary attainment surface. The first four metrics measure the convergence of the Pareto-optimal solutions, while the fifth metric measures the distribution of the Pareto-optimal set obtained by a multi-objective optimization algorithm. The details of these metrics were given in [42]. A comprehensive comparison of the two algorithms were provided in the following.

Table IV shows the means of the computation time of each scenario. In the upper half of Table IV, the second column indicates that the MOACS needs more computation time when the number of concrete services increases, while the third column shows the computation time of MOGA remains almost steady as the number of concrete services increases. This is because, by using the integer array coding scheme, the change in the number of concrete services will not influence the length of the genome. The computation time of both MOACS and MOGA increases when the number of abstract services increases, which is indicated by the lower half of Table IV. The upper half of Table IV indicates that when the number of abstract services and concrete services is small, MOACS is better than MOGA since the means of the computation time of MOACS are lower than those of MOGA except in the scenario where $n = 10$ and $m = 100$. Meanwhile, the lower half of Table IV indicates that MOGA is more scalable than MOACS when there is a large number of concrete services and abstract services.

Table V gives the means of ONVG. By comparing the second and third column of the upper half of Table V, we conclude that MOGA can get more non-dominated solutions than MOACS except in the scenario where $n = 10$ and $m = 10$. On the other hand, the lower half of Table V indicates that MOACS can find more non-dominated solutions than MOGA when the number of abstract services increases except in the scenario where $n = 10$ and $m = 50$.

Table VI provides the means of $PS(A)$. By comparing the second and third column of Table VI, we conclude that MOACS is better than MOGA since MOACS always leads to a higher value of $PS(A)$.

Table VII presents the means of the C metric. The value in the second column is equal to the value in the third column of Table VII. The results indicate that the convergence of the Pareto-optimal solutions of MOACS and MOGA has never been different, so we cannot say one is better than the other with respect to the C metric.

Fig. 7 gives an example of the median summary attainment surface of MOACS and MOGA. The regions where no difference between the points of the median attainment surfaces of the two algorithms could be found were indicated in gray dots, whereas those regions where the points of the two surfaces were found to differ from each other are plotted in stars and squares, respectively. In the regions where the points of the two surfaces were found to differ from each other, there are three situations: 1) if the points of the median attainment surfaces of MOACS dominate those of MOGA, then the label MOACS is put near the points, 2) if the points of the median attainment surfaces of MOGA dominate those

TABLE IV: Means of Computation Time

Scenarios	*MOACS*	*MOGA*
n is fixed at 10, m varies between 10 and 100, in increments of 10	22.3333	29.6667
	23.6190	30.0952
	24.4762	29.8571
	25.0952	31.3333
	26.5238	29.9524
	27.1905	31.2857
	27.6667	31.1429
	29.5238	30.9048
	30.6667	30.7143
	30.4286	30.2381
m is fixed at 50, n varies between 10 and 50, in increments of 10	26.5238	29.9524
	61.2381	30.7143
	98.9048	30.2381
	141.9524	31.1905
	285.5714	31.7143

