A Flexible QoS-aware Web Service Composition Method by Multi-objective Optimization in Cloud Manufacturing

Abstract: Cloud manufacturing, combining Web services via internet to a cooperative manufacturing system, has been an increasing popularity for global manufacturing. It will unlock the tremendous value in the massive amount of data being generated by the manufactories. The problem of QoS-aware Web service composition (QWSC), i.e., selecting appropriate service for each component of a service composition from a pool of functionally identical service to satisfy the users' end-to-end QoS constraints, is a core of the cloud manufacturing. A novel QWSC method by multi-objective optimization is proposed to help users to make a flexible decision. First of all, the problem of QWSC is formulated to a multi-objective optimization model where either QoS performance or QoS risk (variance comparing to the user`s QoS requirement) is the individual optimization objective. And then, an efficient ε-dominance multi-objective evolutionary algorithm (EDMOEA) is developed to solve the presented model. Finally, experimental results verify the effectiveness and efficiency of the proposed method for the large-scale QWSC problem.

Keywords: Cloud manufacturing, Web service composition, Quality of service, Multi-objective optimization; Evolutionary algorithm

1. Introduction

Nowadays, cloud computing or service-oriented computing (SOC), where Web service is the essential element to support rapid, low-cost, and easy composition of distributed applications in heterogeneous environments; have become increasingly popular, and triggers a considerable amount of research efforts in both academia and industry (Papazoglou et al., 2007). SOC provides a new opportunity to the modern manufacturing industry to do e-manufacturing and business collaborations for global manufacturing, which aims to align business processes across the organizations, (Wu, et al., 2013). Based on the trends in devices, growth of data, communication bandwidth, and growth of peer to peer sharing, it is argued that utilizing Web services offered by third-parties with a greatly reduced cost is emerging for manufacturing industries that will unlock the tremendous value in the massive amount of data being generated by the industrial manufactories (Tao and Zhang, 2012).

In most cases, individual Web services are combined to a composite service to solve a complex goal successfully. Fig. 1 shows an example of composite service used in warehouse fire alarm, where individual devices providing simple function can be combined to establish a more powerful alarm service. It involves different tasks including temperature sensors, smoke detectors, infrared sensors, liquid immersion sensors, alarm bells, camera, scupper valves and water sprinklers. Each task can be bound to various Web services providing the same functionality but with different quality properties.

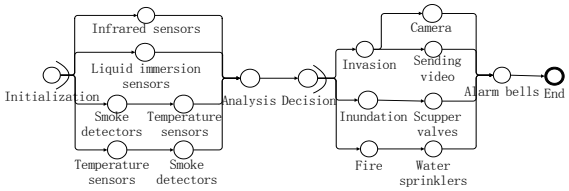


Fig.1 A composite Service of Warehouse Fire Alarm

As many Web services deliver the equivalent functionality, Quality of Service (QoS), such as the cost, response time, reliability, and availability are crucial concern to help users to select appropriate services satisfying their preferences, for instance, cost should be less than $20, and response time should be less than two hours. The concern with QoS introduces a challenging problem of QoS-aware Web service composition (QWSC), i.e., select the best composite service with optimal QoS performance while satisfying given end-to-end QoS constraints (Qi et al., 2012). Unfortunately, due to the increasing number of component services (tasks) in a composite service and the growing spread of candidates for each task, the size of possible composition plans built by selecting different Web service instance for each component service should be in general exponentially enlarging. In other words, the QWSC is a NP-Complete problem in a strong sense which cannot be solved within a reasonable time (Haddad et al., 2010).

Motivated by the large-scale QWSC problem, this study propose a flexible and efficient Web service selection method by multi-objective optimization to provide users enhanced information to choose appropriate services. A QoS model for describing vague concept of QoS with specific features is constructed. Furthermore, traditional optimization methods for QWSC tend to find the best composite service fulfilling user's overall QoS constraint. They may fail to provide a possible solution if unsuitable QoS constraints are submitted. Unfortunately, it is impossible that users are always well-informed to submit reasonable QoS requirements. From this perspective, then the problem of QWSC is formulated into a multi-objective optimization model where either QoS performance or variance from the user's QoS constraints (i.e. QoS risk) is an independent objective. Moreover, a ε-dominance multi-objective evolutionary algorithm (EDMOEA) is developed to solve the presented model. Finally, experimental studies demonstrate the effectiveness and the efficiency of the proposed method.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 presents a QoS model of Web service that defines the QoS criteria including cost, execution time, latency time, reliability and availability; and the QoS aggregation for a composite service. The multi-objective optimization model for QWSC and the solving algorithm are specified in section 4 and section 5 respectively, Section 6 presents the experimental studies, while section 7 offers the concluding remarks and outlines future study.

2. Related Work

As stated, service composition is recognized as a procedure where individual Web services are choreographed into a complex composite service. The main approach for Web service composition is the model-based approach, that uses a composition schema to express the abstract functionalities of component services and the integration logics of component services (Strunk, 2010; Kwon and Lee, 012; Tao et al. 2012). Composition engine schedules and executes concrete component services according to the composition schema. The common process of model-based service composition is to perform firstly a matchmaking process for each component service and return candidate services implementing the required functionality, and then select appreciate Web services to assemble the best composite service fulfilling the QoS constraints. The later procedure is also called the QWSC.

