symmetry

MDPI

Article

Dynamic Service Selection Based on Adaptive Global QoS Constraints Decomposition

Yuan Yuan, Weishi Zhang *, Xiuguo Zhang and Huawei Zhai

College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China; yuanyuandr@126.com (Y.Y.); zhangxg@dlmu.edu.cn (X.Z.); zhw@dlmu.edu.cn (H.Z.)

* Correspondence: teesiv@dlmu.edu.cn; Tel.: +86-139-9862-8052

Received: 13 February 2019; Accepted: 14 March 2019; Published: 19 March 2019

Abstract: As there are more and more available Web services with the same or similar functionalities but different Quality of Service (QoS), the challenge of QoS-aware service composition is to efficiently select appropriate component services to achieve maximum utility and meet the global QoS constraints with low time cost. In this paper, we propose a dynamic service selection approach based on adaptive global QoS constraints decomposition. Fuzzy logic technology and Cultural Genetic Algorithm are used to adaptively decompose global QoS constraints into near-optimal local constraints. According to the near-optimal local constraints, the optimal service is selected for each service class during the running time efficiently. Experimental results show that the proposed approach not only achieves the near-optimal solution, but also significantly reduces the computation time, and has good adaptability and scalability.

Keywords: dynamic service selection; global QoS constraints decomposition; fuzzy logic technology; Cultural Genetic Algorithm

1. Introduction

Service-oriented architecture (SOA) is a modern paradigm to develop software systems that are often described as composite Web services [1]. Currently, composite Web services have been widely used in various areas, such as virtual enterprise, supply chain, accounting, finances, and e-Science. With the emergence of massive Web services, there are more and more candidate services that can provide the same or similar functionality. As a result, it is essential to consider Quality of Service (QoS) such as response time, price, availability, and so on [2,3]. In recent years, QoS-aware service composition has been widely studied [4,5].

Since Web services are invoked in a dynamic environment, the QoS attributes and user's preferences are changing at runtime. In order to deal with the above runtime changes more efficiently and flexibly, dynamic service composition is proposed. It is described as a process containing a set of abstract services, and a concrete service is selected, bound, and invoked for each abstract service at runtime [6]. Moreover, users always put forward global QoS requirements, and as a result, it is demanded to select a concrete service for each abstract service that can satisfy global QoS constraints [7]. Service selection based on global QoS constraints is a combination optimization problem. The time complexity for the global optimization increases exponentially with the increasing of services or QoS attributes. Thus, it is a challenge to develop an efficient QoS-aware dynamic service selection approach that can maximize the utility and satisfy the global QoS constraints and user's preferences as well.

Recently, researchers have carried out intensive studies on service selection based on global QoS constraints. The existing studies can be classified as optimization-based approaches and heuristic-based approaches. Optimization-based approaches aim at finding the optimization solution based on user's QoS constraints [8,9]. These methods usually suffer from poor scalability due to the exponential time complexity. Heuristic-based approaches search for near-optimal solutions with a polynomial time complexity [10,11]. However, as these heuristic-based methods usually require a large number of global data and high cost of communication, they are not appropriate in distributed environments [12].

Currently, there are several heuristic-based methods based on global constraints decomposition [1,6,13,14]. In these methods, the value range of each QoS attribute of each service class is divided into a set of discrete values, which are called quality levels. The global QoS constraints are decomposed into local constraints for each service class by solving the optimal quality level scheme. Finally, local constraints are used to select local component service. These methods can achieve an approximate utility in dramatically reduced time and meet the global constraints as well. However, the above approaches do not discuss the number of quality levels, which is very important for service composition. Due to the constant number of quality levels, the above approaches are short of adaptability. As a result, the user's preferences cannot be satisfied well. For example, some users need to find the composite service quickly instead of high quality; however, some users hope to get a trade-off between searching time and high quality.

To overcome the problems above, this paper proposes a dynamic service selection approach based on adaptive QoS constraints decomposition (AQCD). The main idea is to decompose global QoS constraints into local constraints, and according to the local constraints, select a component service for each service class independently. According to user's preferences, utility function values, and time cost, we use fuzzy logic technology to automatically adjust the number of quality levels in order to generate the optimal quality level scheme. Finally, we use Cultural Genetic Algorithm (CGA) to solve the near-optimal global QoS constraints decomposition. In order to verify the effectiveness of our AQCD approach, extensive experiments were conducted on the real Web services data set QWS and the randomly generated data set RQWS. We compare the performances of AQCD, QCD, and the integer programming-based approach (WS-IP) by the running time and approximation ratio. Experimental results show that AQCD can efficiently solve the QoS-aware service composition problem with near-optimal solutions and low time cost. Further, it can fit well to user's preferences and the increase in candidate services; that is to say, our approach has a good adaptability and scalability.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 briefly describes some basic concepts and formulates the service selection problem. Section 4 describes the process of dynamic service selection based on global QoS constraints decomposition. Section 5 proposes the adaptive adjustment approach based on fuzzy logic technology. Section 6 proposes the global QoS constraints decomposition approach based on CGA. Section 7 proposes the local service selection method. Section 8 presents experimental results. Finally, Section 9 concludes this paper.

2. Related Work

Web-service selection based on global QoS constraints considers the overall constraints provided by users, to find the optimal set of services. It is at the expense of high computation complexity to evaluate all the possible service compositions. To address this problem, a variety of approaches have been proposed. In References [8,9,15], QoS-aware service selection problem was modeled as a multidimensional multi-choice knapsack problem (MMKP). Integer programming methods [8,9,16] have been used to search for the optimal solution. The above approaches are very effective for small-scale problems. However, as the increasing of services, these approaches suffer from poor scalability because of the exponential time complexity.

To address the above problem, several bio-inspired algorithms [17] have been used to deal with QoS-aware service selection problem. There are three common bio-inspired algorithms used for service composition—genetic algorithm (GA) [18,19], ant colony optimization (ACO) algorithm [20,21], and particle swarm optimization (PSO) algorithm [22]. Furthermore, several researchers proposed improved bio-inspired algorithms to get better solutions with less time. References [11,23] provided a hybrid GA to solve the optimal service composition problem. Reference [24] proposed a variable length chromosome GA to deal with QoS-aware service selection. In Reference [25], QoS-aware service selection problem was modeled as a multi-objective optimization problem, and a multi-objective chaos ACO algorithm was proposed to solve it. Reference [26] proposed an immune optimization algorithm based on PSO and verified the better performances in aspects of the convergence rate, the searching ability, and the stability. Reference [10] proposed a cross-modified artificial bee colony algorithm to solve QoS-aware service selection. However, because of the requirement of global data visibility, it is difficult to apply these approaches in a distributed environment.

Recently, several QoS-aware service selection approaches based on global constraints decomposition have been proposed. From the viewpoint of computation time, the approach based on decomposition can be more appropriate and it can be applied in a distributed environment. The global constraints can be transformed to the local constraints through different approaches. Reference [27] discussed the constraints decomposition in the sequential, parallel, and condition structures, and then presented an approach for determining the local constraints in a general structure. Most works [1,6,13,14] divided the value range of each QoS attribute of each service class into discrete values, which were called quality levels. These approaches mapped global constraint into a set of quality levels. Finally, these quality levels were used as local constraints to select component service. However, all of the above approaches neglected the effect of the user's preferences on the global constraints' decomposition.

To overcome the above problem, this paper proposes a novel dynamic service selection approach based on adaptive global QoS constraints decomposition, in which the number of quality level is automatically adjusted.

3. Problem Formulation

In order to introduce the process of dynamic service selection based on global QoS constraints decomposition, this paper lists some basic concepts as below.

Definition 1. *Component service (s): It is the basic unit in service composition, providing services to users.*

Definition 2. *Service class (S_i): A service class $S_i = \{s_{i1}, s_{i2}, s_{i3}, \cdots, s_{im}\}$ denotes the ith abstract service of a composite Web service. It has m candidate services, which have the same functionality but differ in QoS attributes.*

Definition 3. *QoS vector ($Q(s)$): A QoS vector $Q(s) = \{q_1(s), q_2(s), \cdots, q_r(s)\}$ contains r QoS attributes of service s.*

Definition 4. *QoS aggregation for a composite service (CQ): $CQ = \{Cq_1, Cq_2, \cdots, Cq_r\}$ contains r QoS attributes of a composite service. It can be calculated in terms of QoS values of component services and the composition structures.*

Definition 5. *User preferences W: $W = \{w_1, w_2, \cdots, w_r\}$ represents user preferences, where w_k ($1 \le k \le r$) is user's preference for the kth QoS attributes. $\sum_{k=1}^{r} w_k = 1$, and $w_k \in (0, 1)$.*

Definition 6. *Global QoS Constraints (C): $C = \{C_1, C_2, \cdots, C_r\}$ is the set of user's global QoS constraints, which contains r constraints and C_k ($1 \le k \le r$) is a global constraint over q_k. C_k can be shown according to upper and/or lower bounds for the QoS aggregation value Cq_k.*

3.1. QoS Aggregation for a Composite Service

There has been much research about service composition structures and QoS aggregation formulas. In Reference [27], the QoS attributes were divided into three different categories—additive attributes, multiplicative attributes, and max-operator attributes. There are four basic composition structures—sequential, parallel, conditional, and loop structures [27]. This paper only considers the sequential composition structure. Other composition structures can be converted to the sequential composition structure through the methods mentioned in Reference [27]. In this paper, we study five QoS attributes, including price, response time, availability, and throughput. We assume that a composite service is sequentially constructed by n component services. The QoS aggregation formulas of five QoS attributes are as shown in Table 1.