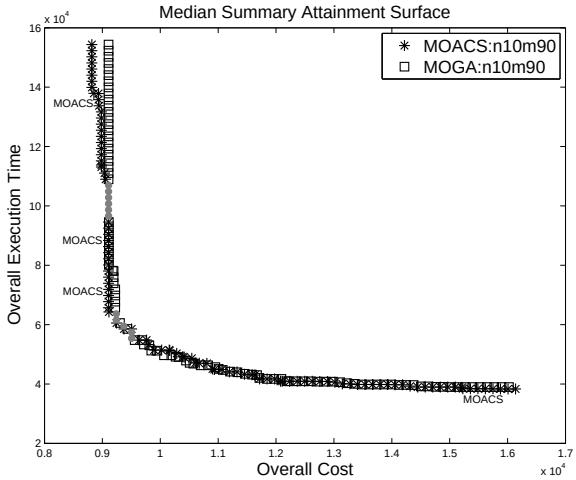


Fig. 7: Median summary attainment surface with $(n, m) = (10, 90)$

TABLE V: Means of ONVG

Scenarios	*MOACS*	*MOGA*
n is fixed at 10, m varies between 10 and 100, in increments of 10	14.9048	14.8571
	32	36.0476
	27.4762	34.4286
	22.0952	27.5714
	35.8571	49.8045
	25.3333	28.0952
	20.1905	27.4286
	38.5238	46.0476
	34.8095	38.3333
	15.3333	16.5714
m is fixed at 50, n varies between 10 and 50, in increments of 10	35.8571	49.8045
	75.4762	69.6196
	90.7619	83.1905
	114.9524	82.3810
	131.5714	84.1429

TABLE VI: Means of $PS(A)$ (%)

Scenarios	*MOACS*	*MOGA*
n is fixed at 10, m varies between 10 and 100, in increments of 10	82.65	82.64
	84.86	84.77
	85.29	85.25
	84.65	84.61
	85.88	85.87
	86.30	86.12
	86.17	86.02
	86.12	85.92
	86.72	86.38
	86.33	86.28
m is fixed at 50, n varies between 10 and 50, in increments of 10	85.88	85.87
	68.96	68.31
	51.77	49.81
	38.01	35.96
	24.74	21.90

TABLE VII: Means of C Metric

Scenarios	*C(MOACS,MOGA)*	*C(MOGA,MOACS)*
n is fixed at 10, m varies between 10 and 100, in increments of 10	0.9524	0.9524
	0.6298	0.6298
	0.6037	0.6037
	0.8429	0.8429
	0.8311	0.8311
	0.5982	0.5982
	0.6219	0.6219
	0.5127	0.5127
	0.4523	0.4523
	0.5189	0.5189
m is fixed at 50, n varies between 10 and 50, in increments of 10	0.8311	0.8311
	0.4530	0.4530
	0.4452	0.4452
	0.2338	0.2338
	0.2094	0.2094

of MOACS, then the label MOGA is put near the points, 3) if the points of the median attainment surfaces of MOACS are not dominated by those of MOGA and the points of the median attainment surfaces of MOGA are not dominated by those of MOACS, then no label is put. The details of all the median summary attainment surface of MOACS and MOGA were given in [42]

The lessons learned from the experimental results are that, when we have a large number of concrete services available for each abstract service, a multi-objective genetic algorithm can achieve better solutions. On the other hand, whenever the number of concrete services available is small, such as in some simple and repetitive scientific computation, a multi-objective ant colony system is to be preferred to a multi-objective genetic algorithm.

V. Ant-Inspired Negotiations in the Data-Intensive Service Provision

Negotiation has been adopted in service provision in order to get better QoS attributes. An iterative negotiation approach for a service composition was presented in [48]. The aim of the approach was to select services for the service-based systems in the scenarios where the QoS constraints were severe. The authors of [49] designed a framework in which the service level agreements for a service composition were established through autonomous agent negotiation. A new negotiation protocol was also proposed to support coordinated negotiation. The study [11] introduced an approach for the Web service selection problem with large scale processes and severe QoS constraints. The Web service selection problem was formalized as a mixed integer programming problem and loop peeling was adopted for optimization in that paper.

The authors of [50] incorporated trust to manage the life cycle of scientific workflow, and applied a genetic algorithm to assign task for autonomous service agent.

