International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **12**, December **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 8527–8544

A HYBRID CLONAL SELECTION ALGORITHM FOR QUALITY OF SERVICE-AWARE WEB SERVICE SELECTION PROBLEM

Xinchao Zhao[1], Panyu Huang[2], Tingting Liu[3] and Xingmei Li[4]

[1]School of Science
Beijing University of Posts and Telecommunications
No. 10, Xitucheng Road, Haidian District, Beijing 100876, P. R. China
xcmmrc@gmail.com

[2]Super Computing Center, Computer Network Information Center
Chinese Academy of Sciences
Beijing 100190, P. R. China

[3]Zhengzhou Information Science and Technology Institute
Zhengzhou 450012, P. R. China

[4]Business Administration School
North China Electric Power University
No. 2, Beinong Road, Huilongguan, Beijing 102206, P. R. China

Received October 2011; revised February 2012

Abstract. *It is imperative to provide service consumers with facilities for selecting required web services according to their Quality of Service (QoS). Several heuristics, which are local adaptation preemptive strategy, elitist perturbation guiding strategy and global tournament strategy for directed local fine-tuning, and a hybrid clonal selection algorithm (CSPLPT) are proposed for QoS-aware web service selection problem. It combines the population-based local search mechanism of clonal selection algorithm and population-based elitist learning mechanism of particle swarm optimization algorithm. The hybridization idea and the efficacy of the proposed strategies are verified and analyzed step by step with web service selection problem. The proposed algorithm CSPLPT is further compared with recent genetic algorithm-based CoDiGA and particle swarm optimization-based iPSOA algorithms with various scales of composite web services. The comprehensive experiments demonstrate the necessity and superiority of the proposed strategies. It also shows that the hybrid algorithm CSPLPT consistently and significantly outperforms CoDiGA and iPSOA in better QoS performance and average evolutionary behaviors while selecting the composite web services with global QoS constraints.*
Keywords: Clonal selection, Particle swarm optimization, Web service, QoS-aware

1. **Introduction.** Service-Oriented architecture enables a multitude of service providers to provide loosely coupled and interoperable web services (WS) at different Quality of Service (QoS) and cost levels. It has become a promising technology [1] for the development of new Internet-based software systems. Some interoperation mechanisms, including service description, service discovery and service binding, are enabled in service-oriented architecture [2] and web services have been largely attended by the academic and industrial domain in recent years [3]. The increasing popularity of employing web services for distributed systems contributes to the significance of service discovery. However, duplicated and similar functional features existing among services require service consumers to include additional aspects to evaluate the services. Generally, the service consumers would have different views on the non-functional characteristics or QoS of service attributes.

How to select the best composite service in theory among available WS candidates for consumers is an interesting issue [4, 5].

Making the provision of services Quality of Service-aware has advantages to both clients and providers [6]. Meanwhile, QoS concerns are becoming crucial to the global success of the WS-based computing paradigms. If the discovered services are provided in various versions, each demonstrating a different set of non-functional properties, then the automated service selection and composition that take place, may consider the QoS preferences of clients in order to optimize their experiences regarding features such as price, availability, reliability, and reputation [7]. On the other hand, QoS can give service providers a significant competitive advantage in the e-business domain, as QoS-aware services meet the user needs better and thus attract more customers. It makes sense to investigate mechanisms to properly select a set of services that satisfy QoS needs and cost constraints of the business process. This problem is referred to in literature as QoS-aware service selection or optimal service selection problem [8-17].

This paper proposes a hybrid clonal selection algorithm for web service selection problem to help service requesters choose the satisfiable composite WSs by taking nonfunctional characteristics (QoS) into consideration. It combines the population-based local search mechanism of clonal selection algorithm (CSA), population-based elitist learning mechanism of particle swarm optimization (PSO), and several other effective heuristics for WS selection problem. Clonal selection algorithm is a population-based local behaviors biasing better solutions, which have different search abilities based on their affinity and, however, have no information communications among solutions. PSO [28] is a population-based elitist learning mechanism from the global/local best particle. It has abundant information exchanging among solutions. However, the neighboring information of the better but not the best solutions is not fully exploited. This is the inherent reason to hybridize them and several heuristic strategies as well for the WSs selection problem. Local adaptation preemptive strategy (LAPS) is inspired by the roulette selection operation of genetic algorithm [29], which means that the larger QoS score of a candidate service has, the higher the probability to be selected is for a task. Elitist perturbation guiding strategy (EPGS) is presented to let the slightly perturbed global best solution guide the search directions of all other solutions. This diversifies the path following strategies along the perturbed best solution and can overcome the prematurity on the basis of elitist learning. At the latter stage of algorithm selection operation of CSA is substituted by global tournament strategy (GTS), which is proposed to rectify the possible improper search direction guiding from a macrostructure of algorithm.

The remainder of this paper is organized as follows. Section 2 introduces some related studies, including some existing works for web service, particle swarm optimization algorithm and artificial immune system. Section 3 presents the QoS terms and computation model. Section 4 describes QoS components for web service selection algorithm. Section 5 presents the improved heuristic strategies and hybrid clonal algorithm. These heuristics are verified step by step based on web service selection benchmark. Section 6 compares the proposed hybrid algorithm CSPLPT with recent GA-based CoDiGA and PSO-based iPSOA algorithms through extensive experiments and analysis. Finally, Section 7 offers the concluding remarks.

2. **Related Studies.**

Web services enable business applications running on distinct platforms and exchanging data over Internet, to be applied in business and daily life regardless of the platforms or locations. Quality-of-service (QoS) in web services encompasses various non-functional issues such as performance, dependability and security. As more

and more web services become available, QoS capability is becoming a decisive factor to distinguishing services.

2.1. **Some solutions for web service selection.** Luo et al. [9] proposed a service composition model with end-to-end QoS constraints and an enhanced cross entropy heuristics based on the observation of characteristic of end-to-end service composition as a novel solution. Simulation results and comparison with other algorithms reveal its better performance. Menascé et al. [12] considered the problem of finding the set of service providers that minimizes the total execution time of the business process subject to cost and execution time constraints. They presented an optimization algorithm that finds the optimal solution without having to explore the entire solution space. The algorithm can be used to find the optimal solutions in problems of moderate size. Wang et al. [13] proposed an improved PSO algorithm to solve WS selection problem and obtained satisfiable results with adaptive weight adjustment and non-uniform mutation strategies. Based on QoS measurement metrics, Hwang et al. [16] proposed multiple criteria decision making and integer programming approaches to select the optimal service with an efficient service selection scheme by considering two different contexts: single QoS-based service discovery and QoS-based optimization of service composition. Wang et al. [34] considered not only the objective factors described by service providers but also the subjective information with trustability evaluations from users who use those services. With a fuzzy query interface to input subjective and objective factors, users can determine the most suitable web service for personal use. Skoutas et al. [35] proposed a service selection framework that integrates the similarity matching scores of multiple parameters obtained from various matchmaking algorithms. The framework relies on the service dominance relationships to determine the relevance between services and users' requests.

