Research Article

A Hybrid Multiobjective Discrete Particle Swarm Optimization Algorithm for a SLA-Aware Service Composition Problem

Hao Yin, Changsheng Zhang, Bin Zhang, Ying Guo, and Tingting Liu

College of Information and Engineering, Northeastern University, Shenyang, China

Correspondence should be addressed to Bin Zhang; zhangbin@ise.neu.edu.cn

Received 17 September 2013; Revised 1 November 2013; Accepted 16 November 2013; Published 23 January 2014

Academic Editor: Robertt A. Fontes Valente

Copyright © 2014 Hao Yin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For SLA-aware service composition problem (SSC), an optimization model for this algorithm is built, and a hybrid multiobjective discrete particle swarm optimization algorithm (HMDPSO) is also proposed in this paper. According to the characteristic of this problem, a particle updating strategy is designed by introducing crossover operator. In order to restrain particle swarm's premature convergence and increase its global search capacity, the swarm diversity indicator is introduced and a particle mutation strategy is proposed to increase the swarm diversity. To accelerate the process of obtaining the feasible particle position, a local search strategy based on constraint domination is proposed and incorporated into the proposed algorithm. At last, some parameters in the algorithm HMDPSO are analyzed and set with relative proper values, and then the algorithm HMDPSO and the algorithm HMDPSO+ incorporated by local search strategy are compared with the recently proposed related algorithms on different scale cases. The results show that algorithm HMDPSO+ can solve the SSC problem more effectively.

1. Introduction

Service-oriented architecture (SOA) is an emerging style of software architecture that reuses and combines loosely coupled services for building, maintaining, and integrating applications in order to improve productivity and cost effectiveness throughout the application life cycle [1]. In SOA, each application is often designed with a set of services and a workflow (or business process). Each service encapsulates the function of an application component. Each workflow defines how services interact with each other. When a service-oriented application operates, it is instantiated as a workflow instance that deploys each service in the application as one or more service instances. Each service instance follows a particular deployment plan; different service instances operate at different quality of service (QoS) levels. When an application is intended to serve different categories of users, it is instantiated with multiple workflow instances, each of which is responsible for offering a specific QoS level to a particular user category.

In SOA, a service-level agreement (SLA) is defined upon a workflow instance as its end-to-end QoS requirements

such as throughput, latency, and cost (e.g., resource utilization fees). In order to satisfy the given SLAs, application developers are required to optimize a composition of service instances, service composition, for each user category by considering which service instances to use for each service and how many service instances to use for each service. This decision-making problem, called the SLA-aware service composition (SSC) problem, is a combinatorial optimization problem that searches the optimal bindings between each service and its service instances.

Although a large number of research work have leveraged linear programming [2, 3] to solve the SSC problem, they suffer from high computational costs. To address the issue of high computational costs, several heuristics [4, 5] have been studied, but they just convert a multiobjective optimization problem into a single-objective optimization problem and do not take SLAs into account. Until 2012 Wada et al. [6] began to convert SSC problem into a multiobjective optimization problem with multiple SLAs and proposed algorithms MOGA and NSGA-II. But there are several problems in these two algorithms. Firstly, the service model for SSC problem associated with those two algorithms contains too

many objectives, which include all QoS attributes of service composition in all SLAs. In this way, the number of objectives will remarkably rise with the increase of the number of service composition's QoS attributes and the number of SLAs. Secondly, in the existing algorithms, when generating new populations, the chromosomes are used to make crossover and mutation by selection of the uniformly random manner, which cannot make some promising candidates to be selected more frequently. Furthermore, for the evaluation of the candidates in the environmental selection process, only the strength of a candidate being dominated is considered, but how many candidates it dominates is not considered. These make the existing algorithms for this problem converge slowly, easily fall into local optimum, and are hard to get satisfying solution sets when problem scale is relatively large. To tackle these problems, we redefine the service model for this SSC problem and propose algorithm HMDPSO to solve it. Our algorithm combined genetic operators and particle swarm optimization algorithm (PSO), which is a widely used swarm intelligence algorithm [7–9] and has the merits like easy realization, high precision, and fast convergence, and so on.

In this algorithm, a particle updating strategy is designed by introducing the crossover operator in genetic algorithm based on the exchange of candidate services; a particle mutation strategy is proposed to increase particle swarm's diversity by introducing new information and then increase the particle swarm's global searching capacity; in order to distinguish small differences among particle positions, a fine-grained domination value assignment method is used to calculate the fitness value of particles which will guide the particles' optimization process more accurately; in order to accelerate the process of obtaining feasible service composition and optimize it, an improved version of algorithm HMDPSO+ incorporated with a local search strategy improving the particle position from its candidates is designed. In the end, the evaluation of these two algorithms including their parameters turning and comparative studies based on 4 different cases is given.

2. Multiobjective Optimization Model for Service Composition

The SLA-aware service composition problem is a combinatorial optimization problem for searching the optima bindings between each service and its service instance, which is a NP-hard problem [6]. A workflow is modeled in Figure 1. It consists of a series of abstract services and each abstract service has several concrete services to realize its function. Concrete services can be bound to an abstract service in parallel and it need to make sure that at least one concrete service is bound to an abstract service.

When every abstract service has deployed with concrete services, it forms a service composition. Multiple service instances can be bound to an abstract service in parallel for improving the service's throughput and fault tolerance, which are called redundant parallel. In order to satisfy multilevel

TABLE 1: QoS aggregation function in abstract service.

QoS	Aggregation function
Throughput	$\sum_{k=1}^M x_{ijk} \times Q_1(s_{jk})$
Latency	$\frac{1}{\sum_{k=1}^M x_{ijk}} \sum_{k=1}^M x_{ijk} \times Q_2(s_{jk})$
Cost	$\sum_{k=1}^M x_{ijk} \times Q_3(s_{jk})$

requirements from SSC problem, a solution representing multilevel service composition has to be designed.

In order to judge whether a service composition satisfies a given SLA, it is required to examine its end-to-end QoS by aggregating QoS measures of individual service instances. Suppose a two-dimensional vector $S = (s_{11}, \dots, s_{NM})$ representing all concrete services, where N denotes the quantity of abstract services included in an abstract workflow and M denotes the quantity of concrete services related to each abstract service, and a three-dimensional vector $X = (cs_1, \dots, cs_L) = (x_{111}, \dots, x_{1NM}, \dots, x_{L11}, \dots, x_{LNM})$ representing a solution of SSC problem, where L denotes the quantity of levels included in the problem. The component cs_i in solution X means a service composition which satisfies the i th level of SLA constraint, and component x_{ijk} means the number of instances for concrete service s_{jk} in service composition cs_i , where $i \in [1, L]$, $i \in \mathbb{Z}$; $j \in [1, N]$, $j \in \mathbb{Z}$; $k \in [1, M]$, $k \in \mathbb{Z}$. In order to be consistent with the former settings, we denote a vector $cs_i = (cands_{i1}, \dots, cands_{iN})$ to represent service composition cs_i , and its component is defined as follows.