However, the negotiation approaches in the above studies are not able to effectively solve the problem in this paper, in which the data plays the dominant role. The cost and response time of services largely depend on the accessing cost and response time of data sets. As described in section III, the decisions of the service composers, the service providers, and the data providers depend on each other. The data provider sells data sets to multiple service providers in order to maximize the data usage and the profit. The cost and response time of data sets for one service provider are affected by the demand of the others. The service providers also play the requester role with respect to the data sets. Thus, service providers will have two aims, one is to lower the access cost and response time of data sets and the other is to maximize their profit and service usage. Also, the service providers compete with other service providers to initiate or maintain a contract with the service composers and are invariably interested in cost saving. The actual usage of services typically encourages the composers to have a long term contract with the service providers. They also select concrete services that best match the QoS requirements. Meanwhile, data-intensive services are typically used in a dynamic and changing environment, and different providers typically have conflicting objectives. In order to automate the process of reaching an agreement in the problem of this paper, a group of agents was exploited to establish agreeable service contracts.

The lifetime of the problem framework was described in [13]. In the lifetime, two-stage negotiation processes were used. In the first stage, a service composer negotiates with multiple service providers over each service in a structured one-to-many negotiation process. In the second stage, each service provider negotiates with a data provider over a set of data sets in a structured one-to-one negotiation process. A multi-phase, multi-party negotiation protocol for the problem was also presented [13]. A negotiation process is the interplay of offers and counter-offers between a buyer and a seller, with different criteria and goals, working to identify a mutually acceptable solution. The decision making model was designed for each agent in order to prepare an offer and a counter-offer.

The details of the evaluation of the ant-inspired negotiation approach were given in [13]. The performance of the negotiation approach was evaluated with respect to three factors: 1) the success rate of finding an optimal solution, 2) the number of negotiation rounds, and 3) the computation time in each negotiation round. To evaluate the effectiveness of our negotiation approach, we compared the success rate of our approach with that of the MIP approach. In our experiments, the success rate of the MIP approach remained zero in all scenarios, while our negotiation approach maintained a higher success rate compared with the MIP approach. Meanwhile, our experimental results indicated that the number of negotiation rounds increased when the number of concrete services increased. When the number of abstract services or the number of data sets increased, the number of negotiation rounds, on the other hand did not exact increase. Also, our results indicated that the time consumption per round increased when the number of abstract services, the number of concrete services, or the number of data sets increased. The experimental results showed that our negotiation-based approach, compared with the traditional non-dynamic method, could facilitate the data-intensive service provision with a better outcome.

VI. Remaining Problems

The explosion of raw data and the dependence on data services are expected to be further amplified, as a result of the enormous proliferation of data-intensive services, for example, in critical areas such as disaster management and health care. Now cloud infrastructure and platforms have become viable mainstream solutions for data accessing, processing, analyzing, storage and distribution. Because the primary motivation for moving to the cloud is to minimize cost (or maximize earnings), i.e., for economic reasons, we should attempt to find approaches to supporting economic data-intensive services provision from holistic perspectives.

A. Strategies to Lower the Total Cost

In the ant-inspired negotiation with severe global QoS constraints, the service providers and the data providers improve their offers based on their predefined ranges [13]. This subsection outlines the strategies that may be used by the service providers and the data providers to lower the total cost of the composite service.

1) Data Pricing Strategy for Service Providers: For each service provider, one strategy to decrease the cost of a service is to choose different data pricing models. The data provider offers the usage-based pricing model, the package-based pricing model, and the subscription-based pricing model [51]. In the subscription-based pricing model, the data requesters need to pay once for the data set and afterwards they can access the data set for a period of time. The access cost of the data set in this period is independent of the number of usages. One service might require many data sets and one data set might also be required by many services. A service provider can choose different pricing model based on the demands. For example, a service provider will prefer to choose the subscription-based pricing model rather than a usage-based and package-based pricing model if the combination of usage and package-based pricing model is not attractive anymore. During the service concretization, the data provider will package the data sets based on the requests. If some data sets are always used together by many services, the data provider will package them and give a discount to the service providers.