Recently, several notable QoS measure and computing approaches for QWSC have been developed. For instance, W3C (http://www.w3.org/Architecture/qos.html) working group defined various QoS attributes for Web services. Zeng et al. (2004) proposed five generic QoS criteria including execution price, execution duration, reputation, successful execution rate, and availability; and a five-dimensional QoS evaluation model is presented. It was commonly used in later researches. As users submit an end-to-end QoS goal rather than specify QoS constrains for each individual service, the overall QoS expression of a composite service were determined ultimately by the QoS values of its individual services on the basis of workflows between them. Aalst et al. (2003) state there are more than twenty different patterns through which individual services can be integrated to form a composite service. However, four well-known workflow patterns among them are essentially considered including sequence, parallel, switch and loop, and several aggregation formulas are then presented to compute the QoS of workflow patterns (Kwon and Lee, 2012; Tao et al., 2012; Sun and zhao, 2012; Ren et al., 2013).

In the context of QWSC, Zeng et al. (2004) first transfer the problem of QWSC into an optimization problem. From then on, the optimization of QWSC became an active topic in the QWSC and attracted much attention over recent years. Zeng et al. (2004) and Alrifai et al. (2012) employed a mixed integer programming to seek the optimal candidates of service composition. Yu et al. (2007) proposed a scheme to optimize the end-to-end QoS for various flow structures. They used a utility function derived from the QoS performance. Then, the QWSC problem was formulated as a 0-1 multi-choice multidimensional knapsack problem (MMKP) and a multi-constraint optimal path approach was used to handle the QoS constraints. Furthermore, Lin et al. (2011) presented a relaxable QWSC approach to handle the case when none of solutions satisfied the QoS constraints. They also formulated the optimization of QWSC as a MMKP, while specified each QoS constraint as either a relaxable constraint or a non-relaxable constraint. If there is no feasible solution to fulfill the QoS constraints, a solution with smaller amount of constraints violation would be presented to fulfill the

non-relaxable QoS constraints by relaxing the relaxable QoS constraints. Liu et al. (2012) modeled the QWSC as a flexible constraint satisfaction problem (CSP), and used a branch and bound-based heuristic algorithm to reduce the computational complexity. Huang et al. (2009) used multiple criteria decision making (MCDM) model to represent the QWSC, as well as Tao and Zhang (2012). Wang et al. (2010) proposed a fuzzy group consensus QWSC model to handle vague preferences or linguistic opinions for users' QoS attributes during the selection of web services. They mapped the QWSC to a MCDM problem and applied linear programming technique to solve this problem.

As the QWSC is known to be NP-Complete in the strong sense, an optimal solution may not be expected to be found in a reasonable amount of time. Hence, many efforts on various intelligent algorithms and heuristic algorithms in QWSC optimization were concentrated to find an approximate optimal solution, such as Evolutionary Algorithm (EA), Particle Swarm Optimization (PSO), Immune Algorithm, Ant Colony Optimization (ACO), and so on. Currently GA and PSO are the most promising concerned algorithms in the QWSC. Mardukhi et al. (2012) used Genetic Algorithm (GA) is used to dynamic QWSC. Tao et al. (2008) mapped the optimal resource service selecting in manufacturing grid system into the multi-objective problem, and Zhao et al. (2012) transformed QWSC into a weighted summation optimization model with QoS constraints, respectively. Both of them used PSO to solve the presented models. Furthermore, Fan et al. (2011) utilized a cooperative evolution algorithm consists of stochastic PSO and simulated annealing (SA) to solve the QWSC.

However, most of the current QWSC optimization approaches may lack of flexibility on the selection of composite services. Specifically, many approaches asked users to specify their expected QoS preferences and even transform them into numeric measurements. Generally, it is impossible for users always to state appropriate QoS constraints, with which the satisfied solution can be available, as they may be unacquainted enough with the service market. Furthermore, most approaches work in a “black box” manner where users submit their QoS constraints and only one optimized solution is automatically selected by system. From this perspective, users cannot flexibly select their desired services depending on the real QoS performance. In practice, users prefer a set of acceptable non-inferior solutions, which support them to make flexible decisions to satisfy their various QoS requirements, to only one solution for the specific QoS constraints.

Aiming to this issue, a multi-objective QWSC optimization method where Pareto optimal solutions (i.e., Pareto frontier) can be obtained is proposed. In contrast to the only one automatically system-selected service, Pareto frontier can present enhanced information for flexible service selection. As an efficient multi-objective solving algorithm is necessary to solve the large-scale QWSC, we develop an ε-dominance multi-objective evolutionary algorithm (EDMOEA), where a new preservation technique of population diversity based on the ε-dominance relation is adopted to improve convergence and diversity of the Pareto frontier.

3 QoS Computing Model

In this section, we give the details on how to evaluate QoS of Web service and aggregate QoS of composite service based on workflow patterns.