Definition 1 (candidate service). A part vector of solution X associated with one abstract service, $cands_{ij} = (x_{ij1}, \dots, x_{ijM})$, is defined as a candidate service if and only if this part vector represents a deployment of concrete services associated with the same abstract service j in level i and it is not equal to zero.

QoS attributes of service composition such as throughput, latency, and cost are represented by Q_1 , Q_2 , and Q_3 , respectively. The end-to-end QoS attributes of a service composition are calculated by aggregating QoS measures of every service instances based on their structure. Each abstract service is interpreted as a collection of concrete services in redundant parallel and the aggregate functions are shown in Table 1. QoS attributes of a candidate service are calculated by aggregating QoS measures of corresponding redundant parallel concrete instance.

Table 2 shows aggregation functions among abstract services for different structure, and the notations $\prod_T$, $\prod_L$, and $\prod_C$ represent aggregation functions of throughput, latency, and cost, respectively.

Suppose there are three level user categories: platinum, gold, and silver users, so the quantity of levels is three, which is $L = 3$; and denote cs_1 , cs_2 , and cs_3 to represent three service compositions for platinum, gold, and silver user, respectively.

FIGURE 1: Service composition model for SSC.

TABLE 2: QoS aggregation function among abstract services.

QoS	Structure	Aggregation function
Throughput ($\prod T$)	Parallel	$\min_{j \in \text{Abstract services parallel connected}} \left(\sum_{k=1}^M x_{ijk} \times Q_1(s_{jk}) \right)$
	Sequence	$\min_{j \in \text{Abstract services connected in sequence}} \left(\sum_{k=1}^M x_{ijk} \times Q_1(s_{jk}) \right)$
Latency ($\prod L$)	Parallel	$\max_{j \in \text{Abstract services parallel connected}} \left(\frac{\sum_{k=1}^M x_{ijk} \times Q_2(s_{jk})}{\sum_{k=1}^M x_{ijk}} \right)$
	Sequence	$\sum_{j \in \text{Abstract services connected in sequence}} \left(\frac{\sum_{k=1}^M x_{ijk} \times Q_2(s_{jk})}{\sum_{k=1}^M x_{ijk}} \right)$
Cost ($\prod C$)	Parallel	$\sum_{j \in \text{Abstract services parallel connected}} \left(\sum_{k=1}^M x_{ijk} \times Q_3(s_{jk}) \right)$
	Sequence	$\sum_{j \in \text{Abstract services connected in sequence}} \left(\sum_{k=1}^M x_{ijk} \times Q_3(s_{jk}) \right)$

Suppose each abstract service is associated with three concrete services that operate at different QoS measures which are high-end (high throughput/latency and high cost), low-end (low throughput/latency and low cost), and intermediate (intermediate throughput, latency, and cost), so quantity of concrete services related to each abstract service is three, which is $M = 3$. According to the supposition, QoS attributes of three level service compositions are calculated as follows:

$$\begin{aligned}
 Q_1(cs_1) &= \prod_{j=1}^N T \left(\sum_{k=1}^M x_{1jk} Q_1(s_{jk}) \right), \\
 Q_2(cs_1) &= \prod_{j=1}^N L \left(\sum_{k=1}^M x_{1jk} Q_2(s_{jk}) \right), \\
 Q_3(cs_1) &= \prod_{j=1}^N C \left(\sum_{k=1}^M x_{1jk} Q_3(s_{jk}) \right), \\
 Q_1(cs_2) &= \prod_{j=1}^N T \left(\sum_{k=1}^M x_{2jk} Q_1(s_{jk}) \right), \\
 Q_2(cs_2) &= \prod_{j=1}^N L \left(\sum_{k=1}^M x_{2jk} Q_2(s_{jk}) \right), \\
 Q_3(cs_2) &= \prod_{j=1}^N C \left(\sum_{k=1}^M x_{2jk} Q_3(s_{jk}) \right), \\
 Q_1(cs_3) &= \prod_{j=1}^N T \left(\sum_{k=1}^M x_{3jk} Q_1(s_{jk}) \right), \\
 Q_2(cs_3) &= \prod_{j=1}^N L \left(\sum_{k=1}^M x_{3jk} Q_2(s_{jk}) \right), \\
 Q_3(cs_3) &= \prod_{j=1}^N C \left(\sum_{k=1}^M x_{3jk} Q_3(s_{jk}) \right).
 \end{aligned} \tag{1}$$

A vector of the SLA QoS constraints is $C = (C_1, \dots, C_7)$, which represents the multilevel constraint and is expressed in terms of a vector of upper or lower bounds for different QoS criteria. In SSC problem, there are a worst case throughput and latency for platinum and gold users, and there are a worst case and a cost limit on silver users; in addition there is a limit on the total costs incurred by all three user categories. And the SLA constraints are expressed in detail as follows:

$$\begin{aligned}
 Q_1(cs_1) &\geq C_1; & Q_2(cs_1) &\leq C_2, \\
 Q_1(cs_2) &\geq C_3; & Q_2(cs_2) &\leq C_4, \\
 Q_3(cs_3) &\geq C_5; & Q_3(cs_3) &\leq C_6, \\
 Q_3(X) &\leq C_7.
 \end{aligned} \tag{2}$$

SSC problem optimized in terms of three objectives; there are total throughput, latency, and cost incurred by all three user categories, which are expressed in detail as follows:

$$\begin{aligned}
 Q_1(X) &= Q_1(cs_1) + Q_1(cs_2) + Q_1(cs_3), \\
 Q_2(X) &= Q_2(cs_1) + Q_2(cs_2) + Q_2(cs_3), \\
 Q_3(X) &= Q_3(cs_1) + Q_3(cs_2) + Q_3(cs_3).
 \end{aligned} \tag{3}$$

Definition 2 (SLA-aware service composition). For a given abstract workflow ζ and the SLA QoS constraints, SLA-aware service composition is to find solutions related to three level user categories, which satisfy the global QoS constraints and optimize total three objectives.