Table 1. Quality of Service (QoS) aggregation formulas for the sequential composition structure.

Price	Response Time	Availability	Throughput	Successful Execution Rate
$P = \sum_{i=1}^{n} P_i$	$T = \sum_{i=1}^{n} T_i$	$A = \prod_{i=1}^{n} A_i$	$R = \min_{i=1}^{n}\{R_i\}$	$S = \prod_{i=1}^{n} S_i$

3.2. Utility Function

There are many candidate services with multiple QoS attributes in service composition. In order to calculate the QoS attributes of candidate services, we use the Simple Additive Weighting (SAW) approach [28] to map the QoS vector $Q(s)$ into a single real value.

Firstly, we should transform each QoS attribute into a real value between 0 and 1, which is called normalization. QoS attributes can be divided into two categories—negative attributes and positive attributes. For negative attributes (e.g., price), the lower the value, the higher the quality. On the contrary, for positive attributes (e.g., throughput), the higher the value, the higher the quality. Negative attributes can be easily converted into positive attributes by multiplying their values by -1. Therefore, this paper only considers the positive attributes. We define $R_k(s_{ij})$ as the normalized value of the kth QoS attribute of service s_{ij} and it can be calculated by Equation (1).

$$R_k(s_{ij}) = \frac{Q_{i,k}^{\max} - q_k(s_{ij})}{Q_{i,k}^{\max} - Q_{i,k}^{\min}} \tag{1}$$

$Q_{i,k}^{\max}$ and $Q_{i,k}^{\min}$ are the maximum and minimum values of the kth attribute of service class S_i. $q_k(s_{ij})$ is the value of the kth attribute of candidate service s_{ij}.

Secondly, the utility function of component service s ($s \in S_i$) is shown as Equation (2), where w_k represents the user's preference assigned to the kth QoS attribute.

$$U(s) = \sum_{k=1}^{r} R_k(s) \times w_k \tag{2}$$

The utility function of a composite service CS can be defined as Equation (3)

$$U(CS) = \sum_{k=1}^{r} \frac{Q_k^{\max} - Cq_k}{Q_k^{\max} - Q_k^{\min}} \times w_k \tag{3}$$

$Q_k^{\max}$ and $Q_k^{\min}$ are the maximum and the minimum values of the kth attribute for CS. According to the formulas in Table 1, they can be calculated by aggregating the maximum and the minimum values of the kth attribute of each service class.

4. Dynamic Service Selection Based on Global QoS Constraints Decomposition

The process of dynamic service composition in sequential composition structure is illustrated in Figure 1. There are n service classes. Each service class has m candidate services associated with r QoS attributes. The concrete component service is selected and bounded from each corresponding abstract service class. The composite service needs to satisfy two conditions, as follows.

(1) The utility function value of the composite service $U(CS)$ is maximum.

(2) The QoS aggregation for the composite service must satisfy the global QoS constraints, i.e., Cq_k meets C_k $(1 \le k \le r)$.

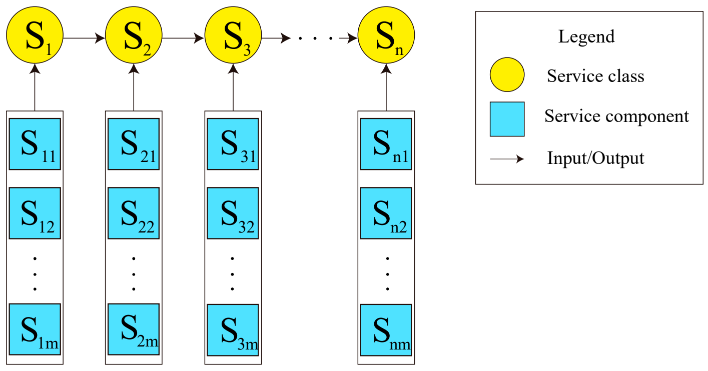

Figure 1. Dynamic service composition in sequential composition structure.

Recently, a number of service selection approaches based on global QoS constraints decomposition have been proposed, which can optimize the utility function value of the composite service while meeting the global constraints. As shown in Figure 2, these approaches basically have two components—global constraints decomposition and local service selection. Firstly, each global QoS constraint C_i is decomposed into n local constraints $c_1, c_2, \cdots, c_n$, where n is the number of service classes. Secondly, the local constraints are used to select the optimal component service. The local constraints need to ensure that, as long as they are achieved, the global constraints are achieved. Furthermore, the local constraints should be general enough, in order to avoid ignoring any possible candidate service.

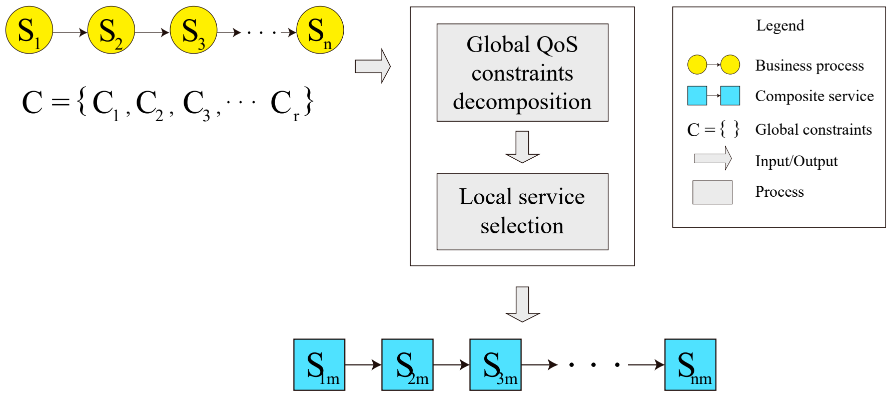

Figure 2. Service selection based on global QoS decomposition.

5. Adaptive Adjustment Approach Based on Fuzzy Logic

In this section, we propose an adaptive adjustment approach, which uses fuzzy logic technology to adjust the number of quality levels automatically.

5.1. The Initialization of Quality Level

Quality levels are initialized for each QoS attribute of each service class by dividing the value range of each QoS attribute into discrete quality values as shown in Figure 3. q_{ik}^m represents the kth QoS attribute value of the mth candidate service of service class S_i; L_{ik}^d indicates the dth quality level of the kth QoS attribute of service class S_i. It can be calculated as

$$L_{ik}^d = L_{ik}^{d-1} + d \times \Delta \tag{4}$$

$d \in [1,\ p_k]$, and p_k is the number of quality levels of the kth QoS attribute of service class S_i. Δ can computed as shown in Equation (5).

$$\Delta = \frac{q_{ik}^{\max} - q_{ik}^{\min}}{p_k} \tag{5}$$

$q_{ik}^{\max}$ and $q_{ik}^{\min}$ are, respectively, the maximum and minimum quality value of service class i for QoS attribute k, and $q_{ik}^{\min} = L_{ik}^{0}$.

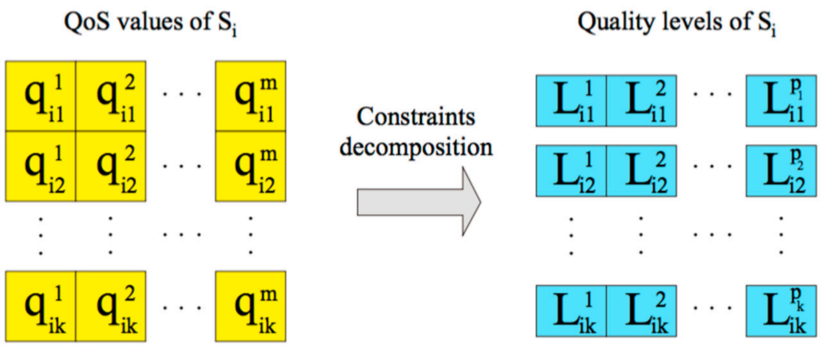
QoS values of S_i
Quality levels of S_i
Constraints decomposition
q_{i1}^{1} q_{i1}^{2} ... q_{i1}^{m}
q_{i2}^{1} q_{i2}^{2} ... q_{i2}^{m}
q_{ik}^{1} q_{ik}^{2} ... q_{ik}^{m}
L_{i1}^{1} L_{i1}^{2} ... $L_{i1}^{p_1}$
L_{i2}^{1} L_{i2}^{2} ... $L_{i2}^{p_2}$
L_{ik}^{1} L_{ik}^{2} ... $L_{ik}^{p_k}$

Figure 3. The initialization of quality level.

In this section, we exemplify the QoS constraints decomposition process by a composite service containing three sequential service classes with three QoS attributes. As shown in Table 2, each QoS attribute is associated to a weight and a constraint.

Table 2. Global QoS constraints and weights.

QoS	k_1 **= Price**	k_2 **= Response Time**	k_3 **= Throughput**
Weight	0.2	0.5	0.3
Constraint	≤ 140	≤ 400	≥ 75

We assume that the number of quality levels of each QoS attribute of each service class is 3. Figure 4 describes the procedure of the constraints' decomposition. For service class S_1, the global price constraint is divided into three local constraints $-35 \leq k_1 \leq 40$, $40 \leq k_1 \leq 45$, and $45 \leq k_1 \leq 50$.