2) Data Placement Strategy for Data Providers: The data provider may improve the quality 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). Before the negotiations, we may assume that the data provider places all data sets in a data center, and all service providers access data from this data center. Fig. 8 illustrates the data placement before the negotiation. During the negotiations, the data provider needs to effectively store all data sets in order to maximize the data usage and the profit. In order to prevent data gathering to one data center and reduce the access response time of data sets, the data provider partitions all data sets and distributes these partitions to different data centers, and replicates portions of the data sets. Fig. 9 illustrates the data placement after the negotiation. The data provider makes decisions based on the access frequency and update rate of the data sets as well as the locality of service providers. The service providers prefer the data sets to be placed in the closest data center as much as possible, as this will efficiently minimize the total data movement. If the service cannot find the required data sets from the closest data center, it needs to access them from the other data centers. The data placement problem is very similar to the minimum K-median problem, especially from a facility location perspective [52]. By investigating the data placement policies, the data providers could lower the total cost of the composite service.

B. Economic Mechanisms for Data-Intensive Service Provision

The economic model described in our earlier work [31] 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.

As mentioned in [51], if a data set has more than one data replica, the service provider needs a data replica selection strategy to select data replicas, since mass data-moving influences the effectiveness and efficiency of the application execution. The literature has presented static replica selection approaches and dynamic replica selection approaches.

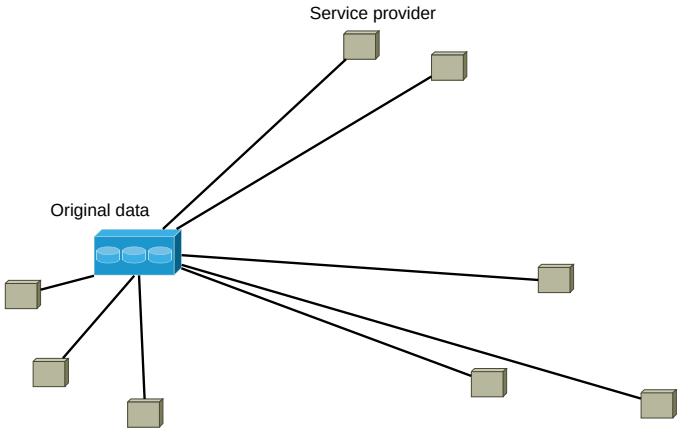


Fig. 8: Before data partition and replication takes place

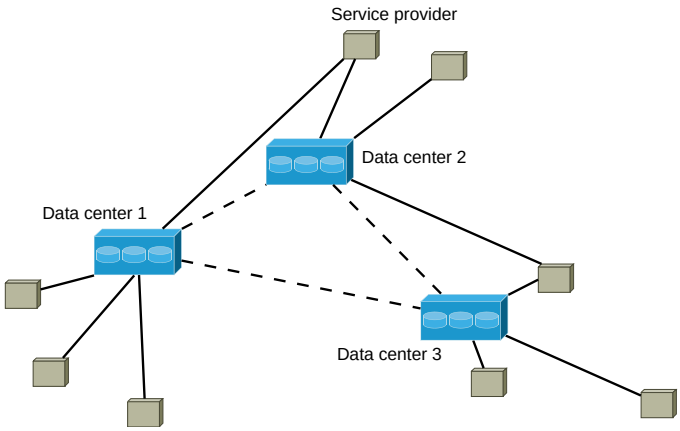


Fig. 9: After data partition and replication has taken place

Only very few studies proposed replica selection strategies based on bio-inspired algorithms. In order to achieve an economic solution for data-intensive service provision problems, novel mechanisms are needed to be developed for selecting data replicas.

We have presented an ant-inspired negotiation approach for the data-intensive service provision [13], [53]. In the real negotiation processes, it is useful for a service composer releases information about the status of the competition to the service providers, enabling them to make decisions about how to purchase the data and how to price their services. More complicated decision making tactics are needed to be investigated to generate counter-proposals for each provider. The overall objective is to build an economic model to ensure enhanced configuration for big data service provision, potentially saving money, energy and space for maintaining huge amount of data in future information infrastructures.