3. Hybrid Multiobjective Discrete Particle Swarm Optimization

In order to use particle optimization algorithm to solve SSC problem, algorithm HMDPSO defined particle update strategy by introducing crossover operator to make global search of the solution space; it designed mutation strategy to prevent particles swarm from premature convergence; and the complete algorithm will be given in detail. In addition, in order to accelerate the process of obtaining feasible solution, algorithm HMDPSO+ was also be designed which was combined local-search strategy and algorithm HMDPSO.

3.1. The Design of Particle Position. Figure 2 shows an example of particle position that represents a solution of SSC problem. Each component encodes the number of instances for each concrete service. In this example, a workflow consists of four abstract services, $N = 4$; according to the assumption of SSC problem, every abstract service has three concrete services ($4 * 3 = 12$ components for one service composition) and a solution representing three service compositions for platinum, gold, and silver users ($12 * 3 = 36$ components).

3.2. Particle Update Strategy. The particle update includes particle velocity and particle position update, and in accordance with the characteristics of the SSC problem, HMDPSO introduces the crossover operator to redefine the way of particle velocity update in particle update strategy, as shown in Figure 3. Particle velocity updated through crossover among three particle positions is particle's individual best position, particle's global best position, and particle's current velocity. Every position will be divided into three parts by two intersection points, which are abstract service nodes, and also bounded to separate every candidate service. The new velocity is made up by three parts from three different particle positions. The new current position is obtained by crossover between the particle's current position and particle's new velocity.

It can be inferred from the particle update strategy that, like the traditional PSO algorithm, the behaviors of particle in particle update strategy of HMDPSO are affected by the current momentum, cognitive component, and social

FIGURE 2: An example of particle's position.

FIGURE 3: Particle velocity's update process.

component; and algorithm HMDPSO also possesses merits of algorithm PSO like easy calculating, fast convergences, and so on. Furthermore, this particle update strategy is also accordant with the feature of our problem model, since our crossover point will maintain the integrity of candidate services which is the minimum unit and represents the QoS attributes of abstract service. If the crossover point is in the middle of abstract service, some better candidate services will be destroyed and even changes on components of candidate service will happen in the mutation operator. Equations for particle updating are shown as follows:

$$\begin{aligned} V_i(k+1) &= V_i(k) \otimes P_{ibest} \otimes P_{gbest}, \\ P_i(k+1) &= P_i(k) \otimes V(k+1), \end{aligned} \quad (4)$$

where symbol $\otimes$ denotes crossover operator; (4) represents particle's velocity update equation and particle's position update equation, respectively.

3.3. Particle Mutation Strategy. It is known from the particle update process that every particle's move follows its individual best position and global best position, so the HMDPSO possesses the merits of quick convergence and simple computation. But the particle's current position will quickly approach the individual best position and global best position, namely, its three positions are the same and algorithm HMDPSO falls into local optimal solution. The reason for particles' falling into local optimal position is that the energy of particle swarm is decreasing and useful information is continuously lost; as a result, particles do

not have the ability to jump out of local optimum [10]. Consequently, swarm diversity indicator has been defined.

Definition 3. The swarm diversity indicator is calculated by summation of particles' energy in a swarm. And for a given particle P_i , $i \in [0, N-1]$, $i \in \mathbb{Z}$, its energy is computed as follows:

$$\text{energy}(P_i) = \frac{\sum_{u=1}^{\dim} \sum_{v=u+1}^{\dim} \text{same}(P_i(u), P_i(v))}{\dim * (\dim - 1)}, \quad (5)$$

where $P_i(u)$ is the u th component of particle P_i , and

$$\text{same}(P_i(u), P_i(v)) = \begin{cases} 0 & \text{if } P_i(u) = P_i(v), \\ 1 & \text{if } P_i(u) \neq P_i(v). \end{cases} \quad (6)$$

The value of swarm diversity indicator is the summation of every particle's energy:

$$\text{Diversity Calculation} = \text{Div} = \sum_{i=1}^N \text{energy}(P_i). \quad (7)$$

It can be inferred that, to a certain extent, the swarm diversity indicator can reflect the global search capability possessed by current particle swarm. When the swarm diversity is decreasing during iterations and its value is lower than the threshold α , it will induce a mutation on particles' individual best position. And the mutation operator is shown as follows:

$$P_{ibest}(k+1) = \text{mutation}(P_{ibest}(k+1)). \quad (8)$$


```

 $g \leftarrow 0$ 
 $P^0 \leftarrow$  randomly generated  $\mu$  particles ( $X_i(0), P_{ibest}(0)$ )
 $G_{best}^0 \leftarrow$  randomly generated  $\lambda$  particles ( $X_i(0)$ )
as global best solution
AssignFitnessValues( $G_{best}^0$ )
repeat until  $k = k_{max}$  {
   $Q^k \leftarrow \phi$ 
   $div = \text{CalculateDiversity}(P^k)$ 
  for each particle  $i$  in  $P^k$  {
    //mutation
    if ( $div < \alpha$ )
       $P_{ibest}(k) \leftarrow \text{Mutation}(P_{ibest}(k))$ 
    end if

    //gbest solution selection via binary tournament
     $G_a(k), G_b(k) \leftarrow \text{RandomSelection}(G_{best}^k)$ 
     $G_v(k) \leftarrow \text{BTSelection}(G_a(k), G_b(k))$ 

    //particle in  $P^k$  update
     $V_i(k+1) = V_i(k) \otimes P_{ibest}(k) \otimes G_v(k)$ 
     $P_i(k+1) = P_i(k) \otimes V_i(k+1)$ 

    //update  $P_{ibest}(k)$  by comparison with  $P_i(k+1)$ 
     $P_{ibest}(k+1) := P_i(k+1)$  if  $P_{ibest}(k+1) \nless P_i(k+1)$ 
    Add  $P_{ibest}(k)$  to  $Q^k$  if  $Q^k$  does not contain  $P_{ibest}(k)$ 
  }
  AssignFitnessValues( $G_{best}^k \cup Q^k$ )
   $G_{best}^{k+1} \leftarrow$  Top  $\lambda$  of  $G_{best}^k \cup Q^k$ 
   $k \leftarrow k+1$ 
}

```

PROCEDURE 1: HMDPSO.

New information is introduced into particle swarm through particle position mutation: it will increase particle swarm diversity, guide particles search the area that they never been, and prevent premature and earlier convergence.