2.2. **Particle swarm optimization algorithm.** A standard particle swarm optimizer [36] maintains a swarm of particles and each individual is composed of three D-dimensional vectors, where D is the dimensionality of the search space. These are the current position x_i, the previous best position p_i, and the velocity v_i. The current position $x_i = (x_{i,1}, \cdots, x_{i,D})$ can be considered as a set of coordinates describing a point in space. The best solution found so far is stored in $p_i = (p_{i,1}, \cdots, p_{i,D})$. New points are chosen by adding $v_i = (v_{i,1}, \cdots, v_{i,D})$ coordinates to x_i, and the algorithm operates by adjusting v_i, which can be seen as a step size.

In essence, the trajectory of each particle is updated according to its own flying experience as well as to that of the best particle in the swarm. Experimental results suggest that it is preferable to initialize the inertia weight to a large value (usually less than 1), giving priority to global exploration of the search space, and gradually decreasing so as to obtain refined solutions [37]. Furthermore, the value of ω and c_1 and c_2 should be set dependently on one another.

$$v_{i,d}^{k+1} = \omega v_{i,d}^{k} + c_1 r_{i1}^{k} \left(p_{i,d}^{k} - x_{i,d}^{k}\right) + c_2 r_{i2}^{k} \left(p_{g,d}^{k} - x_{i,d}^{k}\right) \tag{1}$$

$$x_{i,d}^{k+1} = x_{i,d}^{k} + v_{i,d}^{k+1} \tag{2}$$

where $v_{i,d}^{k}$ is the d-th dimension velocity of particle i in cycle k; $x_{i,d}^{k}$ is the d-th dimension position of particle i in cycle k; $p_{i,d}^{k}$ is the d-th dimension of personal best (pbest) of particle i in cycle k; $p_{g,d}^{k}$ is the d-th dimension of the gbest in cycle k; ω is the inertia weight; c_1 is the cognition weight and c_2 is the social weight; r_{i1}^{k} and r_{i2}^{k} are uniform random numbers in $[0, 1]$.

2.3. **Artificial immune optimization algorithm.** The immune system is mobile, highly distributed and coordinated system. The immune components have high diversity so that they can detect various antigens and so the immune system is highly robust. Therefore, the population of immune components is self-stabilized. Some computational models and applications based on artificial immune systems (AIS) [18, 19] have become a research frontier, such as immune clonal system, immune network system, and negative selection algorithm [20, 21]. These models are widely applied to information security, optimization, robotic control, recognition, data mining, abnormal detection and diagnosis [22] and so on.

CLONALG [23] is the first popular clonal selection algorithm to select part fittest antibodies and clone proportionally to their antigenic affinities. The hypermutation operator performs an affinity maturation process inversely proportional to the fitness values generating the matured clone population. Based on the Baldwin effect, a Baldwinian learning clonal selection algorithm is proposed to deal with numerical optimization problems [24]. The Baldwinian learning operator simulates the learning mechanism in immune system by employing information from the antibody population to alter the search space. Chen et al. [25] developed a hybrid immune multiobjective optimization algorithm based on clonal selection principle. A hybrid mutation operator is proposed with the combination of Gaussian and polynomial mutations (GP-HM operator). Inspired by enhanced index tracking in portfolio management Li et al. [26] proposed a multi-objective optimization scheme which provides a framework of defining the objectives as both maximizing the degree of beating the benchmark index and minimizing the accumulated error of underperforming the benchmark. When dynamic optimization [27] is considered optimization algorithms are to force their readiness for continuous search for new optima occurring in changing locations. Various immune principles are implemented, and immune-based algorithms are proposed and compared on complex environments. Motivated by the negative selection mechanism in biological immune recognition a negative selection algorithm is proposed [20]. Different from the existing immune optimization methods, negative selection algorithm constantly removes the worst solutions and thus it might lead to the loss of design information, which actually help identify better solutions in the search space.

3. **QoS Terms and Model.** QoS is crucial in determining which service best meets clients' desires and objectives. Web service description language [38] has provided a standard model to specify service functionality by separating abstract representations of service input and output messages from the concrete descriptions of end point's bindings.

3.1. **QoS attributes of web service.** This paper will not illustrate the non-functional concerns of WSs with a long list of QoS attributes, but to present several normally researched indexes, such as Cost (C), Time (T), Availability (A) and Reliability (R).

The cost (C) is defined as the fee that a service requester has to pay to the service provider for the service invocation. It is always associated with the value of the service's functionality, i.e., the more complicated functions it provides, the more a service costs. Availability (A) of web service is the probability that the service operation is accessible, which is defined by the proportion of the service's uptime to downtime. Response time (T) is the expected delay between the time instant when a request is sent and the time when the result is obtained. Reliability (R) is a measure of the service invocation trustworthiness. It is defined as the ratio between the numbers of service invocations which comply the negotiated QoS over the total number of service invocations.

QoS is the collective effort of service performance, which determines the degree of satisfaction to the service of a user. "The degree to which a system, component or process

meets customer or user needs or expectations" [31] and "The standard of something when compared to other things like it; how good or bad something is" [32]. Sullivan et al. [33] claimed that complete service descriptions are helpful to service discovery, management, negotiation, composition, and substitution, and that a service is completely represented if both functional and non-functional aspects are described well. However, they did not demonstrate the use of identified non-functional properties nor propose corresponding solutions to the aforementioned operations.

3.2. **QoS terms.** Task is the basic unit of a composite service and each task finishes its corresponding component function. The component functions finished by each task comprise the specific function of a composite service, whose QoS attributes are calculated by the component candidate services. Candidate service provided by different service providers, has different values of nonfunctional attributes. All the candidate services in the same task can finish the corresponding component function of this task with different QoSs. QoS can give service providers a significant competitive advantage in the e-business domain, as QoS-aware services meet the user's needs better and thus attract more customers.

An execution path of composite service can be constructed by a sequence of tasks including an initial task and a terminal task. Each task contains many candidate services with the same functions but different QoSs. Thus, there are various service compositions for each execution path of composite service. Moreover, while the number of supplied candidate services is increasing, the service compositions will become larger and larger. A web service composition is illustrated in Figure 1.