QoS matrix		k_1	k_2	k_3
S_1	s_{11}	35	120	85
	s_{12}	48	90	85
	s_{13}	50	99	70
S_2	s_{21}	34	132	96
	s_{22}	61	140	91
	s_{23}	50	145	85
	s_{24}	44	156	81
S_3	s_{31}	48	110	86
	s_{32}	36	110	98
	s_{33}	60	149	98

Quality level matrix		d_0	d_1	d_2	d_3
S_1	k_1	35	40	45	50
	k_2	90	100	110	120
	k_3	70	75	80	85
S_2	k_1	34	43	52	61
	k_2	132	140	148	156
	k_3	81	86	91	96
S_3	k_1	36	44	52	60
	k_2	110	123	136	149
	k_3	86	90	94	98

Figure 4. The procedure of the constraints' decomposition.

5.2. General Fuzzy Logic System

Fuzzy logic is an approximate reasoning technology based on multi-valued logic, which is suitable to deal with uncertainty [29]. It is able to support decision-making and evaluate uncertain parameters. In the existing researches [30–33], fuzzy logic technology has been used for service ranking in the process of service selection. In this paper, we propose an adaptive adjustment method for the number of quality level based on fuzzy logic (AAQL).

The general fuzzy logic system (FLS) includes four basic parts—fuzzification, fuzzy rule base, fuzzy inference, and defuzzification, as shown in Figure 5. The first step is fuzzification, in which crisp inputs are mapped into the fuzzy set. The fuzzy set can be defined by the membership function. Assuming that U is a normal set, any function $f()$ that can map U to [0,1], can determine a fuzzy set A. $f()$ is called the membership function. For any $u \in U$, $f(u)$ is the membership of u, which describes the degree of u belonging to U. The next step is fuzzy inference, which determines what extent each rule in the fuzzy rule base applies to the current fuzzified inputs. IF–THEN rule is the common fuzzy rule to represent human knowledge in the general FLS. The IF part includes memberships of attributes of an individual, and the THEN part is a special concept called Rank. A fuzzy rule shows which composition of attributes a user is willing to accept to which degree, where attributes and degree of acceptance are vague [34]. A simple IF–THEN fuzzy rule can be:

IF Price = Expensive and Time = Slow THEN Rank = Bad.

Finally, the fuzzified output is converted into a crisp output. This step is called "defuzzification".

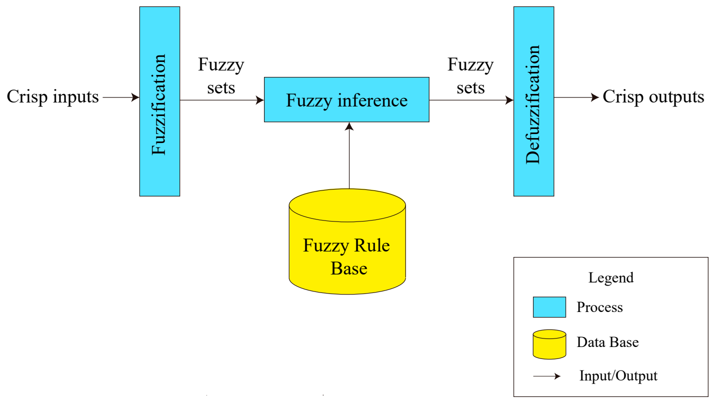

Figure 5. General fuzzy logic system.

5.3. Adaptive Quality Level Based on Fuzzy Logic

AAQL can automatically adjust the number of quality levels according to user's preferences [35], utility function values, and time cost. We assume that there are two input variables and one output variable. The input variables are the normalized number of quality levels of the kth QoS attribute Np_k and the normalized function value of the quality levels NFp. The output variable is cp_k, the change ratio of Np_k. Np_k and NFp can be defined as below:

$$Np_k = \frac{p_k^{\max} - p_k}{p_k^{\max} - p_k^{\min}},\ NFp = \frac{Fp_{\max} - Fp}{Fp_{\max} - Fp_{\min}} \tag{6}$$

where $p_k^{\max}$ or $p_k^{\min}$ is the maximum or the minimum number of quality levels and p_k is the current number of quality levels. $Fp_{\max}$ or $Fp_{\min}$ is the maximum or the minimum function value of the current quality levels. Fp is the current function value of the quality levels, and it can be calculated as Equation (7).

$$Fp = \eta_k \times \frac{NU}{NT} \tag{7}$$

η_k is the user's preference for the kth attribute, and $\eta_k \in (0,1)$. η_k can be dynamically obtained and normalized through the method in Reference [29]. Both NU and NT are the normalization value $NU \in (0,1)$ and $NT \in (0,1)$, and they can be calculated by Equation (8).

$$NU = \frac{U_{\max} - U_{cur}}{U_{\max} - U_{\min}},\ NT = \frac{T_{\max} - T_{cur}}{T_{\max} - T_{\min}} \tag{8}$$

$U_{\max}$ and $U_{\min}$ are, respectively, the maximum and the minimum utility function values of the composite service under the current quality levels. U_{cur} is the current utility function value. $T_{\max}$ and $T_{\min}$ are, respectively, the maximum and the minimum time cost of the process of quality level initialization and quality level combination. T_{cur} is the current time cost.

Firstly, AAQL maps the input variables NFp and Np_k into the fuzzy set. We assume that there are three fuzzy labels—L, M, and B, where L denotes "Little", M denotes "Middle", and B denotes "Big". This paper uses the triangular and trapezoidal shapes to define membership functions of NFp, Np_k, and cp_k, as shown in Figure 6.

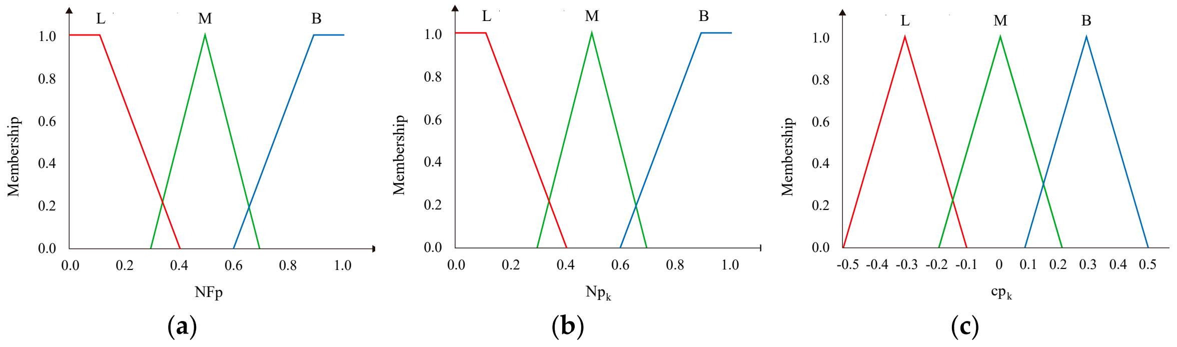

Figure 6. The membership function of NFp (**a**), Np_k (**b**), and cp_k (**c**).

Secondly, according to the fuzzy rules, AAQL uses the fuzzy inference to get the fuzzy output variable cp_k. The nine fuzzy rules of AAQL are as listed in Table 3, and the surface projection is shown as Figure 7. The basic fuzzy inference algorithm is Min–Max algorithm. However, the IF part consists of multiple attributes, which have different effects on the conclusion. For the Min algorithm, the conclusion is only determined by the attribute with the least membership. No matter how the membership of the other attributes change, as long as the minimum membership remains unchanged, the conclusion does not change. That is to say, Min–Max algorithm is insensitive to the changes of input facts. As a result, the common Min–Max algorithm is not suitable for AAQL. In this paper, we assign weights to NFp and Np_k, respectively. Based on the weighted sum of NFp and Np_k, we can infer the membership of the corresponding conclusion. Obviously, the weights are very important. At present, there is no definite method to get the exact weights. We designed a number of experiments by setting different user preferences and other parameters, and we conducted experiments to search for the optimal number of quality levels for each experiment. By summarizing the experimental results, we assign the weight of NFp 0.7 and the weight of Np_k 0.3. Obviously, as the weights are set empirically, they are still not accurate. In our future work, we will improve the method for assigning the weights.

Table 3. The fuzzy rules.

IF		**THEN**
NFp	Np_k	cp_k
B	B	M
B	M	M
B	L	M
M	B	L
M	M	L
M	L	B
L	B	L
L	M	L
L	L	B

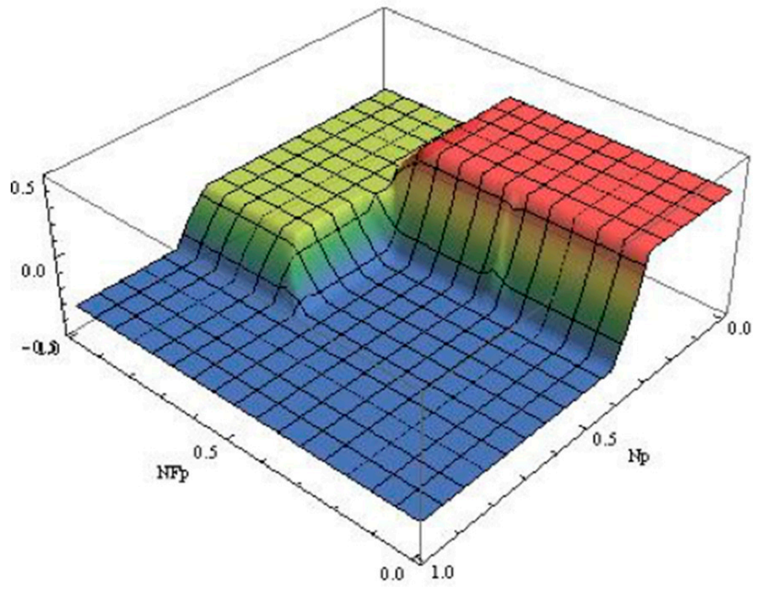

Figure 7. The surface projection of fuzzy rules.