3.4. HMDPSO. Procedure 1 shows the optimization process in HMDPSO. Particle swarm P^0 and global particle swarm G_{best}^0 are randomly initialized in the beginning; namely, particles in P^0 randomly initialize their current position $P_i(0)$, velocity $V_i(0)$, and individual best position $P_{ibest}(0)$; and particles in G_{best}^0 randomly initialize their current position.

In each iteration, swarm diversity indicator of P^k is calculated firstly, and then swarm particle updates and every new particle's individual best position will be added to particle swarm Q^k ; after that fitness of particles in $Q^k \cup G_{best}^k$ is computed by function Assign Fitness Value() and top λ particles with respect to their fitness value used to update G_{best}^k . When G_{best}^k is replaced by $G_{best}^{(k+1)}$, one iteration is ended, and HMDPSO repeats the above iteration for k_{max} times.

For particle i , swarm particle update can be divided into three parts; the first, when swarm particle diversity is lower than α , components of particle i 's individual best position

FIGURE 4: An example domination ranking.

are randomly altered by mutation(); the second, $G_v(k)$ is selected with binary tournaments from G_{best} , as combined with particle i 's velocity and individual best position $P_{ibest}(k)$, particle i 's new current position $P_i(k+1)$ is produced according to the particle update strategy; the third, it needs to make a comparison between $P_i(k+1)$ and $P_{ibest}(k)$ about *constraint domination* [4], when $P_{ibest}(k)$ is said to be *constraint domination* of $P_i(k+1)$, $P_{ibest}(k)$ stays the same; when $P_{ibest}(k)$ is said to dominate $P_i(k+1)$ or they are nondominated, $P_{ibest}(k+1)$ is replaced by $P_i(k+1)$ and $P_{ibest}(k+1)$ is added to particle swarm Q^k .

In fitness function, every particle can get its domination rank through *constraint domination* [11], and domination rank of 1 means the nondominated particles, and higher domination rank means being dominated by more particles. Domination value for particle i in HMDPSO is assigned by the summation of number of particles in its domination rank, the number of particles in the higher domination rank, and the number of particles it dominates, which is detailed in Figure 4.

Fitness values are assigned in the way of Procedure 2; for any feasible particle position satisfying SLA constraint, HMDPSO assigns the product of a domination value, distance from the worst point and sparsity as its fitness value; for any infeasible particle position not satisfying SLAs, the minus value of the individual's SLA violations divided by domination value as its fitness value is assigned. In practice, it uses a fine-grained domination value assignment strategy which incorporates individual's dominating information.

The way particle domination value is obtained in our algorithm is the same as that in NSGA-II [11], so both of their complexity are $O(mN^2)$, where m is the number of objectives and N is the size of the particle swarm. Distance from the worst point and sparsity both need $O(mN)$ computations, so the complexity of fitness function is $O(mN^2) + 2 * O(mN) = O(mN^2)$. And function mutation, random selection, and BT selection in swarm particle update all need $O(N)$ computations. Consequently, the overall complexity of the algorithm now is $K_{max}(O(mN^2) + O(N)) = O(mN^2K_{max})$.

```

AssignFitnessValue(P) {
  DominationRanking(P)
  for each particle's current position  $p_i$  in  $P$  {
    if  $p_i$  is feasible
      // fitness function for a feasible individual
       $f \leftarrow p_i$ 's domination value  $x$ 
       $p_i$ 's distance from the worst point  $x$ 
       $p_i$ 's sparsity
    else
      // for an infeasible individual
       $f \leftarrow 0 - p_i$ 's violation/ $p_i$ 's domination value
       $p_i$ 's fitness value  $\leftarrow f$ 
    }
  }
}

```

PROCEDURE 2: Fitness function in HMDPSO.

Since algorithm HMDPSO has the same fitness function with MOGA, the complexity of algorithm MOGA is $O(mN^2K_{\max})$ too. When the iteration number of NSGA-II is $K_{\max}$, its complexity is also $O(mN^2K_{\max})$; therefore, complexity of algorithm HMDPSO, MOGA, and NSGA-II is the same.

3.5. Local Search Strategy. In order to speed up the process of obtaining the feasible particle position and optimization process, a local search strategy is proposed based on *candidate service domination* and incorporated into the algorithm HMDPSO. And it updates particle position by updating every candidate service to accelerate improvement of its constraint satisfaction and optimize their objectives. Because there is need for seven in-equations to satisfy for a feasible solution in SSC problem when the workflow structure becomes complicated, infeasible solutions tend to be nondominated and it is difficult to obtain a feasible solution. Local search strategy updates every candidate service in a solution through *candidate service domination* to alleviate particles' evolutionary pressure and speed up the process of solution obtaining.

In SSC, every candidate service in a solution has QoS attributes like throughput, latency, and cost. According to the characteristic of aggregation functions in Table 1, for throughput, the service composition's throughput will satisfy its constraint only when its every candidate service's throughput satisfies its constraint; for latency and cost, when the value of these two QoS attributes of each candidate service is as small as possible, service composition related to candidate services will be easier to satisfy its constraints about these two QoS measures. Therefore, a candidate service i is said to dominate a candidate service j when any of the following conditions hold:

- (1) throughput of candidate service i satisfies its constraint;
- (2) value of candidate service i 's latency is lower than that of candidate service j ;
- (3) value of candidate service i 's cost is lower than that of candidate service j .

```

Input: particle  $i$ 's individual best solution  $P_{ibest}$ 
to be update and Global best particle  $k$ 's solution
 $P_{gbest}$  for updating
Output:  $P_{ibest}$  be updated
{
  for each abstract service  $j$  as  $P_{ibest}(S_j)$  and  $P_{gbest}(S_j)$ 
  in  $P_{ibest}$  and  $P_{gbest}$ 
    if ( $P_{gbest}(S_j) < P_{ibest}(S_j)$ )
      update  $P_{ibest}(S_j)$  with  $P_{gbest}(S_j)$ 
    endif
  end for
}

```

PROCEDURE 3: Local search.

Procedure 3 shows the process of local search based on *candidate service domination*, in which every candidate service in both P_{ibest} and P_{gbest} is compared on *candidate service domination* relationship. When a candidate service in P_{gbest} is said to be dominated by a candidate service in P_{ibest} , namely, $P_{gbest}(S_j) < P_{ibest}(S_j)$, replace $P_{ibest}(S_j)$ with $P_{gbest}(S_j)$.