3.1 QoS Properties of Web Service

Five QoS attributes are chosen in this study, which are execution time, latency time, cost, reliability and availability, as shown in Table 1. The proposed QoS metrics is capable to present an overview of QoS properties that users mostly concern.

Table 1. The QoS model for Web service

QoS Attributes	Descriptions
Cost (*C*)	Execution fee per request
Execution time (*ET*)	Time to perform the service functionality
Latency time (*LT*)	Request waiting time before execution
Reliability (*R*)	1- Failure rate
Availability (*A*)	uptime /(uptime + downtime)

Cost is the fee that the users have to pay for the invocation of Web service, which is the most import concern for users to choose the service or not. It is determined or changed by service providers according to the capability of the service, the marketing property or providers' financial policy.

Response time, i.e., a period between sending a request by user and receiving response by provider, is another essential concern for Web service. We suggest the response time as a comprising of execution time and latency time. The execution time measures the expected delay between the moment when a service starts to execute and the moment when the service is finished. The latency time is defined as the time taken prior to service execution, e.g. request queuing time. According to these definitions, service providers will be able to provide more accurate descriptions of their service performance regardless of flawing the measurement with uncertain network fluctuations.

Reliability refers to the ability of a service provider to successfully perform the required service functions under stated conditions for a specified period of time. It is usually evaluated through the service failure rate which is measured as the ratio of execution time and mean time between failures.

Availability is the ability of processing the received requests, which means the degree that a service is operational and accessible when it is required for use. It can be defined as the ratio of the total time a service is capable of being used during a given interval to the length of the interval.

3.2 QoS Aggregation for Composite Service

In the QWSC, a composition schema is commonly used to specify the abstract functionalities of each component service and the integration logics of these component services. A candidate composite service corresponding to the composition schema, denoted composition plan (*CP*), consists of a set of

logically connected concrete Web services.

An aggregative value of every QoS attribute for a *CP* can be calculated on the basis of the specified composition schema (Liu et al., 2012; Wu and Zhu, 2013; Alrifai et al., 2012). Table 2 shows the QoS aggregation functions for various workflow patterns. We assume each case statement in the switch pattern is specified with execution probability p_i, and the expected iteration number k of the loop pattern is also specified. Note that, in the sequence, parallel and switch columns, k indicates the number of services in the same pattern.

Table 2. The QoS aggregation functions for various workflow patterns

QoS Attributes	Sequence,	Parallel	Switch	Loop
Cost (*C*)	$\sum_{i=1}^{k} C(s_i)$	$\sum_{i=1}^{k} C(s_i)$	$\sum_{i=1}^{k} p_i C(s_i)$	$k*C(s_i)$
Execution time (*ET*)	$\sum_{i=1}^{k} RT(s_i)$	Max($RT(s_i)$)	$\sum_{i=1}^{k} p_i RT(s_i)$	$k*RT(s_i)$
Latency time (*LT*)	$\sum_{i=1}^{k} LT(s_i)$	Max($LT(s_i)$)	$\sum_{i=1}^{k} p_i LT(s_i)$	$k*LT(s_i)$
Reliability (*R*)	$\prod_{i=1}^{k} R(s_i)$	Min($R(s_i)$)	$\sum_{i=1}^{k} p_i R(s_i)$	$(R(s_i))^k$
Availability (*A*)	$\prod_{i=1}^{k} A(s_i)$	Min($A(s_i)$)	$\sum_{i=1}^{k} p_i A(s_i)$	$(A(s_i))^k$

For every candidate *CP* where logically connected component services are independently assigned a concrete Web service, the aggregative value of each QoS attribute can be calculated by recursively applying formulations shown in Table 2.

4 Multi-objective Optimization Model for Service Selection

The goal of a multi-objective optimization problem (MOP) is to find a set of compromise solutions regarding different objectives rather than the best one as in single-objective optimization problems. A multi-objective optimization model can be defined as following (Chinchuluun and Pardalos, 2007):

$$\begin{aligned} &\text{Min} \quad \mathrm{f}(x) = (f_1(x), f_1(x), \cdots, f_k(x))^T \\ &s.t. \quad \mathrm{x} \in \Omega \end{aligned} \tag{1}$$

Therein, $\Omega \in \mathrm{R}^n$ is the feasible solutions set that is nonempty, $f(x)$: $\Omega \rightarrow \mathrm{R}^k$ is a vector-valued function, k is the number of objectives.

As stated, the problem of finding optimal *CP*s satisfying the user's QoS constrains is a kind of optimization problem that can be solved by mathematical programming techniques. Generally, users try to get the best *CP* to satisfying their expected QoS. If such a *CP* is unavailable, they may favor some alternatives with the maximum QoS performance and the minimum variance from their expected QoS. From such a perspective, we formulate the QWSC problem to a multi-objective optimization

model where users gain flexibility of decision with a tradeoff among QoS performance and QoS variance comparing to the given constraints. The goal of the multi-objective QWSC optimization is to select *CP* which can minimize the QoS variance from the given constraints (denoted as QoS risk), and maximize the overall QoS performance, i.e. either the QoS risk or the QoS performance is an individual objective.