VII. Conclusion

Data-intensive service provision faces new challenges with the rapid proliferation of services and the development of cloud computing. The outcomes of our earlier studies confirmed the applicability and efficiency of bio-inspired algorithms to solve data-intensive service provision problems. We created a hierarchical taxonomy of Web service concretization approaches. And we also conducted a systematic review of Web service concretization based on the three bio-inspired algorithms. The findings from the systematic review had been the basis for solving the problems. An economic model was established to represent and regulate the interactions among the end users, the service composers, the service providers, and the data providers. An extensible QoS model was also constructed to show the service and data set usage and charging relationship. Then this paper applied an ant colony system and a genetic algorithm to compose data-intensive services. It also evaluated the two algorithms and compared them with other traditional methods. Both algorithms were proposed to optimize the total cost of a composite service. The paper then investigated a multi-objective ant colony system and a multi-objective genetic algorithm for the problem. Further, we proposed an ant-inspired negotiation approach. The lifetime of the data-intensive service provision and the two-stage negotiation processes were described. We also designed a multi-phase, multi-party negotiation protocol, where an ant colony system was applied to select services. Finally, this paper presents the opportunities for future work.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China under Grants No. 61320106007 and No. 61202449, China National High Technology Research and Development Program under Grants No. 2013AA013503, China Specialized Research Fund for the Doctoral Program of Higher Education under Grants No. 20110092130002, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No. 93K-9.

References

[1] Savitz E. Gartner: 10 critical tech trends for the next five years 2012. http://www.forbes.com/sites/ericsavitz/2012/10/22/gartner-10-critical-tech-trends-for-the-next-five-years/.
[2] Bell G, Hey T, Szalay A. Beyond the data deluge. *Science* 2009; **323**(5919):1297–1298.
[3] Huang W, Chen Z, Dong W, Li H, Cao B, Cao J. Mobile internet big data platform in china unicom. *Tsinghua Science and Technology* 2014; **19**(1):95–101.
[4] Tian Y, Liu C, Chen Z, Wan J, Peng X. Performance evaluation and dynamic optimization of speed scaling on web servers in cloud computing. *Tsinghua Science and Technology* 2014; **18**(3):298–307.
[5] Philip Chen C, Zhang C. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences* 2014; **275**:314–347.
[6] Watson W, Bird I, Chen J, Hess B, Kowalski A, Chen Y. A web services data analysis grid. *Concurrency and Computation: Practice and Experience* 2002; **14**:1303–1311.
[7] Woolf A, Haines K, Liu C. A web service model for climate data access on the grid. *International Journal of High Performance Computing Applications* 2003; **17**(3):281–295.
[8] Balasubramaniam S, Botvich D, Carroll R, Mineraud J, Nakano T, Suda T, Donnelly W. Biologically inspired future service environment. *Computer Networks* 2011; **55**(15):3423–3440.
[9] Alonso G, Casati F, Kuno H, Machiraju V. *Web Services - Concepts, Architectures and Applications*. Springer-Verlag: Berlin, Heidelberg, 2004.
[10] Dustdar S, Papazoglou M. Services and service composition - an introduction. *Information Technology* 2008; **50**(2):86–92.
[11] Ardagna D, Pernici B. Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering* 2007; **33**(6):369–384.
[12] Canfora G, Penta M, Esposito R, Villani M. A lightweight approach for qos aware service composition. *Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC '04)*, ACM: New York, NY, USA, 2004; 36–47.
[13] Wang L, Shen J. Multi-phase ant colony system for multi-party data-intensive service provision. *IEEE Transactions on Services Computing, in press* 2014; http://dx.doi.org/10.1109/TSC.2014.2358213.
[14] Dorigo M, Stutzle T. *Ant Colony Optimization*. MIT Press: Cambridge, MA, USA, 2004.
[15] Wang L, He Y. A web service composition algorithm based on global qos optimizing with mocaco. *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011)*, *Advances in Intelligent and Soft Computing*, vol. 111, Jiang L (ed.). Springer-Verlag: Berlin, Heidelberg, 2010; 79–86.
[16] Zhang W, Chang C, Feng T, Jiang H. Qos-based dynamic web service composition with ant colony optimization. *Proceedings of the 34th Annual IEEE International Computer Software and Applications Conference (COMPSAC '10)*, IEEE Computer Society: Washington, DC, USA, 2010; 493–502.