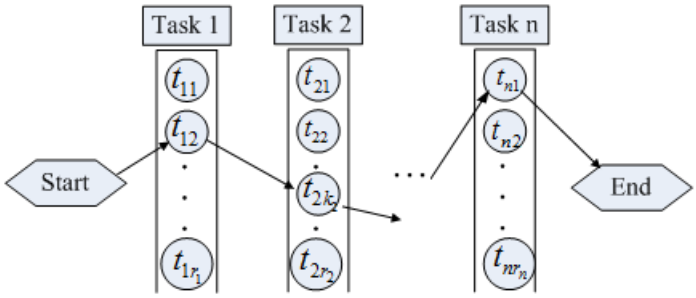


FIGURE 1. Web service composition illustration

As Figure 1 shows, there are n component tasks and each task has t_{ir_i} $(i = 1, \ldots, n)$ candidate services to construct a composite service to assist service consumers. Each candidate service from the candidate service list of a task can complete the corresponding task with different QoSs. What we should do is to choose a best combination of all the candidate services to construct a most satisfiable web service composition for the consumers that has best QoSs.

3.3. **QoS-aware web service selection model.** The goal of QoS-aware web service selection is to select a best web service combination from the lists of the candidate services so that the QoS performance of the constructed composite service can be maximized while satisfying the constraints of users. In order to evaluate the multi-dimensional performance of aggregated QoS attribute values, this paper applies multiple criteria decision making with a weighted sum model as a normalized (in $[0, 1]$) uniform [7] QoS performance evaluation model independent of their units.

The QoS-aware service selection is transformed into an optimization problem with constraints for all the composite web services (CWS), which is illustrated as Equation (3),

$$\begin{cases} \max \text{OBJ(CWS)} = \sum_{i=1}^{m} w_i Q_i \\ \text{s.t. } Q_i \le Q_i^0, \quad i = 1, \ldots, m \\ \qquad Q_{ij} \in R, \quad i = 1, \ldots, m, \ j = 1, \ldots, n \end{cases} \tag{3}$$

where m is the number of the non-functionality attributes (QoS), n is the number of tasks in a composite web service, w_i denotes the corresponding weight of the i-th QoS (Q_i) and $\sum_{i=1}^{m} w_i = 1$. Q_i (including *C, A, T, R* in this paper) is computed through Q_{ij} ($j = 1, \ldots, n$) according to Equations (4) and (5) respectively and Q_i^0 denotes the global constraints given by users. For a given composite web services, the one with higher score has a better QoS performance than others.

3.4. **Motivation of the research.** Research on Service Oriented Architecture and Service Oriented Computing brings a promising technique to create value-added business applications composed by dynamically selected individual services. The web services, a novel paradigm in software technology, have innovative mechanism for rendering services over diversified environment. They promise to allow businesses to adapt rapidly to changes in the business environment and the needs of different customers. The rapid introduction of new web services into a dynamic business environment can adversely affect the service quality and user satisfaction. Consequently, assessment of the quality of web services is of paramount importance in selecting a web service for an application.

Web service composition consists in combining web services, developed by different organizations and offering diverse functional (e.g., ticket purchase, payment), behavioral (e.g., compensative or not), and nonfunctional properties (Quality of Service values, e.g., execution price, success rate, delay time), to offer more complex services. One of the critical challenges [39] in reusing and integrating existing web service components is the efficient query, selection and composition of potentially relevant web service components based functional, operational and QoS requirements.

Our research objective is to provide service consumers with facilities for selecting required composite web services efficiently according to their Quality of Service (QoS). The aim of web service composition is to provide a reliable execution and an optimal QoS composite web service. Our contribution is a hybrid immune particle algorithm for web service composition and several heuristic search techniques, which guarantee that each component web service (WS) of a composite one is best. Furthermore, our approach is scalable because users only have to define their global transaction requirements and need not define the possible termination states of all the component web services.

4. **QoS Components for Web Service Selection Algorithm.** Since many available WSs provide overlapping or identical functionality, albeit with different QoSs, a choice needs to be made to determine which services will participate in a given composite service. The issue of selecting web services is significant for the purpose of their composition in a way that maximizes user's satisfaction expressed as utility functions over QoS attributes.

4.1. **QoS computation.** Composite WS has four basic structures: sequential (a), cycle (b), parallel (c) and branch structure (d) in Figure 2 [13]. In the sequential structure (a), tasks are executed in a sequential order; in the cycle structure (b), a task will be executed for multiple times; in the parallel structure (c), all the parallel tasks can be executed at the same time, but all the parallel tasks should be finished before going to the next

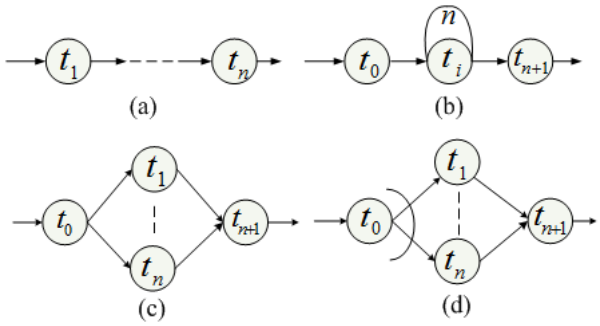


FIGURE 2. Four basic composite models of web service

task; in the branch structure (d), each task in the branch can finish the same component function and system can go to the next task once a task in the branch has been finished.

Since a composite WS is composed of the above four basic structures, QoS of a particular service composition can be calculated by computing the QoSs of its basic structures. Nonfunctional attributes of these structures can be calculated by the following formulas.

$$C = \begin{cases} \sum_{i=1}^{n} C_i, & (a) \\ \sum_{i=1}^{n} C_i, & (b) \\ \sum_{i=1}^{n} C_i, & (c) \\ \min(C_i), & (d) \end{cases} \qquad A = \begin{cases} \prod_{i=1}^{n} A_i, & (a) \\ \prod_{i=1}^{n} A_i, & (b) \\ \min(A_i), & (c) \\ \max(A_i), & (d) \end{cases} \tag{4}$$

$$T = \begin{cases} \sum_{i=1}^{n} T_i, & (a) \\ \sum_{i=1}^{n} T_i, & (b) \\ \max(T_i), & (c) \\ \min(T_i), & (d) \end{cases} \qquad R = \begin{cases} \prod_{i=1}^{n} R_i & (a) \\ \prod_{i=1}^{n} R_i, & (b) \\ \prod_{i=1}^{n} R_i, & (c) \\ \max(R_i), & (d) \end{cases} \tag{5}$$

4.2. **Mapping of solution vector to web service selection.** $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is the i-th antibody/position vector in the current solution/particle population. x_{ij} is an integer ranging from 0 to r_j, where r_j is the number of candidate services in the task j. $F(X_i)$ is the fitness of solution X_i. $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ represents the velocity vector of solution X_i. v_{ij} has the value of 0 or 1.

Solution Representing: An integer string is adopted to represent the candidate service and maps a solution vector to the specific web service composition on the basis of its logical structure of tasks. The solution vector is composed by a serial number of candidate service selected in the corresponding task. For example, a solution vector (2, 14, 3, 0, 2) means that there are five tasks in the logical structure of a composite service, where task one chooses the second candidate service, task two chooses the fourteenth candidate service. 0 indicates that this task is not included in this service composition.