As multi-objective optimization minimizes objectives, the goal of QoS performance maximization is transformed to a reverse objective, i.e., QoS disadvantage degree from an ideal positive solution. The multi-objective optimization model of QWSC is interrupted in details as follow. We use Q= (Q_1, Q_2, … , Q_5) to represent the QoS properties, where Q_1, Q_2, Q_3, Q_4, Q_5 represents the attribute of cost, execution time, latency time, reliability and availability, respectively. q = (q_1, q_2, … , q_5) refers to the aggregative QoS values of a *CP*, where q_i denotes the value of QoS attribute Q_i. Suppose that q^* = (q_1^*, q_2^*, … , q_5^*) is the end-to-end QoS constrains that users can endure. Note that, the average of each QoS attribute will be regarded as the QoS constraints if users don't present QoS constraints.

As each numeric QoS property may be evaluated in various measurements, as well as the QoS constraints, a normalization is used to scale various ranges of QoS attributes to a specified scale [new_q_{min}, new_q_{max}] (Mardukhi et al., 2013; Zhao et al., 2012), for instance [0, 10] or [0, 1], as shown in Eq. (2). Therein q_{max} and q_{min} are the maximal and minimal values among all the *CP*s, respectively, q and q' refer to the original value and its corresponding normalized value. Without loss of generality, we refer q to the normalized QoS values in the following part of this study for convenience.

$$q' = \frac{q - q_{\min}}{q_{\max} - q_{\min}} (\text{new\_q}_{\max} - \text{new\_q}_{\min}) + \text{new\_q}_{\min} \tag{2}$$

We define the QoS risk as the variance between the expected QoS requirements and the actual QoS performance of the *CP*, referring the meaning of risk in finance field (Markowitz, 1952). The QoS properties can be categorized into positive attributes (such as reliability and availability) and negative ones (such as execution time, latency time and cost), denoted as Q^+ and Q^-, respectively. The higher value of positive properties indicates the better QoS, and the higher value of negative properties represents the worse QoS and vice versa. Generally, if and only if the value of a positive property is less than the user's requirement, or the value of a negative property is larger than the user's requirement, user would concern the QoS variance, i.e. the risk emerges. Consequently, we use Eq. (3) to measure the risk of *CP*.

$$R(CP) = \sqrt{\sum_{Q_i \in Q^+} (\max(0, q_i^* - q_i))^2 + \sum_{Q_i \in Q^-} (\max(0, q_i - q_i^*))^2} \tag{3}$$

As the multi-objective optimization refers that each objective function should be minimized, the goal of QoS performance maximization is transformed to a reverse objective, i.e., QoS disadvantage degree from an ideal positive solution. Suppose a positive ideal solution, CP^+, represents the best *CP*

with the highest QoS performance, i.e. the cost, execution time and latency time are minimum, and the reliability and availability are maximum. Note that the CP^+ can be illustrated by assigning the best instance for each component; or represented as a solution whose cost, execute time and latency time are 0, but reliability and availability are 1. The QoS performance of the CP^+ is denoted as $q^+ = (q_1^+, q_2^+, \ldots, q_5^+)$. Then the QoS disadvantage degree D of CP is defined as the Euclidean distance between the QoS vector of CP and that of the CP^+, as is shown in Eq. (4). Obviously, the less value of disadvantage degree indicates the higher QoS of CP.

$$D(CP) = \sqrt{\sum_{i=1}^{5} \left| q_i - q_i^+ \right|^2} \tag{4}$$

In summary, the multi-objective optimization model for QWSC can be represented as following.

$$\begin{aligned} & MinZ_1 = R(CP) = \sqrt{\sum_{Q_i \in Q^+} (\max(0, q_i^* - q_i))^2 + \sum_{Q_i \in Q^-} (\max(0, q_i - q_i^*))^2} \\ & MinZ_2 = D(CP) = \sqrt{\sum_{i=1}^{5} \left| q_i - q_i^+ \right|^2} \\ & s.t.\ \ q_i \geq 0, q_i^* \geq 0, q_i^+ \geq 0 \end{aligned} \tag{5}$$

Therein, $q = (q_1, q_2, \ldots, q_5)$ is the QoS values of CP, $q^* = (q_1^*, q_2^*, \ldots, q_5^*)$ is the user specified QoS constrains, and $q^+ = (q_1^+, q_2^+, \ldots, q_5^+)$ is the ideal QoS parameter. The first objective is minimal the QoS risk, and the second one is maximal the QoS performance.

5 Multi-objective Optimization Solving Algorithm Design

For MOP (shown in Eq. (1)), solution x_1 dominates solution x_2 if and only if $f(x_1)$ is partially less than $f(x_2)$, i.e. $\forall i \in \{1,2, \ldots, k\}, f_i(x_1) \leq f_i(x_2) \wedge \exists i \in \{1,2, \ldots, k\}, f_i(x_1) - \varepsilon_i < f_i(x_2)$, denoted as $x_1 \preceq x_2$. The result of MOP is a set of non-dominated solutions known as Pareto optimal solutions or Pareto frontier. Convergence and distribution (or diversity) are the main concerns in the MOEAs. Though some multi-objective evolutionary algorithms (MOEAs) have been proposed to solve the general MOP (Deb, 2001, 2002; Li et al., 2011), it is essential to develop an efficient algorithm to solve the large-scale QWSC MOP.