[17] Wang Z, Liu Z, Zhou X, Lou Y. An approach for composite web service selection based on dgqos. *The International Journal of Advanced Manufacturing Technology* 2011; **56**(9-12):1167–1179.

[18] Srinivas M, Patnaik L. Genetic algorithms: A survey. *Computer* 1994; **27**(6):17–26.

[19] Zhang C, Su S, Chen J. A novel genetic algorithm for qos-aware web services selection. *Data Engineering Issues in E-Commerce and Services*, *Lecture Notes in Computer Science*, vol. 4055, Lee J, Shim J, Lee S, Bussler C, Shim S (eds.). Springer-Verlag: Berlin, Heidelberg, 2006; 224–235.

[20] Gao C, Cai M, Chen H. Qos-aware service composition based on tree-coded genetic algorithm. *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, IEEE Computer Society: Washington, DC, USA, 2007; 361–367.

[21] Ai L, Tang M. Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm. *IEEE Fourth International Conference on eScience (eScience '08)*, Indianapolis, IN, 2008; 119–126.

[22] Ma Y, Zhang C. Quick convergence of genetic algorithm for qos-driven web service selection. *Computer Networks* 2008; **52**(5):1093–1104.

[23] Tang M, Ai L. A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. *Proceeding of the 2010 World Congress on Computational Intelligence*, IEEE Computer Society: Washington, DC, USA, 2010; 268–275.

[24] Fan X, Jiang C, Fang X. An efficient approach to web service selection. *Web Information Systems and Mining*, *Lecture Notes in Computer Science*, vol. 5854, Liu W, Luo X, Wang F, Lei J (eds.). Springer-Verlag: Berlin, Heidelberg, 2009; 271–280.

[25] Wang W, Sun Q, Zhao X, Yang F. An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems* 2012; **3**(01):18–30.

[26] Zhao X, Song B, Huang P, Wen Z, Weng J, Fan Y. An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing* 2012; **12**(8):2208–2216.

[27] Li Y, Lin C. Qos-aware service composition for workflow-based data-intensive applications. *Proceedings of IEEE International Conference on Web Services*, IEEE Computer Society: Washington, DC, USA, 2011; 452–459.

[28] Milenkovic M, Castro-Leon E, Blakley J. Power-aware management in cloud data centers. *Cloud Computing*, *Lecture Notes in Computer Science*, vol. 5931, Jaatun M, Zhao G, Rong C (eds.). Springer-Verlag: Berlin, Heidelberg, 2009; 668–673.

[29] Song Y, Wang H, Li Y, Feng B, Sun Y. Multi-tiered on demand resources scheduling for vm-based data center. *Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, IEEE Computer Society: Washington, DC, USA, 2009; 148–155.

[30] Winter M. Data center consolidation: A step towards infrastructure clouds. *Cloud Computing*, *Lecture Notes in Computer Science*, vol. 5931, Jaatun M, Zhao G, Rong C (eds.). Springer-Verlag: Berlin, Heidelberg, 2009; 190–199.

[31] Wang L, Shen J, Luo J, Dong F. An improved genetic algorithm for cost-effective data-intensive service composition. *Proceedings of The 9th International Conference on Semantics, Knowledge & Grids (SKG)*, IEEE Computer Society: Washington, DC, USA, 2013; 105–112.

[32] Wang L, Shen J, Beydoun G. Enhanced ant colony algorithm for cost-aware data-intensive service provision. *IEEE 9th World Congress on Services*, IEEE Computer Society: Washington, DC, USA, 2013; 227–234.