Position Difference: $X_i - X_k$ means the position difference of solutions X_i and X_k, which is a velocity vector. The same corresponding components between vectors X_i and X_k deduce the result of 1. Otherwise, the result of 0 is deduced, e.g., $(4, 2, 1, 0, 5) - (2, 2, 1, 4, 5) = (0, 1, 1, 0, 1)$.

Weight: Weight means the probability of a solution/particle keeping its initial flying direction, to the local optimal solution/particle (P_i) and to the global best solution/particle (P_g) during the updating process. For example, term $p_1 V_{i,d}^k$ in Equation (6) means that it flies along its initial velocity with p_1 probability.

Velocity Summation: It is the summation of several velocity vectors, which denotes the final flying/searching direction of a solution as Equation (6) shows. Weight p_j means the probability of the particle to move at the direction of v_j and $\sum_{j=1}^{3} p_j = 1$. For example,

$0.6(1,0,0,1,1) + 0.3(1,1,0,1,0) + 0.1(1,1,0,0,0) = (1,*,0,*,*)$, the '$*$' in the second dimension of the vector means that the number here is 1 with a 0.6 probability and 0 with a 0.4 $(0.3 + 0.1)$ probability. The '$*$' in the fourth dimension of the vector means that the number here is 1 with a 0.9 probability and 0 with a 0.1 probability.

Summation between Position and Velocity ($\oplus$): Summation $\oplus$ indicates a new solution with the summation of the original position and the final flying/searching direction. Component 1 in the velocity vector means that the corresponding candidate service in the solution/position vector remains unchanged. Otherwise, the component choice will be modified by some strategies. For example, $(1,2,4,3,5) \oplus (1,0,1,0,1) = (1,*,4,*,5)$. The result vector $(1,*,4,*,5)$ means that the candidate services of the second and the fourth tasks will be altered with some methods. The velocity and position update equations for web service selection problem are illustrated as follows.

$$V_{i,d}^{(k+1)} = p_1 V_{i,d}^k + p_2 \left(P_{i,d}^k - X_{i,d}^k\right) + p_3 \left(P_{g,d}^k - X_{i,d}^k\right) \tag{6}$$

$$X_{i,d}^{k+1} = X_{i,d}^k \oplus V_{i,d}^{k+1} \tag{7}$$

4.3. **Clonal selection operation.** Antigen stands for the optimization problem and antibody represents the variable vector. Fitness of the t-th composite web service (CWS) is computed as follows.

$$\text{OBJ(CWS)} = \sum_{i=1}^{4}(\omega_i Q_i) = \omega_1 \sum_{j=1}^{n} C_j + \omega_2 \prod_{j=1}^{n} A_j + \omega_3 \sum_{j=1}^{n} T_j + \omega_4 \prod_{j=1}^{n} R_j \tag{8}$$

where Q_i $(i = 1,2,3,4)$ are the QoS attribute values (C, A, T, R) of composite service respectively; n is the task number; Q_{ij} $(i = 1,\ldots,4; j = 1,\ldots,n)$ represents the i-th QoS attribute value of candidate service in the task j; $\sum_{i=1}^{4} \omega_i = 1$ and $0 < \omega_i < 1$.

Antibodies with higher affinity are more eminent, therefore they should be proliferated more copies to sustain the superb properties of the antibody population. The number of an antibody cloning is calculated by the following equation.

$$num_t = \sigma P \times \left[\frac{affinity_t}{\sum_{t=1}^{P} affinity_t}\right] \tag{9}$$

where σ is a constant that an antibody population is multiple cloned and $affinity_t = \text{OBJ}_t \cdot [\rho_t + \delta]^{-1}$. ρ_t is the *Euclidean* distance between the t-th composite service and the optimal service scheme found so far, δ is a positive non-zero constant.

Hypermutation is the primary operator for the global exploration and local search around the eminent antibodies. Random mutation is adopted for the component task with a candidate service in this paper for elitism and efficiency. First, σP antibodies are cloned with Equation (9) on the basis of the present antibody population. Secondly, hypermutation operator is applied to each component task of the cloned antibodies with a certain mutation probability. A new candidate service is randomly selected from its candidate service list to substitute the current candidate service of the composite task.

4.4. **CSPLPT algorithm.**

Step 1. Initialize solutions swarm, velocity and parameters;
Step 2. Evaluate the fitness of all solutions;
Step 3. Calculate the affinity of all solutions;
Step 4. Apply clonal proliferation to current solution population with Equation (9);
Step 5. LAPS for clonal selection solution population;
Step 6. EPGS for the current solution population;
Step 7. IF current iteration number is less than the maximal iteration number $\times S\_switch\%$

THEN clonal selection operation is executed;
ELSE global tournament selection operation is used;
Step 8. Apply the elitist perturbation guiding strategy;
Step 9. If termination condition reaches, output the final results; else, goto Step 2.

5. **Hybrid Heuristics and Verification for Web Service.** The improved heuristic strategies and the hybrid algorithm are proposed in this section and the validity and necessity to propose them are also verified step by step and one by one.

5.1. **Experimental setup.** This paper did not pay much attention to considering these parameters. The experimental parameters are given as follows according to our comparative simulation experiences and the related [13, 30]. Most parameters are decided after some comparative experiments with the web service instance with 40 tasks and 15 candidate services, which include population size, maximal generation number, mutation rate, tournament size and the switch point *S_switch* of different selection operation. They are not listed here one by one due to the space limitation.

Population size $P = 10$ and the maximal generation number is 200. The multiple clone size is two, i.e., $\sigma = 2$. The elitist perturbation probability is 0.2 and hypermutation probability is 0.05. The point of switching clonal selection operation to global tournament selection operation *S_switch* is 40% and tournament size is 4. Four QoS attributes are indiscriminating in this paper. That is, the weights of four QoS attributes (C, T, A, R) are uniformly set 0.25 respectively. All the results are obtained from 30 independent runs and the same initial population is used to avoid the impact of non-algorithm factor.

Average evolutionary behaviors of different algorithms are considered to prove the necessity and the improvement of every strategy. The average best fitness values per iteration over 30 runs versus iteration number are plotted to show the evolution of mean best fitness values found by the pre-improved and post-improved algorithms in the following subsections. The data are sampled every ten iterations.

All the experiments of this paper are made on a *GATEWAY T6832c* Notebook with *Intel Duo2 T5750, 2G RAM* and *Mathematical 7.0* on *Windows XP* operation system.