This paper adopts the center of gravity (COG) defuzzification method [36] to get the accurate and concrete value of cp_k. According to cp_k, the new number of quality levels can be adaptively adjusted as Equation (9), where Np_k^{new} is the new Np_k and Np_k^{old} is the original one.

$$Np_k^{new} = Np_k^{old} \times (1 + cp_k) \quad (9)$$

For example, we assume that $NFp = 0.63$ and $Np_k = 0.37$. The calculation process of cp_k is as follows:

(1) Fuzzification of the input variables.

According to the membership functions in Figure 6, we can get the membership values of NFp and Np_k, as shown in Tables 4 and 5.

Table 4. The membership values of NFp.

Fuzzy Label	Membership
L	0
M	0.35
B	0.1

Table 5. The membership values of Np_k.

Fuzzy Label	Membership
L	0.1
M	0.35
B	0

(2) Invoking the corresponding fuzzy rules.

When the fuzzy label of NFp is L, the membership value is 0, and when the fuzzy label of Np_k is B, the membership value is 0. So, the fuzzy rules in Table 4, of which IF parts contain the fuzzy label of NFp is L or the fuzzy label Np_k is B, are not be invoked. Finally, there are four rules invoked, which are listed in Table 6.

Table 6. The activated rules.

IF		THEN
NFp	***Np$_k$***	***cp$_k$***
M	L	B
M	M	L
B	L	M
B	M	M

Rule 1: The membership value of NFp belonging to "M" is 0.35, and the membership value of Np_k belonging to "L" is 0.1, so the degree of Rule 1 can be calculated as: $degree(R1) = 0.35 \times 0.7 + 0.1 \times 0.3 = 0.275$.

Rule 2: The membership value of NFp belonging to "M" is 0.35, and the membership value of Np_k belonging to "M" is 0.35, so the degree of Rule 2 can be calculated as: $degree(R2) = 0.35 \times 0.7 + 0.35 \times 0.3 = 0.35$.

Rule 3: The membership value of NFp belonging to "B" is 0.1, and the membership value of Np_k belonging to "L" is 0.1, so the degree of Rule 3 can be calculated as: $degree(R3) = 0.1 \times 0.7 + 0.1 \times 0.3 = 0.1$.

Rule 4: The membership value of NFp belonging to "B" is 0.1, and the membership value of Np_k belonging to "M" is 0.35, so the degree of Rule 4 can be calculated as: $degree(R4) = 0.1 \times 0.7 + 0.35 \times 0.3 = 0.175$.

The conclusion of Rule 1 is the fuzzy label of cp_k is B, so the membership value of cp_k belonging to "B" is 0.275. For Rule 2, the fuzzy label of cp_k is L, so the membership value of cp_k belonging to "L" is 0.35. For Rule 3 and Rule 4, the fuzzy label is M, so the membership value of cp_k belonging to "M" is $\max(0.1,\ 0.175) = 0.175$.

(3) Defuzzification of the output variable.

$$cp_k = \frac{0.3 \times 0.275 + (-0.3) \times 0.35 + 0 \times 0.175}{0.275 + 0.35 + 0.175} \approx -0.03$$

6. Global QoS Constraints Decomposition Based on CGA

6.1. Near-Optimal Quality Level Scheme

The nature of the optimal global QoS constraints decomposition is to search for an optimal quality level combination for each service class. Each service class has a number of QoS attributes and each QoS attribute has a set of quality levels, so there are many quality level schemes. It is a challenging problem to search for the optimal quality level scheme. It is a constrained multi-objected combinatorial optimization problem [37,38]. For large-scale searching space, in order to deal with this problem quickly, we use a novel evolution CGA by integrating Genetic Algorithm into the framework of Cultural Algorithm [1].

6.2. Cultural Genetic Algorithm

On account of parallelism and effective utilization of global information, GA is very suitable to solve combinatorial optimization problems. However, the prematurity phenomenon of GA is the biggest shortcoming. In order to assure the whole astringency, some approaches should be proposed to obtain the global optimal solution.

Cultural Algorithm (CA) is a new evolutionary algorithm to simulate the evolution of human society, proposed by Reynolds in 1994. As shown in Figure 8, CA is mainly composed of three parts—belief space, population space, and operation function. Belief space and population space can evolve at different speeds, which are relatively independent and interrelated. At the micro level, the population space simulates biological evolution and uses the evolutionary function PEvolve () to produce experiential knowledge. The evaluation function Evaluate () evaluates the information of the individuals in the population space, and then the useful information is passed to the belief space by function Accept (). Finally, the selection function Select () gives the optimal or approximate optimal solution. At the macro level, belief space uses Update () to receive, integrate, and preserve the empirical knowledge transmitted by the population space. Finally, the useful empirical knowledge is transferred to the population space by Influence (), so as to guide the evolution of individuals in the population space.

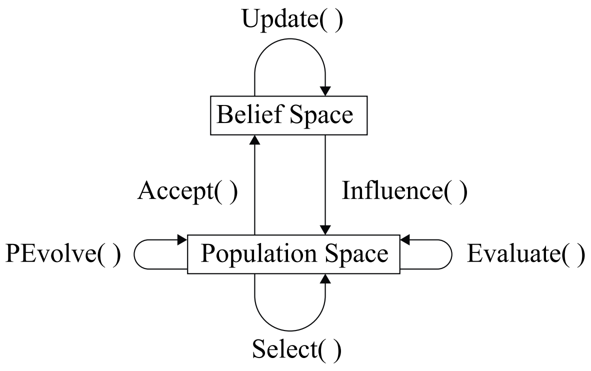

Figure 8. The framework for Cultural Algorithm (CA).

To solve the constrained combinatorial optimization problem, it is the key problem how to transfer the constraints of the problem to the empirical knowledge and store it in the belief space. CA divides the searching space into several regions with different characteristics by using the constraint conditions.

We divide the searching space into smaller areas called cells. Some units are completely in the feasible domain, some units are completely in the infeasible domain, and the others are in both the feasible domain and infeasible domain. These units are the basic units that constitute the belief space of CA, called Belief Cells (BC). The belief space uses BC to represent and store the constraint conditions of the problem, in order to guide the evolution in the population space.

We take a constrained combinatorial optimization problem with two independent variables as an example. As shown in Figure 9, the solution space is a region in a two-dimensional plane. By several searching operations, a region that most likely produces excellent individuals is generated. We use a curve to represent the constraint boundary. The outer part of the curve is the feasible region and the inner part is the infeasible region. As shown in Figure 9b, the whole region is divided into smaller regions, which are BC. The BC belonging to the feasible region and infeasible region are represented by white and black squares, respectively, and the BC belonging to semi-feasible region are represented by gray squares. Constraint boundary and its internal constraint knowledge are saved as empirical knowledge to guide the population evolution in the population space, so that more individuals are generated in the feasible domain and semi-feasible domain, and the generation of individuals in the infeasible domain is restrained.

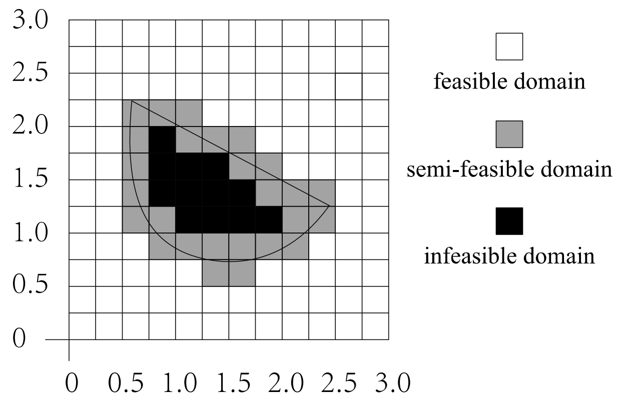

Figure 9. Expression of constraints in area.

For an n-dimensional belief space containing m BC, its structure is defined as below:

$$BS = \langle N[n], \quad C[m] \rangle$$

$N[n]$ is the regional information set, and $N[j]$ represents the regional information of dimension j; $C[m]$ is the set of BC, and $C[i]$ is the ith belief unit.

$N[j]$ can be represented as follows:

$$N[j] = \langle I_i, \quad L_j, \quad U_j \rangle$$

I_j is a closed interval that represents the value range of independent variables in the region of dimension j. $I_j = [l_j, u_j] = \{x | l_j \le x \le u_j, x \in R\}$, where l_j is the upper limit value, and u_j is the lower limit value. L_j and U_j are presented the evaluation function value of l_j and u_j, respectively.