To achieve better convergence and diversity of the Pareto frontier, we develop a ε-dominance multi-objective evolutionary algorithm (EDMOEA) where a new preservation technique of population diversity based on the ε-dominance relation is adopted to converge to the Pareto optimal set much faster. The EDMOEA adopts pair-comparison selection instead of the fitness assignment or Pareto ranking (Deb, 2012) and designs a diversity preservation technique of population based on the ε-dominance relation (Li et al., 2011). The ε-dominance relation is a kind of the relaxed form of Pareto dominance (Laumanns et al., 2002), where ε represents the allowable tolerance in each objective below which two solutions are identical. The corresponding explanation is that solution x_1 ε-dominates solution x_2 if and only if $f_i(x_1) - \varepsilon_i \leq f_i(x_2) \wedge \exists\ i \in \{1,2, \ldots, k\}, f_i(x_1) - \varepsilon_i < f_i(x_2)$ ($\varepsilon_i \in R^k$), denoted as $x_1 \preceq_\varepsilon x_2$.

5.1 EDMOEA Designs

Details of the EDMOEA is presented in this section.

(1) Encoding: The real-coded chromosomes are used in EDMOEA. We number and quote all candidate Web service instances of each component as integers respectively. Each chromosome (individual) *m* genes represents a candidate *CP*. Every gene g_i showed the identification of the chose service instances for the corresponding task, as shown following:

g_1	g_2	…	g_m

(2) Elitist participation strategy: EDMOEA evolves two populations in parallel in a single run of the algorithm. One is main population *P*, and another is an archive population *A* containing non-dominated individuals (i.e. elite individuals) found previously. The strongest elite participation is employed in EDMOEA, where one parent is chosen from the main population while the other is chosen from the current archive population for producing two offspring.

To improve the convergence and the diversity, the ε-dominance relation is used to find non-dominated solutions and a novel update strategy of archive population is designed. Specifically, Pareto dominance relation and ε-dominance relation are checked respectively between the new offspring and all the members of the archive population. Thus it could reduce the cost by checking the dominance relation of the offspring ahead instead of adding both of them into the archive population directly.

(3) Genetic operators: We use two-point crossover method to exchange information between two parent individuals to produce two offspring, i.e., two crossover points are generated randomly and then genes of the two individuals between the two points are exchanged to produce two offspring. The crossover probability is 0.8 and the rate of mutation is 0.1.

The EDMOEA outlined in Table 3 consists of three phases: (1) initialization phase (Tab. 3, Lines 2-4), (2) selection-production phase (Tab. 3, Lines 6-11), and (3) population-updating phase, i.e., main population update (Tab. 3, Lines 12) and archive truncating (Tab. 3, Lines 13-21). EDMOEA use Pareto selection method in the first parent selection from population *P*, i.e. Select two individuals randomly from *P*, compare the two individuals according to dominance relationship and choose the non-dominated one. If they are non-dominated then choose one of them randomly. After that the second parent individual is selected randomly from the archive set *A*. In order to avoid to remove non-dominated individual, the parent replacement method is used to update the main population *P* with two offspring: Compare the two offspring according to dominance relationship and choose the non-dominated one. If they are non-dominated, choose one of them randomly. Replace the parent individual in *P* with this offspring.

The archive population is updated by ε-ArchiveUpdate algorithm as shown in Table 4. In the ε-ArchiveUpdate algorithm, an individual *x* is checked through the following three steps. First, *x* is compared with the members of the archive population. The solutions included in the set dominated(*x*) are removed. Second, the previous inferior ones would be removed if they are ε-dominated by *x*;

otherwise they are retained in *A*. Lastly, if *x* is not dominated by, nor does it dominate, individuals in the archive population, the ε-Pareto dominance relation is used to determine whether to take in *x* or not. This procedure has the advantages of both diversity of population and the higher efficiency of computation over other archive truncating approaches.

Table 3. The design of EDMOEA

```
Begin
Generate N individuals randomly, where N is the size of population P.
Evaluate the objective values for each individual in population P.
Copy the non-dominated individuals to the archive population A.
Repeat
    Choose an individual in P as the first parent using Pareto selection method: Select two individuals
      randomly from P. Compare the two individuals according to dominance relationship and choose
      the non-dominated one. If they are non-dominated then choose one of them randomly.
    Choose randomly one individual from A as the second parent into the mating pool.
    If these two parent individuals are identical
        Select another individual from A randomly to replace either of them
    End If
    Apply crossover and mutation to the two parents to producing two offspring: s1, s2
    Update the population P using Parent replacement method with two offspring: Compare the two
      offspring according to dominance relationship and choose the non-dominated one. If they are
      non-dominated, choose one of them randomly. Replace the parent individual in P with this
      offspring.
    If s1 ε-dominates s2
        ε-ArchiveUpdate(A, s1)
    Else If s2 ε-dominates s1
        ε-ArchiveUpdate(A, s2)
        Else
            ε-ArchiveUpdate(A, s1)
            ε-ArchiveUpdate(A, s2)
        End If
    End If
Until stopping condition is satisfied.
End
```