5.2. **Clonal selection algorithm hybridized with PSO.** Two different population search mechanisms are hybridized to solve the web service selection problem in this subsection. The clonal selection immune algorithm focuses on the local search around the neighborhood of the current preferable solutions. However, there are no information exchanging among solutions. PSO is a population-based elitist learning mechanism from the social experience and the search history of itself. Accordingly, PSO has abundant information exchanging among solutions. However, the neighboring information of the better but not the best solutions are not fully exploited. These observations are the motivations to research the hybrid mechanisms between them and several heuristics.

The hybrid algorithm between clonal selection algorithm (CSA) and PSO is denoted as CSP. A moderate scale composite web service, which has 40 tasks and each task has 15 candidate service, is used to show the feasibility, necessity and the superiority of hybridizing CSA and PSO. The average best fitness values per iteration over 30 runs versus iteration number for three algorithms are plotted as Figure 3.

Observed from the average evolutionary comparison of Figure 3 it can be seen that hybrid CSP algorithm outperforms CSA and PSO greatly, in which our suppositions are verified. CSP outperforms CSA and PSO at about the fifth iteration although they begin to search with the same initial solution population. It also can be seen that CSA preponderates over PSO due to its population-based local search around the excellent solutions. However, the average evolutionary behaviors of CSP and PSO indicate that they still have

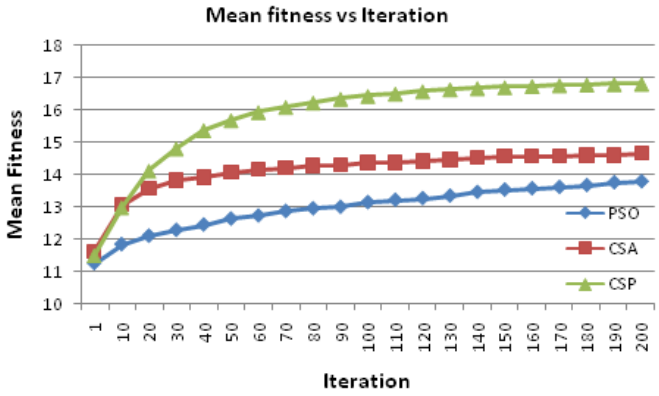


FIGURE 3. Comparison between CSA, PSO and CSP

potentials for even better QoSs for composite WSs selection. The evolutionary stagnation of CSA appears after about 100 iterations.

These experiments demonstrate that the average evolutionary behavior of CSP is much better than those of CSA and PSO. It also implies that two different population-based search mechanisms of CSA and PSO benefit each other for web service selection problem.

5.3. **Local adaptation preemptive strategy hybridized with CSP.** The excellent performance of CSP has been proved for web service selection and composition. Local adaptation preemptive strategy (LAPS) is proposed to further improve the search guideline for the candidate service selection and this hybrid algorithm is denoted as CSPL.

As we know, the composite WS is possible to have high QoS scores if every component task has excellent QoS performance. The population will have more powerful evolutionary impetus for high fitness values if most particles have better QoS scores. LAPS means that the larger QoS score of a candidate service has, the higher probability to be selected is for the component task. This will promote the convergence speed of the population evolution for CSP algorithm at a local search level. The probabilities of candidate services to be selected to fulfill the component tasks are calculated as the roulette selection operation of genetic algorithms [29].

Figure 4 compares the evolutionary behaviors on the average best fitness per iteration over 30 independent runs versus iteration number for CSP with and without LAPS, which uses a WS selection problem with 40 tasks and 15 candidate services for each task. The performance of CSP is enhanced by LAPS as Figure 4 indicates. Two plotted curves seem to be overlapped at the latter stage of algorithms. Figure 4(b) is plotted from the partial data of the last quarter process of two algorithms in a finer quantity scale. It tells us that CSPL outperforms CSP a little when solving web service selection problem.

5.4. **Elitist perturbation guiding strategy hybridized with CSPL.** The population diversity is possible to be affected with the proposition and hybridization of two different population search mechanisms and LAPS. An elitist perturbation guiding search strategy (EPGS), inspired by the perturbed particle swarm algorithm for numerical optimization [40], is presented to overcome the improper balance between local exploitation and global exploration and this hybrid algorithm is denoted as CSPLP.

It is known that the global best solution has great influence on the performance of PSO because velocities of all other particles are guided by it. So the swarm evolution is possible to be stagnated if the global best solution is trapped by a local optimum or it executes immoderate imposes on other solutions. This is the instinctive reason to propose the elitist perturbation guiding strategy.

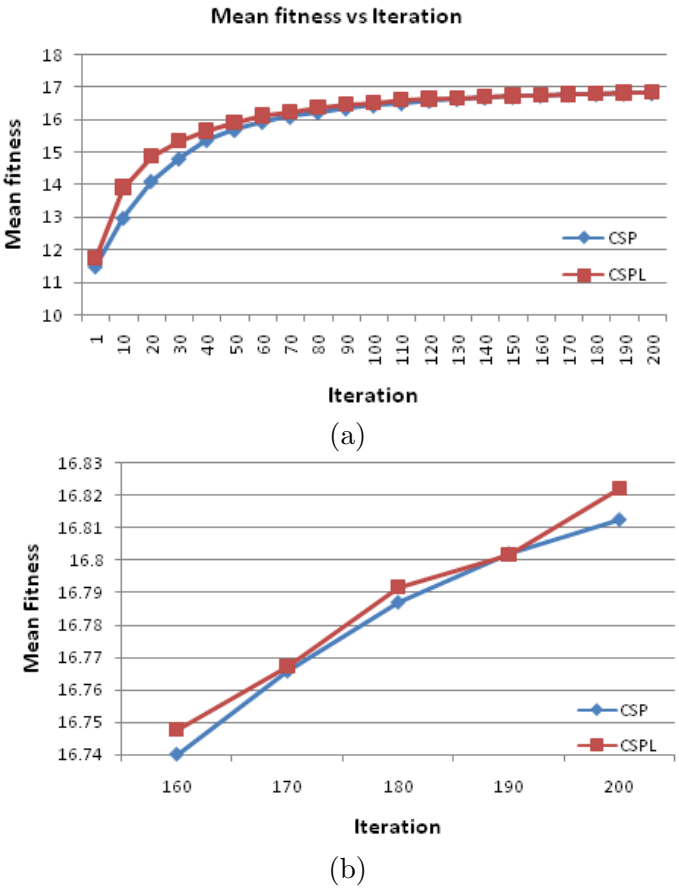
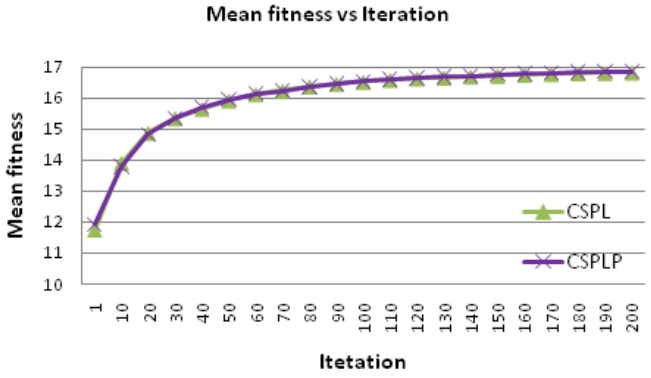