$C[i]$ can be represented as follows:

$$C[i] = \langle Class_i, \quad Cnt1_i, \quad Cnt2_i \;, \quad W_i, \quad Pos_i, \quad size_i \rangle$$

$Class_i$ represents characteristics of the ith belief cell, and the value set of $Class_i$ is $S = \{0, 1, 2, 3\}$. 0 is feasible, 1 is not feasible, 2 is semi-feasible, and 3 is unknown. $Cnt1_i$ and $Cnt2_i$ are two counters that record the number of feasible and infeasible individuals in $C[i]$, respectively. W_i is the weight of belief cell. The higher the credibility, the higher the weight of the belief cell. Pos_i is the left-most angular position coordinate of the ith belief unit and $size_i$ represents the size of the belief unit.

GA realizes population evolution by iterative operation through genetic operations such as selection, crossover, and mutation. This process is unguided and completely randomly. It ignores the important influence of feature information generated in the process of population evolution on solving problems. As a result, we take GA as the evolutionary algorithm of population space in CA and proposes knowledge guidance-based GA, namely Cultural Genetic Algorithm. CGA retains the selection and crossover operations of GA, and guides mutation operations with the empirical knowledge of the belief space, so as to effectively improve the convergence speed and the quality of results and reduce the computation time as well.

6.3. Global QoS Constraints Decomposition Algorithm Based on CGA

Suppose a composite service contains n service classes and m QoS attributes, and each QoS attribute is divided into t constraint partitions. The flow chart of CGA is as shown in Figure 10. The optimal global QoS constraint decomposition algorithm based on CGA mainly includes two parts—the renewal of belief space and the evolution of population space.

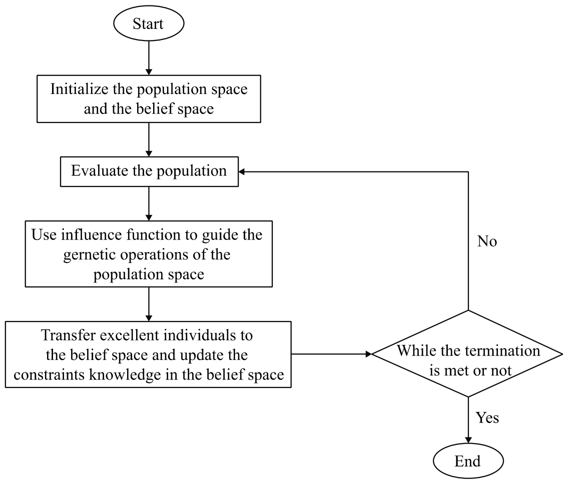
Start
Initialize the population space and the belief space
Evaluate the population
Use influence function to guide the gernetic operations of the population space
Transfer excellent individuals to the belief space and update the constraints knowledge in the belief space
While the termination is met or not
No
Yes
End

Figure 10. The flow chart of Cultural Genetic Algorithm (CGA).

6.3.1. Belief Space Renewal

Belief space can be regarded as a hypercube composed of belief units in n-dimensional space. Through the functions Accept () and Update (), the belief space receives and integrates the empirical knowledge generated by the evolution of individuals in the population space, so as to constantly update the size of the hypercube.

The function Accept () sorts the individuals in the current population according to the size of their objective function values, and transfers the excellent individuals with large objective function values to the belief space. The number of individuals transferred to Accept () changes with the generations of population evolution. If the value of the overall objective function of the current population is greater than that of the previous generation, the number of individuals to be transferred will decrease. If it remains unchanged, the number of individuals to be transferred will remain unchanged. If it is less than that of the previous generation, the number of individuals to be transferred will increase. The number of excellent individuals transferred by the function Accept () in the paper is expressed by n_{accept}. The specific calculation formula is as follows:

$$n_{accept} = N \times p + \frac{h \times (1-p) \times N}{g} \tag{10}$$

N represents the number of individuals in the population; p is a parameter set up according to experience, generally 0.2. g is the algebra of evolution; h is a multiple of expansion. If the value of the overall objective function of the current population $P(t)$ is larger than that of the population $P(t-1)$ of the previous generation, or if the value of the overall objective function is constant, then h = 1. Otherwise, h = 2; that is, the number of selected individuals is increased. With the increase of population evolution algebra, when the population gradually converges to the optimal solution or approximate optimal solution, the computation amount and search time are gradually reduced, and the number of individuals is increased when the value of the objective function is not ideal.

According to the individual experience knowledge transferred by Accept (), Update () updates the belief space $BS = \langle\ N[n],\ C[m]\ \rangle$. Suppose that the ith outstanding individual X_i transferred in generation t determines the lower limit of the independent variable value of $N[j]$, and the pth outstanding individual X_p determines the upper limit of the independent variable value of $N[j]$. Update $N[j]$ by the following calculation formula:

$$l_j^{t+1} = \begin{cases} x_{i,j}^t, & \text{if } x_{i,j}^t \le l_j^t \quad \text{or} \quad Evaluate(x_i^t) < L_j^t \\ L_j^t, & \text{otherwise} \end{cases} \tag{11}$$

$$L_j^{t+1} = \begin{cases} Evaluate(x_i^t), & \text{if } x_{i,j}^t \le l_j^t \quad \text{or} \quad Evaluate(x_i^t) < L_j^t \\ L_j^t, & \text{otherwise} \end{cases} \tag{12}$$

$$u_j^{t+1} = \begin{cases} x_{p,j}^t, & \text{if } x_{p,j}^t \le u_j^t \quad \text{or} \quad Evaluate(x_p) < U_j^t \\ U_j^t, & \text{otherwise} \end{cases} \tag{13}$$

$$U_j^{t+1} = \begin{cases} Evaluate(x_p), & \text{if } x_{p,j} \le u_j^t \quad \text{or} \quad Evaluate(x_p) > U_j^t \\ U_j^t, & \text{otherwise} \end{cases} \tag{14}$$

Among them, the $x_{i,j}^t$ and $x_{p,j}^t$ represent the jth independent variable of X_i and X_p in the tth generation; and l_j^t and u_j^t, respectively, represent lower limit and upper limit of independent variables in the tth generation of L_j^t and U_j^t, respectively, represent the utility function value of l_j^t and u_j^t.

6.3.2. Population Space Evolution

CGA takes GA as the evolutionary algorithm for population evolution in population space, which mainly includes the following key steps:

1. Encoding

The purpose of global QoS constraint decomposition is to find an appropriate constraint partition for each QoS attribute of each service class. A service composition scheme contains a number of service classes. There is a corresponding constraint partition set for each service class. Therefore, this paper designs a two-dimensional coding method to decompose a real, global QoS constraint. As shown in Figure 11, l^{x}_{nr} represents the xth constraints division of the rth QoS attribute Cq_r of service class S_n. Each column represents constraints division set of a service class.

$$\begin{array}{c|ccccc|}
 & S_1 & S_2 & S_3 & \cdots & S_n \\
\hline
Cq_1 & l^{a}_{1,1} & l^{e}_{2,1} & l^{i}_{3,1} & \cdots & l^{m}_{n,1} \\
Cq_2 & l^{b}_{1,2} & l^{f}_{2,2} & l^{j}_{3,2} & \cdots & l^{o}_{n,2} \\
Cq_3 & l^{c}_{1,3} & l^{g}_{2,3} & l^{k}_{3,3} & \cdots & l^{p}_{n,3} \\
\vdots & \vdots & \vdots & \vdots & & \vdots \\
Cq_r & l^{d}_{1,r} & l^{h}_{2,r} & l^{l}_{3,r} & \cdots & l^{q}_{n,r} \\
\hline
\end{array}$$

Figure 11. The encoding method.

2. Evaluation Function

The evaluation function of CGA is to calculate the individual objective function value in the population. The specific calculation formula is as follows:

$$Evaluate() = \max \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{d=1}^{t} l^{d}_{i,k} \times x^{d}_{i,k} \times w_k \tag{15}$$

Among them, $l^{d}_{i,k}$ represents the dth constraint division of the kth QoS attributes of service class. $x^{d}_{i,k}$ represents if $l^{d}_{i,k}$ is selected or not, if selected $x^{d}_{i,k} = 1$ or $x^{d}_{i,k} = 0$; w_k represents the user preference for the kth QoS attribute. C_k represents the constraint condition for attribute k of composite service, and the evaluation function needs to meet the following conditions:

$$\forall k : \sum_{i=1}^{n} \sum_{d=1}^{t} l^{d}_{i,k} \times x^{d}_{i,k} \geq C_k,\ 1 \leq k \leq m \tag{16}$$

$$\forall i, k : \sum_{d=1}^{t} x^{d}_{i,k} = 1,\ 1 \leq k \leq m \tag{17}$$

3. Selection Operation

According to the law of biological evolution, the larger the value of the objective function is, the greater the chance that individuals will be retained. That is to say, the probability of individuals being selected is directly proportional to the value of the objective function. The roulette selection method is adopted for individual selection operation. Let the population size be N, and the probability of the ith individual X_i being selected p_i is:

$$p_i = \frac{Evaluate(X_i)}{\sum_{i=1}^{N} Evaluate(X_i)} \tag{18}$$

4. Crossover and Mutation Operation

After the selection operation, we select two individuals from the current population and perform the following operations:

$$X_i^{t+1} = \alpha \times X_i^t + (1-\alpha) \times X_j^t \tag{19}$$

$$X_j^{t+1} = \alpha \times X_j^t + (1-\alpha) \times X_i^t \tag{20}$$

Among them, X_i^t and X_j^t are two individuals of the tth generation, X_i^{t+1} and X_j^{t+1} are new individuals, and $a \in (0,1)$ is a random number. Choose one individual X_i^t in the current population as parent individual. Its mutation operation is shown as follows, where X_i^{t+1} is a new individual, β is mutation step length, and d is the variation direction.