Table 4. The ε-ArchiveUpdate procedure

```
Procedure ε-ArchiveUpdate(A, x)
Flag←0
If ∃x′ ∈ A such that x′ ⪯ x, then
    Flag←1
Else
    D={x′ ∈ A | x ⪯ x′}, T←T\D
    If D≠Φ then
        T←T∪{x}, A←T, Flag←1
    End If
End If
If Flag=1, then
    Return
End If
If ∃x″ ∈ T such that x″ ≺ε x, then
    Return
Else
    T←T∪{x}, A←T
End If
END Procedure
```

5.2 Time complexity of EDMOEA

In this section, computational complexity of EDMOEA is analyzed. In details, we compare the EDMOEA with NSGA-II (Deb et al., 2002), which is one of the most popular multi-objective optimization algorithms. In The EDMOEA used the ε-dominance relation, but the NSGA-II employed a strict dominance relation. However, a procedure of fitness assignment is just with in NSGA-II, but without in EDMOEA.

NSGA-II executes an operation of fitness or ranking assignment, it requires comparisons of $O(2(N+\lambda)^2)$, where λ is the offspring population size. In the population-updating phase, the time complexity of NSGA-II is $O((2+1)(N+\lambda)\log(N+\lambda))$. Thus, the time complexity of NSGA-II is $O(2(N+\lambda)^2)+O((2+1)(N+\lambda)\log(N+\lambda))$.

Since the fitness assignment is not required in the EDMOEA, the time mainly cost in the selection-production phase and the population-updating phase. The selection-production procedure requires $O(m)$ operations on genes, where m is the number of tasks. In the population-updating phase, time involves the dominance comparison of updating an offspring. It depends on the number of objectives, i.e., 2 in this study, size of main population N, and size of archive population N_E. In the worst case of generating an offspring, the overall time for updating an offspring per iteration is $O(2(N + N_E))$. Although N_E is constantly changing with the evolutionary process, N_E is usually less than N. Thus, the worst case of time complexity is $O(2N)$, when N_E equals to N. The time spent on generating and updating an offspring is $O(m + 2N)$, which is linear with the number of tasks, number of objectives, and size of the main population. The time complexity of EDMOEA is $O(mN + 2N^2)$, which is smaller than the $O(2(N+\lambda)^2)+O((2+1)(N+\lambda)\log(N+\lambda))$ in NSGA-II.

6 Experiment Studies

In order to verify the proposed method, we performed simulation experiments on an illustrated web service composition scenario. The algorithm was coded in Matlab7.1 and ran on an Intel Core 8×3.4 MHz PC with 4 GB RAM under Windows 7.

6.1 Experiment Setup and Service Instances

A composition schema with m=10 tasks (shown in Fig.2) was illustrated to perform the experiments where Sequence, Parallel, Switch and Loop were represented by symbols →, ⊕, ⊗ and *, respectively. Each task were associated to n=10 candidate service instances. For the schema in Fig.2, there were 10^9 ($(10^2+10)*10^7 \approx 10^9$) candidate *CP*s with different QoS performance, which meat that the manual web service assignment was almost impossible.

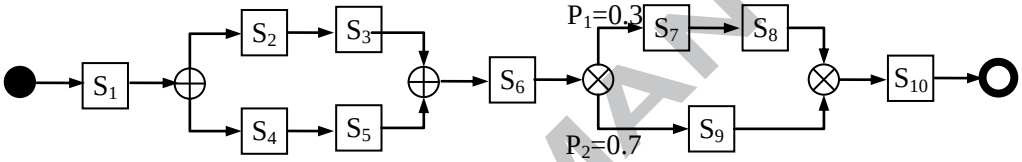


Fig.2 An illustrated composition schema

We used cost, execution time, latency time, reliability and availability as the QoS criteria; and generated a synthetic QoS dataset. Cost, execution time and latency time are uniformly assigned in [0, 1] respectively; and the reliability and availability are randomly generated with a uniform distribution between [0.7, 1]. Each aggregative QoS value of the candidate *CP* of the illustrated composition schema was normalized to the same interval [0, 10]. The normalized user`s global QoS constrains were set to q^* = (5, 5, 5, 3, 3), i.e., cost<=5, execution time<=5, latency time<=5, reliability>=3 and availability>=3 (note that reliability or availability of each component is not less than 0.8). The QoS of the positive ideal solution was q^+ = (0, 0, 0, 0, 10, 10), i.e., cost=0, execution time=0, latency time=0, reliability=10 and availability=10.