In algorithm HMDPSO+, global best solution is randomly selected from particle swarm G_{best} , and it has been updated by comparison with mutated individual best particle position according to local search strategy. In addition, local search needs $O(N)$ computations, where N is the number of abstract service, so complexity of algorithm HMDPSO+ is also $O(mN^2K_{\max})$.

4. Experimental Design

To verify the effectiveness of the algorithm, this section will test the proposed algorithm HMDPSO and HMDPSO+ on four different scale-test cases and discuss the influence of parameters swarm_size and swarm diversity threshold α and then compare and analyze them with algorithms NSGA-II [11] and MOGA [4]. Experiments will evaluate how algorithms optimize service compositions, which is based on the objectives performances and index of hypervolume [12]. All algorithms are implemented in c language and executed on a core (TM) 2, 2.00 GHz, 3 GB RAM computer.

4.1. Test Cases. Three different structures of abstract workflow are been designed in this paper, which are related to four different test cases. For each test case, a set of solutions lying near Pareto-optimal front from both algorithms HMDPSO+ and HMDPSO will be compared with that solution sets from both algorithms NSGA-II and MOGA.

Because complexity of all compared algorithm is the same, the termination condition for all algorithms on each test case is set based on the maximum fitness-function evaluation number, which is set as $\text{Level} \times \text{Length} \times 10^4$, where Level is the total level number and Length is the total abstract service number in all levels.

The first abstract workflow includes four abstract services, which is associated with Case 1, and its structure is

TABLE 3: QoS attributes of concrete services.

Abstract service	Concrete service	Prob distribution and QoS measures			Cost
		Prob (%)	Throughput	Latency	
1	1	0.85	9000	60	90
		0.05	10000	50	
		0.05	6000	80	
		0.05	0	0	
	2	0.80	5500	60	50
		0.15	4000	100	
		0.05	0	0	
		0.30	2000	200	
	3	0.30	3000	180	10
		0.20	1500	250	
2	1	0.70	2000	20	50
		0.30	2300	18	
		0.90	4000	15	
		0.05	6000	13	
	2	0.05	3000	20	100
		0.70	4000	25	
		0.20	3000	23	
		0.05	2500	30	
	3	0.05	0	0	70
		0.05	0	0	
3	1	0.70	1500	30	30
		0.30	2000	20	
		0.80	3000	12	
		0.10	5000	20	
	2	0.10	500	80	80
		0.50	1000	60	
		0.30	500	50	
		0.20	0	0	
	3	0.75	2500	50	20
		0.25	3000	55	
4	1	0.90	6000	15	70
		0.05	4000	20	
		0.05	3000	20	
		0.85	1000	90	
	2	0.05	500	120	5
		0.05	100	150	
		0.05	0	0	
		0.05	0	0	
	3	0.80	5500	18	76
		0.10	2500	23	
5	1	0.10	1400	27	44
		0.75	3800	25	
		0.25	1500	35	
		0.85	3100	30	
	2	0.10	1000	45	30
		0.05	900	60	
	3	0.10	1000	45	
		0.05	900	60	

TABLE 3: Continued.

Abstract service	Concrete service	Prob distribution and QoS measures			Cost
		Prob (%)	Throughput	Latency	
6	1	0.80	2000	15	67
		0.15	1600	22	
		0.05	100	25	
		0.65	4600	35	
	2	0.25	3800	45	45
		0.10	2500	50	
		0.85	3100	40	
		0.15	2900	55	
	3	0.60	7800	35	80
		0.25	6500	65	
7	1	0.15	600	70	51
		0.70	5600	55	
		0.20	3900	65	
		0.10	300	70	
	2	0.80	1900	140	10
		0.20	900	175

FIGURE 5: Case 2 abstract workflow.

shown as in Figure 1. And each abstract service is associated with three concrete services that operate at different QoS measures in Table 3: high-end (high throughput/latency and high cost), low-end (low throughput/latency and low cost), and intermediate (intermediate throughput/latency and low cost). Data set for Case 1 includes the concrete services and their distribution associated with the former four abstract services.

The SLA for Case 1 is illustrated in Table 4, and platinum and gold users have SLA on their throughput and latency; silver users have SLA on their throughput and cost; and there is a SLA on their total cost. Objective functions of Case 1 are obtained according to its workflow structure as in Figure 1, and objective formula (3); in the same way, SLA constraint functions are obtained according to SLA constraint for Case 1 in Table 4 and constraint formula (2). Termination condition of Case 1, the maximum fitness function evaluation number, is set as 1.2×10^5 . All results are the average of 10 independent runs.

The second abstract workflow includes seven abstract services, which is related to Case 2 and its structure, including branch structure, as shown in Figure 5. This branch structure means that when one abstract service execution is completed, only one of several abstract services connected with it will be executed. Each abstract service in Case 2 is also associated with three concrete services operating at different

TABLE 4: SLA constraints for Case 1.

Users	Constraints (upper/lower bound)			Total cost (upper)
	Throughput (lower)	Latency (upper)	Cost (upper)	
Platinum	12000	100	—	2000
Gold	6000	130	—	
Silver	2000	—	250	

TABLE 5: SLA constraints for Case 2.

Users	Constraints (upper/lower bound)			Total cost (upper)
	Throughput (lower)	Latency (upper)	Cost (upper)	
Platinum	12000	150	—	3000
Gold	6000	195	—	
Silver	2000	—	375	

TABLE 6: Concrete service for Cases 3 and 4.

Users	QoS measures		Cost
	Throughput	Latency	
1	10000	60	100
2	5500	100	50
3	2000	200	20

TABLE 7: SLA constraints for Cases 3 and 4.

Users	Constraints (upper/lower bound)			Total cost (upper)
	Throughput (lower)	Latency (upper)	Cost (upper)	
Platinum	40000	80 * M	—	1000 * M
Gold	20000	120 * M	—	
Silver	15000	—	200 * M	

QoS measures, and data set and SLA constraint are shown in Tables 3 and 5, respectively.

In the same way, objective functions of Case 2 are obtained according to its workflow structure as in Figure 4, and objective formula (3); SLA constraint functions are obtained according to SLA constraint for Case 2, as in Table 5, and constraint formula (2). Termination condition for Case 2, the maximum fitness function evaluation number, is set as 2.1×10^5 . All results are the average of 10 independent runs.