$$X_i^{t+1} = X_i^t + \beta \times d \tag{21}$$

5. Mutation Operation Based on Influence ()

New individuals generated by the basic mutation operation based on Equation (21) often violate the constraint conditions. In this paper, a mutation operation guided by the function Influence () in CA is used to generate new individuals. The specific calculation formula is divided into two situations:

- If the parent individual X_i^t belongs to the feasible region or semi-feasible region, it continues mutating near the feasible region. Its computation formula is as follows:

$$x_{i,j}^{t+1} = x_{i,j}^t + \gamma \times \left(u_j^t - l_j^t\right) \times a \tag{22}$$

  Among them, the $x_{i,j}^t$ is the jth independent variable of the tth generation individual X_i; γ is a artificial positive value; u_j^t and l_j^t, respectively, respect the upper limit and lower limit of the independent variable of tth generation for $N[j]$; a represents a random value between 0 and 1.
- If the parent individual X_i^t belongs to the infeasible region, we adopt the concept of the sliding window based on interval, X_i^t move to four BC according to different probability. The computation formula is as follows:

$$x_{i,j}^{t+1} = moveTo(choose(C[m])) \tag{23}$$

  moreTo () is a migration function that moves the parent generation of an unfeasible domain to the selected target belief unit. $choose(C[m])$ represents that according to the probability W_i of each BC $C[i]$, selecting the target BC by the roulette selection method for *moreTo* () invoking. When the ith BC is selected, the formula of *moreTo* () is as follows:

$$X_i^{t+1} = Left_i + uniform(0,1) \times size_i \tag{24}$$

  $Left_i$ is a $1 \times n$ array, representing the most left position of the BC $C[i]$; $size_i$ is a $1 \times n$ array, representing the size of BC $C[i]$ in each dimension. $uniform(0,1)$ is a $1 \times n$ array generated according to uniform distribution.

The pseudo code of the optimal global QoS constraints decomposition based on CGA is listed in Algorithm 1. For the scenario in Figure 4, we apply Algorithm 1 to obtain the near-optimal constraints decomposition scheme. The quality level matrix is the input set and the final result is shown in Table 7.

Algorithm 1 The optimal global QoS constraints decomposition based on CGA

```
Input: The initial quality partition set of each service class
Output: The optimal global QoS constraints decomposition scheme
1. Initialize the population space
2. Initialize the belief space
3. Do
4. Evaluate the individuals in the population space by Equation (15)
7. Do selection operation by Equation (18)
8. Do crossover operation by Equations (19) and (20)
9. Do mutation operation by Equations (22) and (23)
10. Calculate the number of excellent individuals transferred to the belief space Equation (10)
11. Update the constraints knowledge in the belief space by Equation (11)–(14)
10. While the termination condition is not met
11. Return the optimal constraints decomposition scheme
```

Table 7. Optimal quality level combination.

Optimal Quality Level	k_1	k_2	k_3
S_1	50	100	75
S_2	52	148	86
S_3	44	123	98

Therefore, the global QoS constraints in Table 2 are decomposed into local ones for each service class as shown in Table 8.

Table 8. Local QoS constraints.

QoS	k_1 = Price	k_2 = Response Time	k_3 = Throughput
S_1	$45 \leq k_1 \leq 50$	$90 \leq k_2 \leq 100$	$75 \leq k_3 \leq 80$
S_2	$43 \leq k_1 \leq 52$	$140 \leq k_2 \leq 148$	$81 \leq k_3 \leq 86$
S_3	$36 \leq k_1 \leq 44$	$110 \leq k_2 \leq 123$	$94 \leq k_3 \leq 98$

7. Local Service Selection

The local constraints are used to select component services from each service class in parallel. By the local constraints, we can reduce the number of the candidate services in each service class. In the reduced set of candidate services, the service with the biggest utility is selected as the final local component service. The utility function of component service can be defined as follows:

$$U(s_{ij}) = \sum_{k=1}^{r} \frac{Q_{i,k}^{\max} - q_{ij}^{k}}{Q_k^{\max} - Q_k^{\min}} \times w_k \tag{25}$$

where q_{ij}^k is the kth QoS attribute value of service s_{ij}, $Q_{i,k}^{\max}$ is the maximum value of the kth QoS attribute values of service class S_i, and w_k is the weight of the kth QoS attribute. Note that in Equation (25), we use the global utility function $Q_k^{\max} - Q_k^{\min}$ as the denominator, instead of the local utility function $Q_{i,\,k}^{\max} - Q_{i,\,k}^{\min}$ in Equation (1), so as to make the utility functions of the candidate services focus on global attributes.

According to the local constraints in Table 8, for service class S_1, only the candidate service s_{13} can meet the local constraints. Therefore, the candidate service s_{13} is optimal to be bound to service class S_1. Finally, the service combination (s_{13}, s_{23}, s_{32}) is generated as the optimal service composition scheme.

8. Experimental Evaluation

So as to evaluate the performance of the proposed approach (ACQD), we compare it with the integer programming-based approach (WS-IP) [16] and the QoS constraints decomposition approach based on CGA (QCD), in which the number of quality level is constant. Integer programming is widely applied for service selection [8,9,16].

Assuming that a service composition process in sequential structure contains six service classes, each service class has m candidate services and m varies from 100 to 1000. The data set consists of two parts: (1) QWS, a data set containing 2508 real Web services with 10 QoS attributes [39], and (2) RQWS, a data set artificially simulated and expanded according to QWS by Eclipse programming tool. In RQWS, the attributes and their ranges of services are the same as those of QWS. There are three global QoS attributes including price, response time, and availability. For RQWS, the QoS attribute values for each Web service are randomly generated, following Reference [40]. Table 9 shows the global QoS constraints and user's QoS preferences in detail.

Table 9. Global QoS constraints and user's QoS preferences.

Item	Price	Response Time	Availability
constraints	$<$ 140 dollars	$<$ 130 s	$>$ 0.7
preferences	0.45	0.3	0.25

In our experiments, WS-IP was implemented by the open source system LpSolve (lp_solve version 5.5). For QCD and AQCD, we implemented them by Microsoft's Visual C. Net and the rates of crossover and mutation operators were set as 0.80 and 0.10, respectively. All of the experiments were conducted on a PC with an Inter Core i5 (1.6 GHz) CPU and 4 GB RAM.

We evaluate the above approaches according to two metrics—running time and approximation ratio.

- Running time: the CPU time consumed by an algorithm.
- Approximation ratio: the ratio of the global utility achieved by an algorithm to the optimal utility. It can be computed as follows:

$$approximation\ ratio = \frac{U(CS)}{U_{optimal}(CS)} \tag{26}$$

$U(CS)$ represents the utility function value and $U_{optimal}(CS)$ denotes the optimal utility function value. The optimal utility function value is generated by WS-IP. As a result, the approximation ratio of WS-IP is 1.

There are two experiments to evaluate the performance of AQCD. The first experiment evaluates the adaptability of AQCD compared with QCD and WS-IP, which consists of two parts. The first part evaluates how the number of quality levels affects QCD, in order to verify the importance of adjusting the number of quality levels. By analyzing the experimental results, the optimal number of quality number of quality levels is determined. The second part evaluates the performance of AQCD compared with QCD and WS-IP with different user's preferences. Different user's preferences or different numbers of candidate services per service class are set for every experiment. The average value of each experiment running 50 times is used to evaluate the method's performance.

8.1. Evaluation of Adaptability

In this section, we firstly verify the importance of the number of quality levels to the performance of QCD. Then, we compare the performance of AQCD, QCD, and WS-IP with different user preferences to evaluate the adaptability of our approach AQCD.

8.1.1. The Number of Quality Level for QCD

The number of candidate services in each service class is set as 100 and the number of quality levels varies from 10 to 40. The running time and approximation ratio of QCD are recorded as shown in Figures 12 and 13.

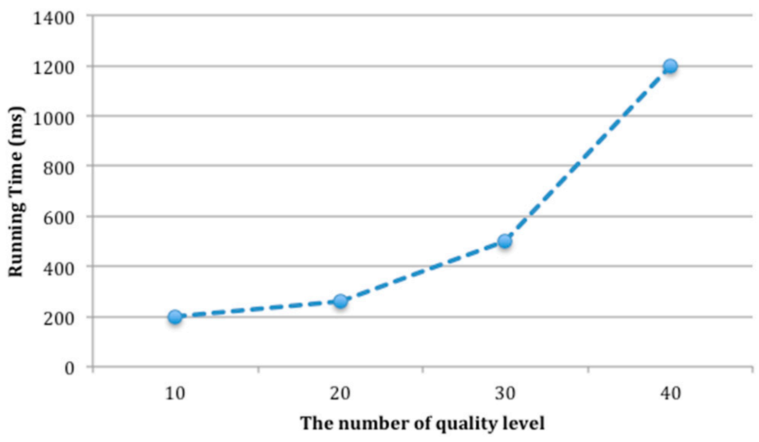

Figure 12. Running time of QCD with different number of quality level.