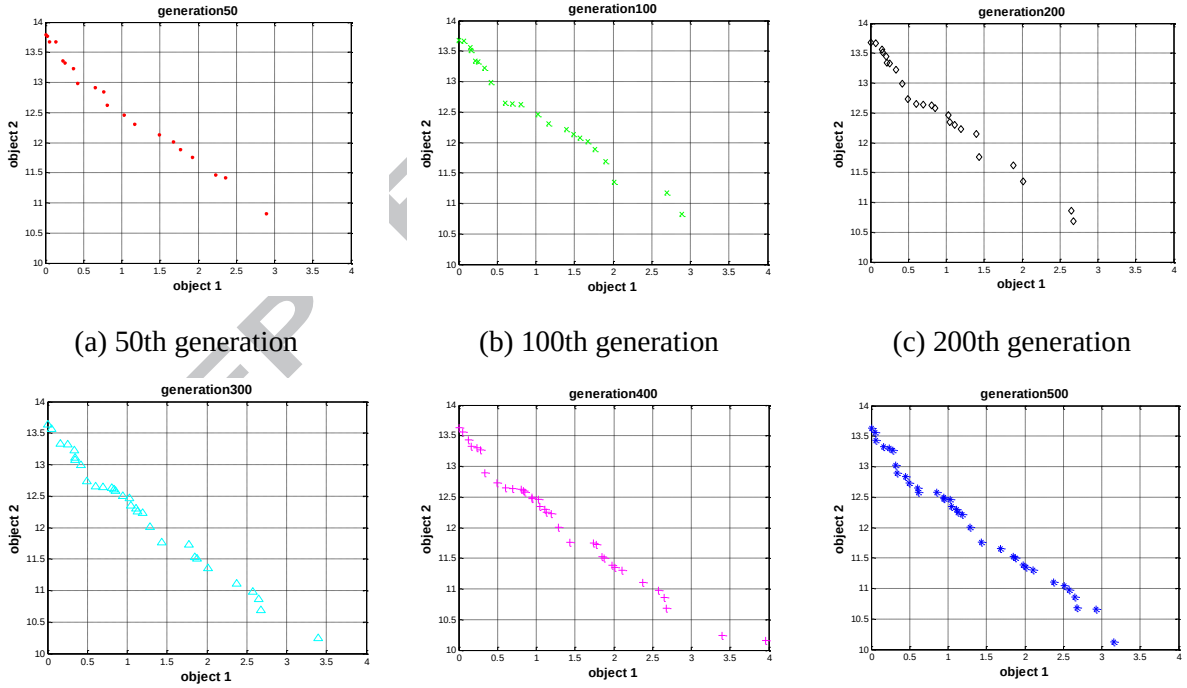
For EDMOEA, the crossover probability was 0.8 and the mutation probability was 0.1. The population size *np* was 100 and the max generation *maxgen* (the number of evolution cycles) was 500. The tolerance for objective was ε=0.01.

6.2 Performance evaluation of the EDMOEA

In this section, experiments were performed to investigate the performance the proposed EDMOEA algorithm. The non-dominated solutions for some specific generations, 50th, 100th, 200th, 300th, 400th and 500th; are shown in Fig.3. And the evolution of non-dominated solutions is summarily shown in Fig.4 simultaneously, where the solutions colored for various generations, respectively.

As indicated, the EDMOEA almost converge to the true Pareto optimal set within 300 generations. In other words, there are significant evolutionary processes during the first 300 generations. From then on, the rest processes in evolution increase inconspicuously, i.e. the Pareto frontiers in the later generations just changed a little. The outstanding efficiency benefits from the strongest elite participation. The strongest elite participation strategy selects elite solutions from the archive population (non-dominated solutions obtained previously). Furthermore, the offspring individuals are generated by two parents chosen from the archive set and an extreme point of the current Pareto front respectively. Both of them contribute to effectively driving the search toward the true Pareto frontier.

As shown in Fig.4, the solutions converge to the Pareto optimal frontier gradually during the evolution. We can see either the intermediate non-dominated solutions or the final Pareto optimal frontier, corresponding to the archive populations, distribute uniformly. At the same time, the Pareto frontier has a good diversity. These results demonstrate the efficiency and effectiveness of the proposed EDMOEA. The outstanding performance of the EDMOEA can be attributed to the archive population updating approach which can preserve boundary points and diversity in the archive population.



(a) 50th generation (b) 100th generation (c) 200th generation

(c) 300th generation (d) 400th generation (e) 500th generation

Fig.3. Non-dominated solutions of the specific generation

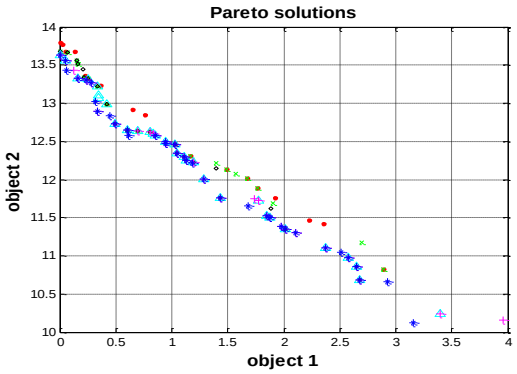
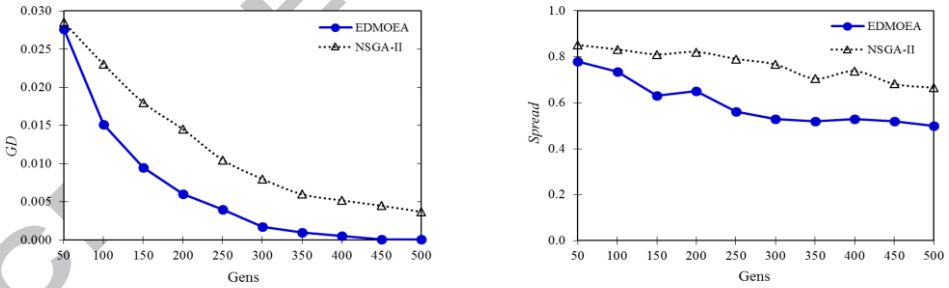