The third abstract workflow consists of a sequence of various numbers of abstract services, which is related to Case 3, and Case 4; and Case 3 includes 10 abstract services while Case 4 includes 15. In Case 3 and Case 4 each abstract service is associated with 3 concrete services whose QoS measures are constant as in Table 6. SLA constraint for Case 3 and Case 4 is shown in Table 7, where M represents the number of abstract services included in the third abstract workflow. Objective functions of Case 3 and Case 4 are obtained according to their workflow structure and objective formula (3); SLA constraint functions are obtained according to SLA constraint for Case 3 and Case 4 as in Table 7 and constraint formula (2). Termination conditions for Case 3 and

Case 4, the maximum fitness function evaluation number, are set as 3.0×10^6 and 4.5×10^6 , respectively; all results are the average of 10 independent runs.

Assume that each concrete service has at most 10 instances, so a service composition for one user category has a search space of $(10^3 - 1)^M$, and the search space is expanded to $(10^3 - 1)^{3M} \approx 1 \times 10^{9M}$ for a solution representing three categories of users. This number indicates the total number of possible combinations of service instances. The search space of the workflow in Case 1 is approximately 1×10^{36} , the search space of the workflow in Case 3 is approximately 1×10^{90} , and the search space of the workflow in Case 4 is approximately 1×10^{135} . This analysis shows that the SSC problem tends to have a huge search space even if small numbers of abstract and concrete services are involved.

4.2. Parameter Turning and Convergence Analysis. In our proposed algorithms HMDPSO and HMDPSO+, the main parameters are particle swarm size N_s and the swarm diversity threshold α . The influence of parameter N_s on the algorithms' performance is obvious if the complexity is not taken into account, and when N_s is larger than a certain value, the particle swarm will be difficult to converge and reduce the performance of algorithms. The range of parameter N_s is set to be from 50 to 200 with an increment of 25. Parameter α is more sensitive than N_s , so its range is set to be from 0.1 to 0.4 with an increment of 0.05 and its default value is set as $\alpha = 0.35$.

In order to perform parameter exploration studies, all four test cases, associated with four different data sets, are involved. But the hypervolumes of solution sets obtained from these test cases are not of the same order of magnitude, so they will be converted into R -values using the following formula according to different test cases:

$$r_{ij}^k = \frac{H_{ij}^k}{(10 * G_{ij}(\max_k(H_{ij}^k)))}, \quad (9)$$

where H_{ij}^k is the hypervolume of solution set obtained from algorithm HMDPSO, when it runs on case i and the parameter j is set as the k th value, and $i \in \{1, 2, 3, 4\}$, $j \in \{1, 2\}$, $k \in [1, 7]$, $k \in \mathbb{Z}$. When i and j stay the same for all the possible values of k , we can get the maximum H_{ij}^k , namely, $\max_k(H_{ij}^k)$; $G_{ij}(\max_k(H_{ij}^k))$ represents the quantity level of $\max_k(H_{ij}^k)$; for example, the quantity level of 0.005 is 0.001.

In this process of parameter turning, swarm-size N_s will be turned first and then the parameter α . For the parameter N_s , we vary its value one at a time, while setting the value of parameter α to its default value; for the next unturned parameter α , we vary its value one at a time, while setting the value of turned parameter N_s to the obtained most appropriate ones. During the process, the algorithm HMDPSO with each parameter configuration runs 20 times on each used test case; the result is converted by formula (9) and shown in Figure 6. From Figure 6(a), we can see that the maximum R -value for Case 1, Case 2, and Case 3 is obtained

FIGURE 6: The effects of different parameter configurations.

FIGURE 7: HMDPSO's convergent curves in test cases.

when $N_s = 100$; and algorithm HMDPSO has the similar R -value in Case 4 for $N_s = 100$ and $N_s = 125$. The maximum R -value for all cases is obtained when $\alpha = 0.30$ as shown in Figure 6(b). So the comparatively better settings for these parameters are $N_s = 100$ and $\alpha = 0.30$ for the algorithm HMDPSO.

At each run, the hypervolume of sets of solution lying near Pareto-optimal front obtained during each interval is recorded. Then the hyper-volume values achieved at the corresponding time intervals in the 20 runs of algorithm HMDPSO on four test cases are averaged. In order to compare these averaged hypervolume values in similar order of magnitude, they are all converted into r -values by formula (9). By this way, the obtained convergent lines for algorithm HMDPSO on all cases are shown in Figure 7, which depicts

the averaged r -value and reflects the effect of the algorithm's hyper-volume. We can see that algorithm HMDPSO has different convergent performances on different test cases. In Cases 1 and 2, the algorithm achieves convergent value faster than that in Case 3 and Case 4; that is because compared with Cases 1 and 2, Cases 3 and 4 in Section 4.1 are more complicated, like including more abstract services, and have relatively more strict SLA constraints.

At the same time, hypervolumes of algorithm HMDPSO on all test cases have fluctuations before they get their convergent values, because particles' behaviors are affected by different strategies and fitness function, and all their effects cannot fit perfectly with the effects of hypervolume's computation rule on solution sets. Furthermore, hypervalue obtained from algorithm HMDPSO on test cases are converged to a certain value after 3000 iterations.

4.3. Compared with Recently Proposed Algorithms. For SSC problem, both algorithms HMDPSO (hybrid multiobjective discrete particle optimization algorithm) and HMDPSO+ (hybrid multiobjective discrete particle optimization algorithm incorporated with local search) are compared with recently proposed algorithms MOGA and NSGA-II on four cases which are constructed in former section with different abstract workflow structures. The parameters of the algorithms HMDPSO+ and HMDPSO and termination conditions of all these algorithms are set as in Section 4.1. Experiment was displayed into two parts: the first part displays how objectives are optimized by all compared algorithms, and the solution set distribution is shown in Figure 7; the second part compares hyper-volume of all algorithms on different test cases which are shown in Figure 9.

(1) Objective Optimization. In this part, solution sets obtained from algorithms HMDPSO+, HMDPSO, NSGA-II, and MOGA are compared on four different cases set

FIGURE 8: The solution sets distribution on different cases.

as before. The 100-solution sets are selected with largest hyper-volume among the results in 20 independent runs on four different cases, and every solution is associated with three objectives which are normalized and the larger, the better. All the statistical experiment results of the mean (mean) and standard deviation (STD) of all three objectives associated with solution sets are summarized in Table 8 among HMDPSO+, HMDPSO, MOGA, and NSGA-II. The two-tailed t -test is used to verify whether every objective of HMDPSO+, HMDPSO, MOGA, and NSGA-II has distinctive difference with 99 degrees of freedom and significance level $\alpha = 0.05$.