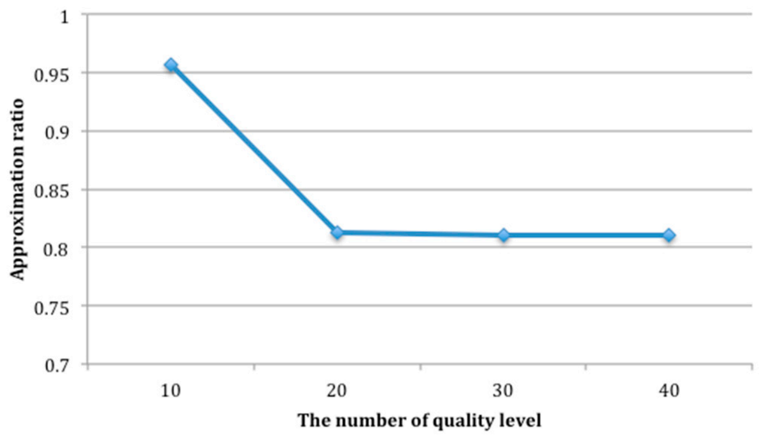

Figure 13. Approximation ratio of QCD with different number of quality level.

As shown in Figures 12 and 13, the number of quality levels has a prominent effect on the performance of service selection. In Figure 12, the running time increases rapidly with the increasing of the number of quality levels. Figure 13 indicates that when the number of quality level is 10, QCD can not only run fast, but also can obtain better solution. In other words, the optimal number of quality levels is 10, which is set for QCD.

8.1.2. Adaptability

So as to evaluate the adaptability of AQCD, we compare the performance of AQCD and other methods by changing the user's preferences. We assume that there are 10 sets of user's preferences. Among them, the user's preferences in set 1 are set as shown in Table 9, and other sets are randomly generated. There are 100 candidate services in each service class and the number of quality levels for QCD is set as 10. The running time and approximation ratio of AQCD, QCD, and WS-IP with different user's preferences are shown in Figures 14 and 15.

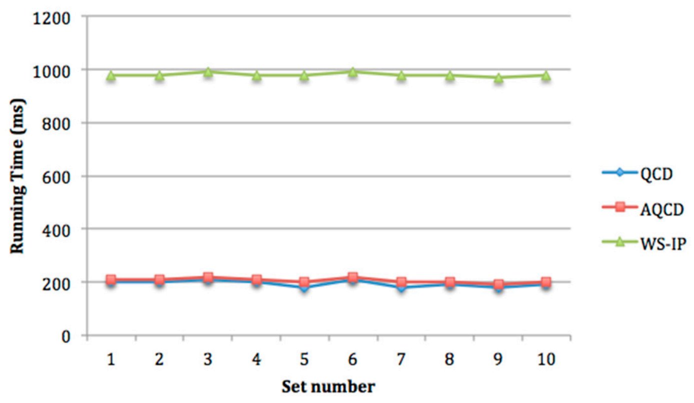

Figure 14. Running time of QCD, AQCD, and WS-IP with different user's preferences.

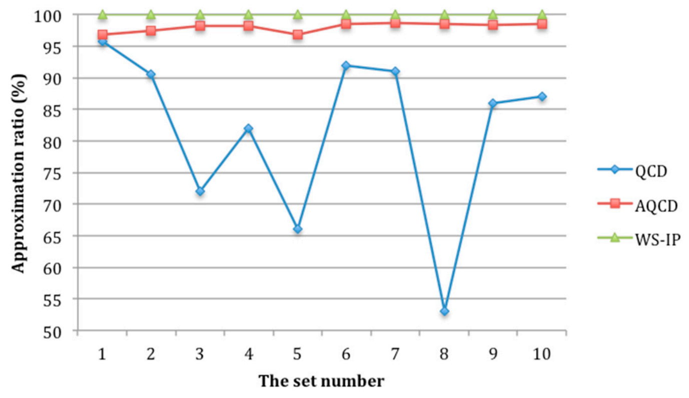

Figure 15. Approximation ratio of AQCD, QCD, and WS-IP with different user's preferences.

As shown in Figure 14, the running time of AQCD, QCD and WS-IP is almost stable with different user's preference. The running time of AQCD is close to that of QCD all the time, and they are significantly shorter than WS-IP. Figure 15 shows that the approximation ratio of AQCD is better than QCD and close to the optimal method WS-IP. Furthermore, the approximation ratio of AQCD is stable with different user's preferences. However, QCD suffers from user's preferences. The approximation ratio of QCD changes with different user's preferences.

8.2. Evaluation of Scalability

In order to evaluate the scalability of the proposed approach AQCD, we set the number of candidate services per service class from 100 to 1000. The experimental results of running time and approximate ration are shown in Figures 16 and 17.

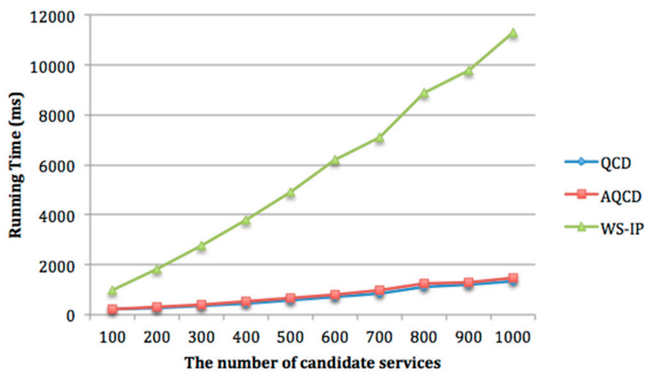

Figure 16. Running time of QCD, AQCD, and WS-IP with different numbers of candidate services.

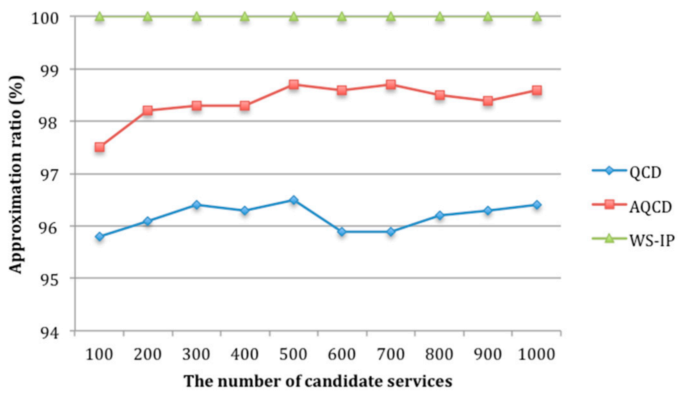

Figure 17. Approximation ratio of QCD, AQCD, and WS-IP with different number of candidate services.

Figure 16 shows that with the increasing of candidate services, the running time of WS-IP is far more than that of AQCD and QCD. The running time of AQCD and QCD increases slowly. For a composite service composition with n service classes, each service class contains m candidate services and each service consists of r QoS attribute. Each attribute can be divided into t quality levels. In the worst case, the time complexity of WS-IP is $O(2^{n\times m})$. The running time of AQCD and QCD consists of two parts, the time of global constraints decomposition and local service selection. Among them, the local service selection is to select the component service for each service class in parallel. Its time complexity is $O(m)$. Therefore, the time complexity of AQCD and QCD mainly comes from the decomposition process of global constraints. The system needs to select one scheme from $n \times r \times t$ possible schemes. The time complexity is $O(n \times r \times t)$, which is independent of the number of candidate services m. Therefore, when the number of quality levels t is less than m/r, the search space of AQCD and QCD is small. The running time of AQCD is slightly higher than QCD because it needs to adjust the number of quality level adaptively. However, this time is negligible. Figure 17 indicates that the approximation ratio of AQCD is higher than QCD and close to WS-IP. The approximation ratios of all the test cases are higher than 97%.

9. Conclusions

In this paper, a dynamic service selection approach based on adaptive global QoS constraints decomposition is proposed. We divided the global QoS constraints into local ones, by which the optimal component service for each service class is selected. We use fuzzy logic to automatically adjust the number of quality level considering the user's preferences. In order to improve the efficiency of service selection, we use the Culture Genetic Algorithm to solve the near-optimal global constraints decomposition schema. Based on the real dataset QWS and the artificial generated dataset RQWS, we have evaluated the performance of the proposed approach AQCD in comparison with QCD and WS-IP. The experimental results show that the number of quality levels has a prominent effect on the performance of service selection. By adjusting the number of quality levels adaptively, the approximation ratio of AQCD can be stable with uncertain user's preferences. With the increase in candidate services, AQCD can achieve near-optimal solutions with slowly increasing time, which is far less than that of WS-IP. In conclusion, our AQCD approach can achieve the near-optimal solution and significantly reduce the computation time. In addition, AQCD has good adaptability and scalability.

In our future work, we will increase the number of fuzzy sets and formulate more appropriate fuzzy rules in order to enhance the inference ability and manage complex situations; we will also apply the constraints decomposition approach to the problem of QoS-aware service composition in distributed environment where QoS values are maintained by a group of distributed QoS registries.