FIGURE 4. Comparison between CSP and CSPL

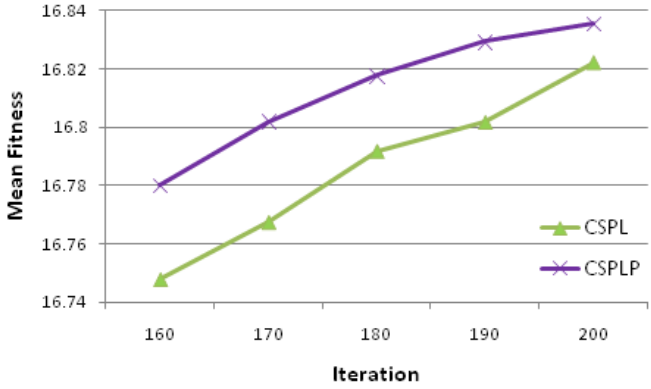
The comparison is presented in Figure 5 for the average best fitness values per iteration over 30 runs versus iteration number. The performance of CSPLP is enhanced by elitist perturbation guiding search strategy as Figure 5 indicates. The curves of Figure 5(b) are plotted from the partial data of the last quarter process of two algorithms in a finer quantity scale. It can be seen that the performance of CSPLP is enhanced again on the basis of CSPL for web service selection.

5.5. **Global tournament strategy hybridized with CSPLP.** What clonal selection operation and LAPS emphasize are the local search behaviors. The influence of global optimal solution is possible to emphasize immoderately to some extent when velocities and positions are updated in the population. That is, the above operations and strategies are more like to be looked on as local search behaviors or evolution orientation. This is the inherent reason to propose global tournament selection (GTS) operation at the latter stage of algorithm and this final algorithm is denoted as CSPLPT. It is possible to rectify the possible improper search directions for algorithm from a macroscopic perspective. As Figure 6 shows that this consideration is reasonable and effective for algorithm to solve the web service selection problem.

At the latter stage of algorithm, global tournament operation is executed as follows. Four solutions or WS composition schemes are chosen randomly from the current population and the best one is putted into the next population and this operation repeats until the maximal population size is reached. The convincing improved performance of CSPLPT supports our analysis forcefully as Figure 6 shows.



(a)



(b)

FIGURE 5. Comparison between CSPL and CSPLP

TABLE 1. Comparison among six algorithms for composite web service selection

Items	PSO	CSA	CSP	CSPL	CSPLP	CSPLPT
Best	14.4799	15.0101	16.93	16.935	16.935	16.935
Mean Best	13.7831	14.6504	16.8126	16.8221	16.8354	16.8796
STD	1.13e-1	4.01e-2	6.04e-3	4.67e-3	8.69e-3	1.05e-3
Time	6.05	5.21	14.29	16.40	17.18	16.98
Iteration	170	173	198	179	182	176

5.6. **Overall evolutionary performance comparison for the gradually improved algorithms.** The above experiments and analysis demonstrate the underlying necessity and excellence of hybridizing CSA, PSO and other heuristics, including local adaptation preemptive strategy, elitist perturbation guiding strategy and global tournament strategy. Performance comparison among these six algorithms are plotted as Figure 7 with their average best fitness per iteration over 30 independent runs versus iteration numbers. The final statistic data of 30 independent runs are also given in Table 1 besides the performance comparison plots when these algorithms are used to solve the composite web service selection.

Figure 7 and Figure 3 clearly illuminate the necessity and superiority of hybridizing CSA and PSO. Together with Figures 4-6, Figure 7 also testifies the benefits of the proposed strategies for the web service selection respectively. The curves of Figure 7(b) are finer displaying results in the "Mean fitness" from Figure 7(a).

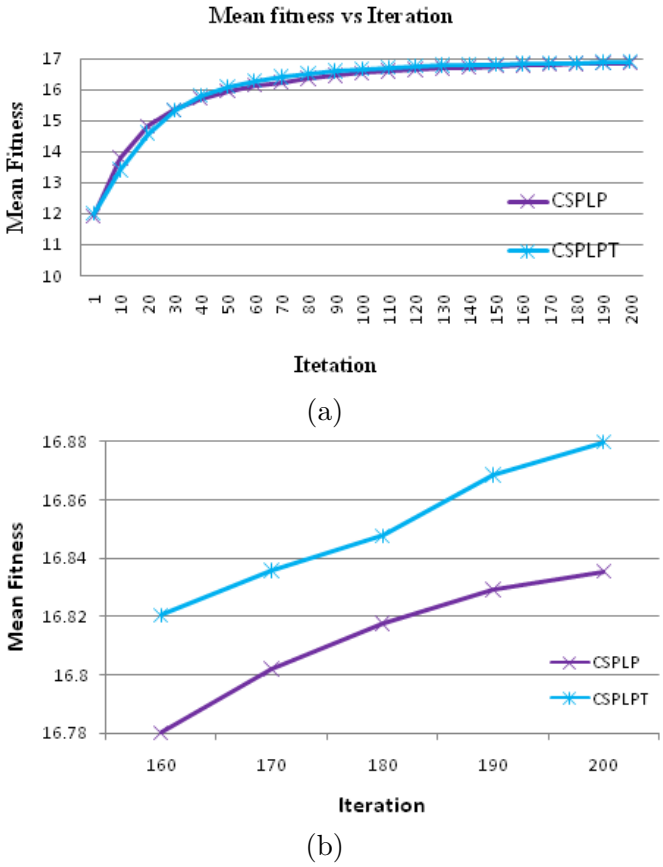
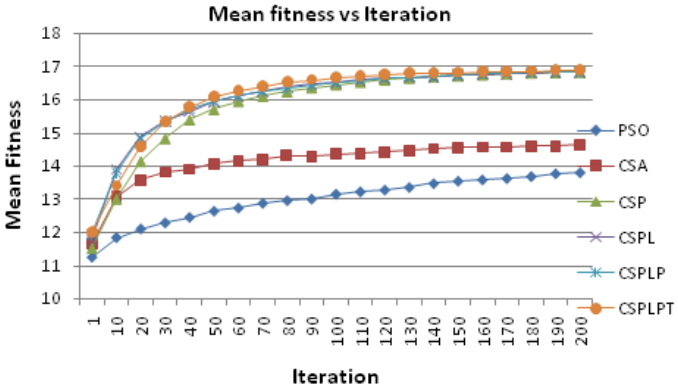