As observed from Table 8, we can see that the items of “all three objectives” of algorithm HMDPSO+ are much

better than those of algorithms HMDPSO, MOGA, and NSGA-II. The values of the test statistics between algorithms HMDPSO+ and HMDPSO and MOGA and NSGA-II are given as the t -test column in the Table 8, which shows that the mean results between algorithm HMDPSO+ and algorithm HMDPSO, MOGA/NSGA-II have significant difference at the 5% level. So solutions produced by algorithm HMDPSO+ are obviously better than that produced by algorithm HMDPSO, MOGA, and NSGA-II, on all objectives and all cases, and this is further proved by Figure 7.

Solution sets with largest hypervolume are also plotted in a three-dimensional coordinate system according to different cases in Figure 8. And we can see that the effect of solution sets plotted in Figure 8 is accordant with the analysis in

TABLE 8: Performance comparisons among algorithms HMDPSO+, HMDPSO, MOGA, and NSGA-II with all objectives of composite web service in different sets. The value of t with 99 degrees of freedom is significant at $\alpha = 0.05$ by a two-tailed test.

Cases	Items	Objective	Cost	t -test	Latency	t -test	Throughput	t -test
Case 1	Mean (STD)	NSGA-II	0.2220 (0.2191)	12.48	0.3309 (0.1661)	24.68	0.2261 (0.1486)	13.105
		MOGA	0.3721 (0.2312)	6.909	0.5953 (0.2219)	9.286	0.3001 (0.1651)	8.798
		HMDPSO	0.3122 (0.2210)	9.367	0.5470 (0.1938)	12.23	0.3301 (0.1608)	7.685
		HMDPSO+	0.6046 (0.2187)		0.8265 (0.1128)		0.5137 (0.1439)	
Case 2	Mean (STD)	NSGA-II	0.3748 (0.2715)	7.517	0.2156 (0.0855)	11.232	0.2004 (0.1429)	14.631
		MOGA	0.4288 (0.2691)	5.976	0.3161 (0.1803)	7.134	0.2602 (0.1591)	11.86
		HMDPSO	0.4676 (0.2478)	5.084	0.3779 (0.2215)	5.037	0.4194 (0.2058)	5.254
		HMDPSO+	0.6329 (0.2101)		0.5656 (0.2996)		0.5762 (0.2033)	
Case 3	Mean (STD)	NSGA-II	0.2852 (0.2091)	9.130	0.3405 (0.1724)	24.696	0.2374 (0.1754)	9.822
		MOGA	0.3823 (0.2327)	6.243	0.4533 (0.0956)	27.835	0.3664 (0.1707)	5.324
		HMDPSO	0.4178 (0.1966)	5.604	0.5233 (0.1821)	14.718	0.3383 (0.1774)	6..237
		HMDPSO+	0.6113 (0.2836)		0.8242 (0.0927)		0.5156 (0.222)	
Case 4	Mean (STD)	NSGA-II	0.3424 (0.2526)	9.057	0.3424 (0.1186)	42.87	0.2838 (0.1445)	9.144
		MOGA	0.4248 (0.2825)	6.264	0.4873 (0.1174)	32.038	0.3416 (0.1985)	6.462
		HMDPSO	0.4147 (0.1860)	8.055	0.3908 (0.1283)	36.58	0.4874 (0.2290)	1.817
		HMDPSO+	0.6560 (0.2290)		0.9013 (0.0479)		0.5496 (0.2470)	

Table 8. For example in Case 1, mean of “all three objectives” of algorithm HMDPSO+ is better than those of algorithms HMDPSO, MOGA, and NSGA-II as in Table 8, and the coordinate value of solutions from algorithm HMDPSO+ is relatively larger than those of the compared algorithm. The STD value of “all three objectives” is related to the degree of distribution concentration, since solution set from algorithm HMDPSO+ with relatively smaller STD value, solutions in that set distributes more concentrate as in Figure 8 accordingly.

The t -test value of “all three objectives” represents the level of significance of differences and the larger the t -test value is, the bigger the difference is. The difference of “all three objectives” between algorithm HMDPSO+ and algorithm NSGA-II is the largest among all the differences of “all three objectives” between algorithm HMDPSO+ and other compared algorithms as in Table 8; accordingly, we can see that distribution of solution sets obtained from algorithm HMDPSO+ is roughly farthest from the distribution of solution sets obtained from algorithm NSGA-II. The t -test values of “all three objectives” between algorithm HMDPSO/MOGA and algorithm HMDPSO+ are near as in Table 8, and distribution of solution sets obtained from algorithms HMDPSO and MOGA overlaps each other with some area, and they are relatively the same far away from the distribution of solution sets obtained from algorithm HMDPSO+. As far as other cases are considered, similar situations appear again that algorithm HMDPSO+ greatly outperforms algorithms HMDPSO, MOGA, and NSGA-II and the superiority of MDOSP+ over HMDPSO, MOGA, and NSGA-II is more and more clear.

These results indicate that algorithm HMDPSO+ with local search strategies proposed in this paper has powerful

search ability and excellent convergence property when compared with algorithms HMDPSO, MOGA, and NSGA-II for SLA-aware service composition problem with different scales and different workflow structures.

(2) *Hypervolume Comparison.* In this part, we compare the hyper-volumes of solution sets from algorithms HMDPSO+, HMDPSO, MOGA, and NSGA-II on 4 different scale-test cases set as in Section 4.1. We run each algorithm twenty times on each test case. The maximum hypervolume, minimum hyper-volume, mean, and the coefficient of variation obtained by each compared algorithm in the twenty runs on each instance are given in Table 9. Coefficient of variation, which is dimensionless and scale invariant variance, is defined as the ratio of the standard deviation: $CV = \sigma/\mu$, where σ and μ are the standard deviation and the mean of QoS values, respectively. Lower CV means that solutions are more equally distributed over the objective space. We can see that the maximum hyper-volume, minimum hyper-volume, and mean value obtained by the algorithm HMDPSO+ for each test case are larger than those obtained by the other compared algorithms. It also has achieved relatively the smallest CV values for all cases. Therefore, we can clearly get that the algorithm HMDPSO+ is more stable than the other compared algorithms and can perform better than all the other compared algorithms. This can be further proved by Figure 9, which explicitly shows the statistical results using the box plot based on the hyper-volume obtained by the compared algorithms on each test case. It gives the distribution of the hyper-volume obtained by each algorithm, including the smallest observation, lower quartile, median, mean, upper quartile, and the largest observation. We can see that all these statistical

TABLE 9: The hypervolume obtained by the compared algorithms [max/min/ave(cv%)].