Author Contributions: Y.Y. conceived the research subject and wrote the paper. W.Z., X.Z., and H.Z. contributed to the revisions.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 51479021, 51679025), Liaoning Provincial Natural Science Foundation of China (Grant No. 20170520196), the Fundamental Research Funds for the Central Universities (Grant No. 3132016308, 3132018197).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, Z.Z.; Xue, X.; Shen, J.Q.; Li, W.R. Web service dynamic composition based on decomposition of global QoS constraints. *Int. J. Adv. Manuf. Technol.* **2013**, *69*, 2247–2260. [CrossRef]
2. Liu, C.Y.; Cao, J.; Wang, J. A Reliable and Efficient Distributed Service Composition Approach in Pervasive Environments. *IEEE Trans. Mob. Comput.* **2017**, *16*, 1231–1245. [CrossRef]
3. Wang, P.W.; Liu, T.; Zhan, Y.; Du, X.Y. A Bayesian Nash Equilibrium of QoS-aware Web Service Composition. In Proceedings of the 24th IEEE International Conference on Web Services, Honolulu, HI, USA, 25–30 June 2017; pp. 676–683.
4. Min, X.Y.; Xu, X.F.; Liu, Z.Z.; Chu, D.H.; Wang, J.Z. An approach to resource and QoS-aware service optimal composition in the big service and internet of things. *IEEE Access* **2018**, *6*, 39895–39906. [CrossRef]
5. Bellavista, P.; Corradai, A.; Foschini, L.; Monti, S. Improved Adaptive and Survivability via Dynamic Service Composition of Ubiquitous Composition Middleware. *IEEE Access* **2018**, *6*, 33604–33620. [CrossRef]
6. Ardagna, D.; Pernici, B. Adaptive service composition in flexible processes. *IEEE Trans. Softw. Eng.* **2007**, *33*, 369–384. [CrossRef]
7. Siriweera, T.H.A.S.; Paik, I. QoS-Aware Rule-Based Traffic-Efficient Multiobjective Service Selection in Big Data Space. *IEEE Access* **2018**, *6*, 48797–448814. [CrossRef]
8. Zeng, L.; Benatallah, B.; Kalagnanam, J.; Chang, H. QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.* **2004**, *19*, 311–327. [CrossRef]
9. Ding, Z.J.; Liu, J.J.; Sun, Y.Q.; Jiang, C.J.; Zhou, M.C. A Transaction and QoS-Aware Service Selection Approach Based on Genetic Algorithm. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *45*, 1035–1046. [CrossRef]
10. Huo, L.; Wang, Z.L. Service Composition Instantiation Based on Cross-Modified Artificial Bee Colony Algorithm. *China Commun.* **2016**, *13*, 233–244. [CrossRef]
11. Yi, Q.; Wei, Z.; Chen, H.L. Improved adaptive immune genetic algorithm for optimal QoS-aware service composition selection in cloud manufacturing. *Int. J. Adv. Manuf. Technol.* **2018**, *96*, 4455–4465.
12. Moustafa, A.; Zhang, M.J.; Bai, Q. Trustworthy Stigmergic Service Composition and Adaptation in Decentralized Environments. *IEEE Trans. Serv. Comput.* **2016**, *9*, 317–329. [CrossRef]
13. Yong, Z.; Wei, L.; Luo, J.Z.; Zheng, X. A Novel Two-Phase Approach for QoS-Aware Service Composition Based on History Records. In Proceedings of the Fifth IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Taipei, Taiwan, 17–19 December 2012; pp. 1–8.
14. Yu, T.; Zhang, Y.; Lin, K.J. Efficient algorithms for Web services selection with end-to-end QoS constraints. *ACM Trans. Web* **2007**, *1*, 6–12. [CrossRef]
15. Lu, W.; Wang, W.D.; Bao, E. FAQS: Fast Web Service Composition Algorithm Based on QoS-Aware Sampling. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2016**, *E99A*, 826–834. [CrossRef]
16. Wang, L.J.; Shen, J. A Systematic Review of Bio-Inspired Service Concretization. *IEEE Trans. Serv. Comput.* **2017**, *10*, 493–505. [CrossRef]
17. Tang, M.L.; Ai, L.F. A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. In Proceedings of the World Congress on Computational Intelligence, Barcelona, Spain, 18–23 July 2010; pp. 1–8.
18. Lecue, F.; Mehandjiev, N. Seeking Quality of Web Service Composition in a Semantic Dimension. *IEEE Trans. Knowl. Data Eng.* **2011**, *23*, 921–959. [CrossRef]
19. Zheng, X.; Luo, J.; Song, A. Ant Colony System Based Algorithm for QoS-Aware Web Service Selection. In Proceedings of the 4th International Conference on Grid Service Engineering and Management (GSEM), Leipzig, Germany, 25–26 September 2007; pp. 39–50.

20. Xia, Y.; Chen, J.; Meng, X. On the Dynamic Ant Colony Algorithm Optimization Based on Multi-Pheromones. In Proceedings of the Seventh IEEE ACIS International Conference on Computer and Information Science (ICIS '08), Portland, OR, USA, 14–16 May 2008; pp. 619–635.
21. Wang, W.; Sun, Q.; Zhao, X.; Yang, F. An Improved Particle Swarm Optimization Algorithm for QoS-Aware Web Service Selection in Service Oriented Communication. *Int. J. Comput. Intell. Syst.* **2012**, *3*, 18–19. [CrossRef]
22. Cho, J.H.; Choi, J.H.; Ko, H.G.; Ko, I.Y. An Adaptive Quality Level Selection Method for Efficient QoS-aware Service Composition. In Proceedings of the IEEE 36th International Conference on Computer Software and Applications Workshops, Izmir, Turkey, 16–20 July 2012; pp. 20–25.
23. Jiang, H.H.; Yang, X.H.; Yin, K.T.; Jerry, A. Multi-path QoS Aware Service Composition using Variable Length Chromosome Genetic Algorithm. *Comput. Integr. Manuf. Syst.* **2011**, *10*, 113–119. [CrossRef]
24. Wang, L.; He, Y. A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO. In Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011), Melbourne, Australia, 19–20 November 2011; Volume 111, pp. 79–86.
25. 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. *Appl. Soft Comput.* **2012**, *12*, 2208–2216. [CrossRef]
26. Sun, S.A.; Zhao, J. A decomposition-based approach for service composition with global QoS guarantees. *Inf. Sci.* **2012**, *199*, 138–153. [CrossRef]
27. Wang, S.G.; Sun, Q.B.; Yang, F.C. Web service dynamic selection by the decomposition of global QoS constraints. *J. Softw.* **2011**, *22*, 1426–1439. [CrossRef]
28. Zhang, W.X.; Lang, Y.S. *The Mathematical Basis of Genetic Algorithm (Version 2)*; Jiaotong University Press: Xi'an, China, 2004; pp. 66–72.
29. Bakhshi, M.; Mardukhi, F. A Fuzzy-Based Approach for Selecting the Optimal Composition of Services According to User Preferences. In Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 26–28 February 2010; pp. 129–135.
30. Sharifara, P.; Yari, A.; Mansour, R.K. An Evolutionary Algorithmic based Web Service Composition with Quality of Service. In Proceedings of the 7th International Symposium on Telecommunications, Tehran, Iran, 9–11 September 2014; pp. 56–65.
31. Kashyap, N.; Tyagi, K. Dynamic Composition of Web Services Based on Qos Parameters Using Fuzzy Logic. In Proceedings of the International Conference on Advances in Computer Engineering and Applications (ICACEA), Ghaziabad, India, 19–20 March 2015; pp. 308–782.
32. Wu, Z.P.; Yuan, M. User-Preference-Based Service Selection Using Fuzzy Logic. In Proceedings of the 2010 International Conference on Network and Service Management, Niagara Falls, ON, Canada, 25–29 October 2010; pp. 321–345.
33. Silvana, D.G.A.; Karim, D. Fuzzy Logic Based QoS Optimization Mechanism for Service Composition. In Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oeiented System Engineering, Redwood City, CA, USA, 25–28 March 2013; pp. 182–191.
34. Chuang, S.N.; Chan, A.T.S. Dynamic QoS adaptation for mobile middleware. *IEEE Trans. Softw. Eng.* **2008**, *34*, 738–752.
35. Wang, H.B.; Zou, B.; Guo, G.B.; Yang, D.R.; Zhang, J. Integrating Trust with User Preference for Effective Web Service Composition. *IEEE Trans. Serv. Comput.* **2017**, *10*, 574–588. [CrossRef]
36. Branson, J.S.; Lilly, J.H. Incorporation, Characterization, and Conversion of Negative Rules into Fuzzy Inference Systems. *IEEE Trans. Fuzzy Syst.* **2001**, *9*, 253–268. [CrossRef]
37. Niu, S.; Zou, G.B.; Gan, Y.L.; Xiang, Y.; Zhang, B.F. Towards Uncertain QoS-aware Service Composition via Multi-objective Optimization. In Proceedings of the IEEE 24th International Conference on Web Services, Honolulu, HI, USA, 25–30 June 2017; pp. 894–897.
38. Roy, R.; Dehuri, S.; Cho, S. A Novel Paricle Swarm Optimization Algorithm for Multi-Objective Combinational Optimization Problem. *Int. J. Appl. Metaheurisitic Comput.* **2011**, *2*, 41–57. [CrossRef]

39. Al-Masri, E.; Mahmoud, Q.H. The QWS Dataset. Available online: http://www.uoguelph.ca/~qmahmoud/qws/index.html (accessed on 18 March 2019).
40. Ren, L.F.; Wang, W.J.; Xu, H. A Reinforcement Learning Method for Constraint-Satisfied Services Composition. *IEEE Trans. Serv. Comput.* **2018**, *7*, 32–39. [CrossRef]