FIGURE 6. Comparison between CSPLP and CSPLPT

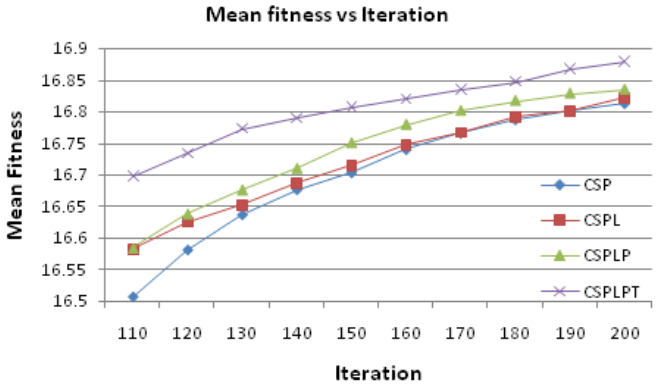
Items of "Best", "Mean Best" and "STD" in Table 1 are the best, average results and standard deviation of the optimal results in 30 independent runs. "Time" is the average convergent time (second) and "Iteration" is the average iteration numbers when the best results are obtained. The data of the second to the fourth rows of Table 1 clearly display the gradually enhancing performance of algorithms with the hybridization of different strategies, which support the above observation and analysis. By the way, the costs of the hybrid algorithms are about triple to the standard algorithms CSA and PSO when better results are obtained.

6. **Performance Comparison with Recent Swarm Algorithms.** The proposed strategies and their benefits to algorithm have been analyzed and verified step by step. More extensive experiments and analysis will be presented in this section to compare with a genetic algorithm variant (CoDiGA) [30] and an improved particle swarm optimization (iPSOA) [13]. Various scales of instances are used to compare them so as to understand and verify the features of the proposed strategies more fully and comprehensively.

6.1. **Compared algorithms and algorithmic setup.** CoDiGA [30] is characterized by the special relation matrix coding scheme of chromosomes, the population competition mechanism, the enhanced initial population policy and the evolution policy. The simulation results on web service selection with global QoS constraints have shown that prematurity was overcome effectively, and convergence and stability of genetic algorithm were improved greatly. The iPSOA [13] is proposed to solve the web service selection problem with some beneficial strategies, such as adaptive weight adjustment and non-uniform mutation strategies.



(a)



(b)

FIGURE 7. Comparisons among six algorithms with 40 tasks

TABLE 2. Comparison among CSPLPT, CoDiGA and iPSOA

	CSPLPT				**CoDiGA**				**iPSOA**			
Items	40	60	80	100	40	60	80	100	40	60	80	100
Best	16.585	24.594	33.148	39.390	14.204	20.447	27.111	32.4788	15.136	21.801	28.859	34.472
Mean	16.540	23.287	30.655	36.941	13.788	19.918	26.155	31.543	14.441	20.915	27.545	33.470
STD	1.36e-3	1.07	1.72	2.59	0.075	0.123	0.173	0.186	0.149	0.137	0.463	0.244
Time	14.03	15.25	23.37	26.73	5.42	6.38	12.20	10.89	8.45	10.78	21.35	22.26

The parameters of CSPLPT are the same as the above experiments except for the instance scale. The parameters of CoDiGA and iPSOA are set according to the suggestions of the authors except for the population size (10), the maximal iteration number (200) and the independent run times (30) as this paper. The maximal mutation rate is 0.5 and the systemic parameter b of non-uniform mutation is 5 for iPSO. The crossover and mutation probabilities of CoDiGA are 0.7 and 0.1 respectively. The details of CoDiGA and iPSOA can be found in [13, 30]. Even larger scales of web service instances with 40, 60, 80 and 100 tasks are adopted and every task has 15 candidate services.

6.2. **Performance comparison and algorithmic analysis.** The statistic results from 30 independent runs for comparison among CSPLPT, CoDiGA and iPSOA on various scales of instances are presented in Table 2 and Figure 8. The items of "Best", "Mean" and "STD" are the maximal, average and the standard deviation of the final results over 30 runs. The "Time" is the average CPU time (s) when the optimal results are reached.

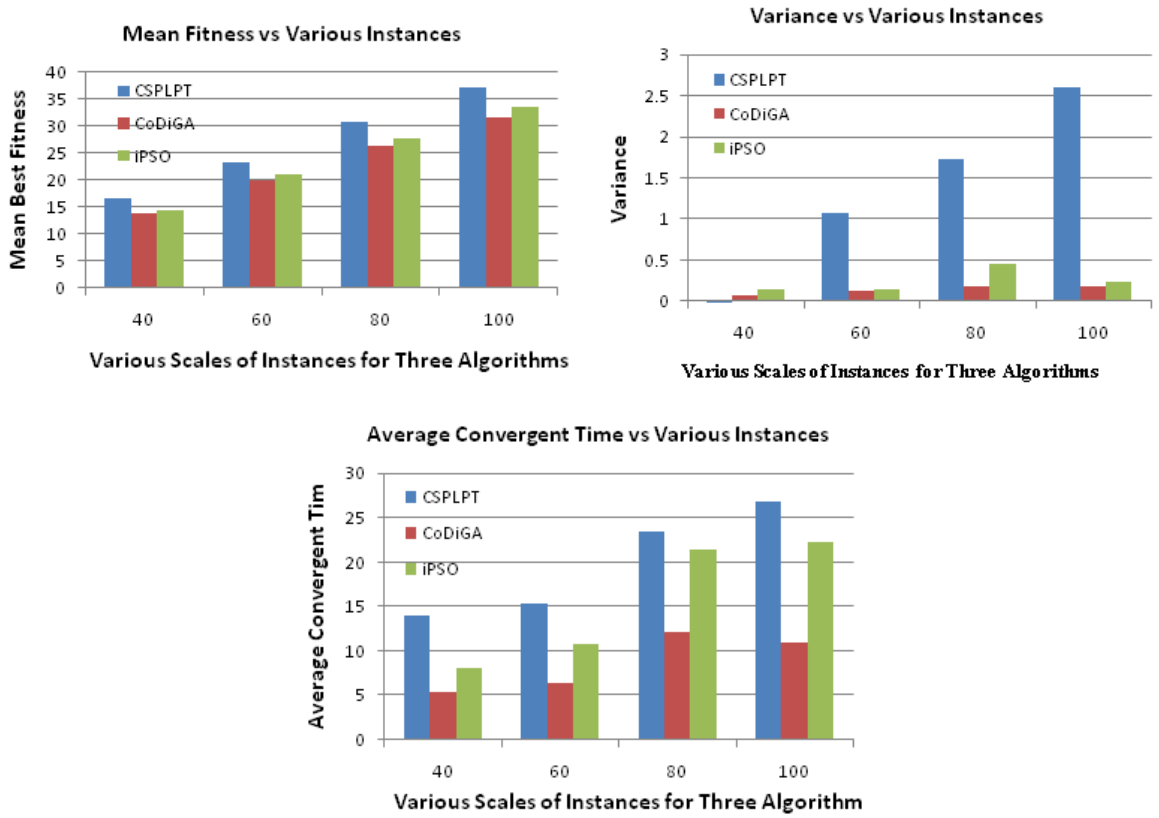


FIGURE 8. Mean fitness, STD and the average CPU time (s) comparisons among CSPLPT, CoDiGA and iPSOA