[33] Dorigo M, Gambardella L. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1997; **1**(1):53–66.

[34] Alrifai M, Risse T, Nejdl W. A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web* 2012; **6**(2):7:1–7:31.

[35] Ramacher R, Monch L. Cost-minimizing service selection in the presence of end-to-end qos constraints and complex charging models. *Proceedings of IEEE 9th International Conference on Services Computing (SCC)*, IEEE Computer Society: Washington, DC, USA, 2012; 154–161.

[36] Canfora G, Penta M, Espositio R, Villani M. An approach for qos-aware service composition based on genetic algorithms. *Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05)*, ACM: New York, NY, USA, 2005; 1069–1075.

[37] Jaeger M, Muehl G. Qos-based selection of services: The implementation of a genetic algorithm. *Proceedings of KiVS 2007 Workshop: Service-Oriented Architectures and Service-Oriented Computing (SOA/SOC)*, Braun T, Carle G, Stiller B (eds.), VDE: Bern, Switzerland, 2007; 359–370.

[38] Zeng L, Benatallah B, Ngu A, Dumas M, Kalagnanam J, Chang H. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering* 2004; **30**(5):311–327.

[39] Berkelaar M, Eikland K, Notebaert P. lpsolve: Open source (mixed-integer) linear programming system 2004. http://lpsolve.sourceforge.net/.

[40] Taboada H, Espiritu J, Coit D. Moms-ga: A multi-objective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Transactions on Reliability* 2008; **57**(1):182–191.

[41] Angelo J, Barbosa H. Ant colony algorithms for multiobjective optimization. *Ant Colony Optimization - Methods and Applications*, Ostfeld A (ed.). InTech: Manhattan, NY, USA, 2011. http://www.intechopen.com/books/ant-colony-optimization-methods-and-applications/ant-colony-algorithms-for-multiobjective-optimization,accessed 15.03.2015.

[42] Wang L, Shen J, Luo J. Multi-objective ant colony system for data-intensive service provision 2014. Accepted by the 2nd International Conference on Advanced Cloud and Big Data.

[43] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation* 1999; **3**(4):257–271.

[44] Srinivas N, Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* 1994; **2**(3):221–248.

[45] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 2002; **6**(2):182–197.

[46] Deb K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* 2000; **186**(2-4):311–338.

[47] Deb K, Agrawal R. Simulated binary crossover for continuous search space. *Complex Systems* 1995; **9**(2):115–148.

[48] He Q, Yang Y, Yan J, Jin H. Insc: An iterative negotiation approach for service compositions. *Proceedings of the IEEE 9th International Conference on Services Computing (SCC)*, IEEE Computer Society: Washington, DC, USA, 2012; 170–177.

[49] Yan J, Kowalczyk R, Lin J, Chhetri M, Goh S, Zhang J. Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems* 2007; **23**(6):748–759.

[50] Wang M, Ramamohanarao K, Chen J. Trust-based robust scheduling and runtime adaptation of scientific workflow. *Concurrency and Computation: Practice and Experience* 2009; **21**(16):1982–1998.

[51] Wang L, Luo J, Shen J, Dong F. Cost and time aware ant colony algorithm for data replica in alpha magnetic spectrometer experiment. *IEEE 2nd International Congress on Big Data*, IEEE Computer Society: Washington, DC, USA, 2013; 254–261.

[52] Byrka J. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, *Lecture Notes in Computer Science*, vol. 4627, Charikar M, Jansen K, Reingold O, Rolim JDP (eds.). Springer-Verlag: Berlin, Heidelberg, 2007; 29–43.

[53] Wang L, Shen J, Zhou Q, Beydoun G. Ant-inspired multi-phase and multi-party negotiation in the data-intensive service provision. *Proceedings of the IEEE 11th International Conference on Services Computing (SCC)*, IEEE Computer Society: Washington, DC, USA, 2014; 211–218.