Fig.4. Evolution of the Non-dominated Solutions

6.3 Comparisons of EDMOEA and NSGA-II

In this section, we compared the EDMOEA to NSGA-II (Deb et al., 2002). The EDMOEA used the ε-dominance relation, but the NSGA-II employed a strict dominance relation. Generational distance (*GD*) (Deb, 2001) and *Spread* (Deb et al., 2002), the mostly used measures in multi-objective optimization algorithms, were used to evaluate the performance of EDMOEA in terms of convergence and distribution, respectively. The *GD* indicates close proximity to the true Pareto frontier *P** and the *Spread* measures distribution. The smaller a *GD/Spread* is, the better convergence/distribution a Pareto frontier has.

For a fair comparison, we took the union of the optimal solutions of EDMOEA and that of NSGA-II obtained by running 20 times with 500 generations as the approximate P*. Experiment was independently ran 20 times, and the average of measure was used for the evaluation. The comparisons were shown in Fig. 5.



(a) *GD*s with various *Gen*s (b) *Spread*s with various *Gen*s

Fig.5 Comparisons of EDMOEA with NSGA-II

As shown in Fig. 5(a), the *GD* of EDMOEA was absolutely better than the *GD* of NSGA-II in any specific generations. It indicated that the EDMOEA converged toward the *P** with a higher speed than the NSGA-II, i.e., the convergence of EDMOEA was better than that of NSGA-II. Fig. 5(b) showed the comparisons in *Spread* measures with specific generations. As indicated in Fig. 5(b), the distribution of EDMOEA outperformed the NSGA-II. Experimental results demonstrated that the proposed EDMOEA achieved better performance than the NSGA-II in terms of convergence and distribution.

6.4 Empirical analysis of the Pareto solutions

As stated in MOP of QWSC, the objective1 represents the gap among real QoS comparing to the

global QoS constraints, and the objective2 measures the QoS performance, which is denoted as the distance between QoS performance of *CP* and that of the positive ideal solution. The Pareto optimal solutions were compromised in the presence of the trade-offs between QoS variance and QoS performance. The less the objective1 or the objective 2 is, the better a *CP* is.

As mentioned, the Pareto solutions provide rich information contrast to the traditional optimization methods. Users can make a flexible decision on *CP*s selection with their own risk attitudes or QoS preference. To demonstrate it, we investigate the information of the Pareto frontier, and it was divided into three areas, as shown in Fig. 6.

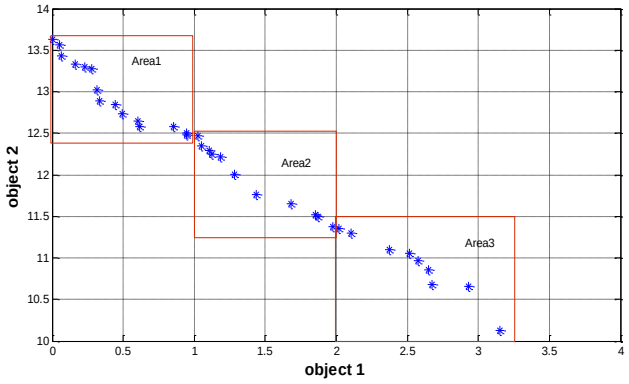


Fig.6 Distribution of the Pareto optimal solutions

Fig.6 indicated that the QoS risk was gradually increasing from areas 1 to 3, whereas the overall QoS performance was also upgrading. The plans of area 1 have lower QoS risk but worse QoS performance. Users with risk-averse attitude, i.e., prefer to little variance between real QoS and constraint; would likely choose solutions in area 1 although they cannot get the best QoS performance. However, if users prefer to the QoS performance, they may choose the plans in part 3 which have higher QoS performance but wider gap to the QoS constrains. The plans in area 2 expose the middle QoS variance and performance. The services may be favored by users with moderate QoS variance. The results demonstrated that the Pareto frontier provided rich information for users to make flexible decisions, especially in the case where none of solutions can satisfy the users' QoS constraints or users are not well-informed.

7 Conclusions and Future Work

In order to offer flexible alternative choice to users, we propose a novel QWSC method based on multi-objective optimization. The problem of QWSC is transformed to a MOP where both the QoS performance and the QoS risk are the optimization objective. Moreover, addressing the large-scale QWSC, an efficient EDMOEA is developed. The proposed method tries to find the Pareto optimal services, which supports users to select suitable services with the tradeoff of the QoS performance and the QoS risk. Experiments validate that the proposed method is capable to address the large-scale QWSC problem, and to offer users more enhanced information to make a flexible decision. We argue