Algorithm	Case 1	Case 2
NSGA-II	0.12134/0.03124/0.08554 (0.30647)	0.18884/0.00329/0.08504 (0.52292)
MOGA	0.2392/0.09813/0.16818 (0.26302)	0.34518/0.08022/0.20868 (0.42632)
HMDPSO	0.39121/0.21005/0.28248 (0.14920)	0.40814/0.22588/ 0.31253 (0.18645)
HMDPSO+	0.49362/0.29415/0.39474 (0.12102)	0.57412/0.38251/0.46592 (0.13765)
Algorithm	Case 3	Case 4
NSGA-II	1.7816/0.13241/0.741601 (0.65037)	0.91408/0.01245/0.3163 (0.91166)
MOGA	2.89841/0.178023/1.52789 (0.53576)	2.03127/0.21215/1.03266 (0.55141)
HMDPSO	3.33161/1.21582/2.48222 (0.42745)	3.14132/1.17618/2.0651 (0.295435)
HMDPSO+	4.78321/2.27678/3.65611 (0.25738)	4.98451/2.60531/3.75521 (0.14593)

FIGURE 9: The statistical results of hypervolume of the compared algorithms on different cases.

data obtained by the algorithm HMDPSO+ on each case are obviously larger than that of other compared algorithms. At same time, we also can see that algorithm HMDPSO is always better than the compared algorithms except algorithm HMDPSO+ and the distribution of its statistical data is more concentrated than that of algorithms MOGA and NSGA-II, which makes algorithms HMDPSO+ and HMDPSO more stable than algorithm MOGA. Furthermore, as mentioned

before, different cases are associated with different abstract workflows and different scales, and we can see that algorithms HMDPSO and HMDPSO+ have good performance on all cases.

The main reasons for algorithm HMDPSO's good performances lie in fitness function's guide effect in the optimization process and global search capacity of algorithm PSO. Compared with fitness function in MOGA, our fine-grained

fitness function will assign different fitness value to particles in the same domination rank to prevent particles with better QoS attributes from being discarded and improve the accuracy of particles distinguishing. When particles represent infeasible solution, this fitness function will accelerate the feasible solution searching process; when particles represent feasible solution, it will improve the optimization effects of our algorithm. Optimization process in NSGA-II only focuses on the nondominated individuals, and its accuracy of solution distinguishing is even lower than that of MOGA, so algorithm HMDPSO performances better than these two algorithms on SSC problem.

And the reason for algorithm HMDPSO+'s better performances lies in merits inherited from algorithm HMDPSO and its incorporation with local search strategy. Local search strategy can update poor QoS attribute candidate service with better candidate service and keep the good QoS attribute candidate service for a particle; then, it will decrease the possibility of a particle with good QoS attribute candidate service to be discarded and increase the possibility of a particle with poor QoS attribute service entering next iteration to promote the optimization performance; more than that, efficiency of algorithm will not be reduced after the incorporation with local search strategy, since its complexity is $O(n)$. So, we can conclude that the algorithm HMDPSO+ (algorithm HMDPSO incorporated with local search strategy) outperforms the compared methods in terms of the hypervolume and possesses competitive performance for the large scale service selection problem with all kinds of connection structures.

5. Conclusions

For the SLA-aware service composition problem, multiobjective discrete particle optimization algorithm (HMDPSO) is proposed and its improved algorithm HMDPSO+ which incorporates local search strategy into algorithm HMDPSO is also designed. In algorithm HMDPSO, particle update strategy is defined to search the solution space in global range. Particle mutation strategy is proposed, and together with it, indicator of swarm diversity is defined. When swarm diversity is lower than the threshold of swarm diversity α , the individual best position will be mutated to introduce new information, increase the swarm diversity, and restrain particle swarm's premature convergence. And its fitness function also has positive effect on algorithm HMDPSO's optimization process. In algorithm HMDPSO+, the incorporated local search strategy improves the degree of particle position's constraint satisfaction in every abstract service, and then it accelerates the process of obtaining the feasible particle position and optimizes all the three objectives of particle positions. In the end, these two algorithms HMDPSO+ and HMDPSO are compared with recently proposed algorithms MOGA and NSGA-II on four cases with different scales and workflow structures, and the experiments illustrate that algorithm HMDPSO+ has an excellent performance in terms of powerful searching ability, excellent convergence ability, and stable performance.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is sponsored by NSFC Major Research Program (nos. 61073062, 61100090, and 61100027) and the Special Fund for Fundamental Research of Central Universities of Northeastern University (no. 110204006).

References

- [1] Y. Yin, B. Zhang, and X.-Z. Zhang, "An active and opportunistic service replacement algorithm orienting transactional composite service dynamic adaptation," *Chinese Journal of Computers*, vol. 33, no. 11, pp. 2147–2162, 2010.
- [2] V. Cardellini, E. Casalicchio, V. Grassi et al., "Moses: a framework for qos driven runtime adaptation of service-oriented systems," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1138–1159, 2012.
- [3] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Transactions on the Web*, vol. 1, no. 1, article 6, 2007.
- [4] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.
- [5] Y.-M. Xia, B. Cheng, J.-L. Chen, X.-W. Meng, and D. Liu, "Optimizing services composition based on improved ant colony algorithm," *Chinese Journal of Computers*, vol. 35, no. 2, pp. 270–281, 2012.
- [6] H. Wada, J. Suzuki, Y. Yamano et al., "E³: a multiobjective optimization framework for SLA-aware service composition," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 358–372, 2012.
- [7] S.-G. Wang, Q.-B. Sun, and F.-C. Yang, "Web service dynamic selection by the decomposition of global QoS constraints," *Journal of Software*, vol. 22, no. 7, pp. 1426–1439, 2011.
- [8] F. L. Huang, S. C. Zhang, and X. F. Zhu, "Discovering network community based on multi-objective optimization," *Journal of Software*, vol. 24, no. 9, pp. 2062–2077, 2013.
- [9] W. Tao, S. Guo-Jun, and G. Quan, "Web service composition based on modified particle swarm optimization," *Chinese Journal of Computers*, vol. 36, no. 5, pp. 1031–1046, 2013.
- [10] Z. Chang, "A hybrid algorithm for flow-shop scheduling problem," *Acta Automatica Sinica*, vol. 35, no. 3, pp. 332–336, 2009.
- [11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [12] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, 2012.