Observed from Table 2 we can find that the "Best" and "Mean" items of CSPLPT are all much better than those of CoDiGA and iPSOA, which demonstrates the even more powerful search capability of CSPLPT. Larger "STD" items of CSPLPT indicate the obvious influence of the random initial population on CSPLPT. However, the average CPU time, i.e., the efficiency of CSPLPT is worse than those of CoDiGA and iPSOA. That is, algorithm CSPLPT obtained much better WS selection schemes with the costs of computing efficiency.

The mean fitness values and the average convergent times are plotted in Figure 8 for an intuitive comparison among CSPLPT, CoDiGA and iPSOA algorithms. Figure 8 indicates that CSPLPT outperforms the other two algorithms and CoDiGA has worst performance in terms of statistical views although it has an enhanced initialization policy. The largest variance of CSPLPT demonstrates that its performance is different from the different initial populations and both CoDiGA and iPSOA are easily trapped by the local optima. Together with their performance, the facts that the average convergent time of CSPLPT is highest and CoDiGA is lowest indicate that CSPLPT has the most powerful search capability and CoDiGA is most possible to be trapped into local optima. The performance of iPSOA is in-betweens.

6.3. **Average evolutionary behavior comparison among CSPLPT, CoDiGA and iPSOA.** The dynamic evolutionary behaviors of three algorithms are also considered and compared each other besides the final static computational performance. The evolutionary behaviors of the average best fitness values per iteration over 30 runs versus iteration number demonstrate the progress of the statistical behaviors of algorithms CSPLPT,

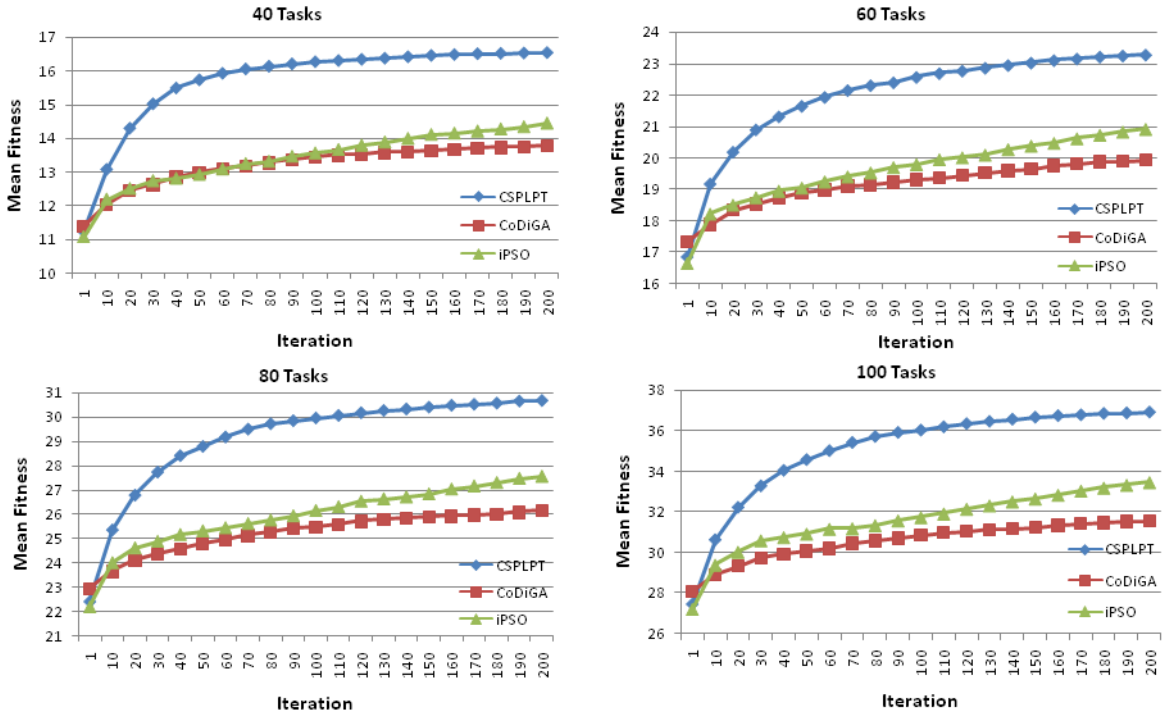


FIGURE 9. Average evolutionary behaviors comparison of CSPLPT, CoDiGA and iPSOA for various scales of WSs

CoDiGA and iPSOA (Figure 9). In my opinion the average evolutionary behaviors of algorithms have more persuasions than the final static results for its statistical evolutionary performance comparison from large sampling in the whole process of algorithms.

Observed from Figure 9 it is easy to see that the average statistical evolutionary behaviors of three algorithms have obvious difference over 30 independent runs. In other words, the performance of algorithm CSPLPT has distinct superiority to CoDiGA and iPSOA. As the same as Table 2 shows, the average evolutionary behaviors of CSPLPT in 30 runs are great better than the performance of CoDiGA and iPSOA and iPSOA outperforms CoDiGA for all the instances. It can be seen that the average fitness of CoDiGA is better than CSPLPT and iPSOA at the initial stage of algorithms which should attribute to its enhanced initial population policy [30]. However, CoDiGA always gets behind in the performance improvement for all the benchmarks. Furthermore, it is obvious that the statistical evolutionary performance of CSPLPT and iPOSA have potential to be further improved if the iteration numbers were increased as Figure 9 shows.

7. **Conclusions.** Web service selection with global QoS constraints is an active research area. It is very important to select which services to be used in a composite web service according to requester's QoS requirements. Based on the features of web service selection with global QoS constraints, a novel hybrid swarm intelligence algorithm (CSPLPT) is proposed for this problem. The main contributions are summarized as follows.

(1) Both clonal selection algorithm and particle swarm algorithm are population-based heuristic algorithms with different search mechanisms. CSA focuses on population-based local search around the neighborhood of the current solutions, which have no communication between each other. PSO is a population-based elitist learning mechanism, which has abundant information exchanging between solutions. However, the neighboring information of the better but not the best solutions are not fully exploited. Hence the proposed algorithm CSP and herein CSPLPT are population-based local search algorithm with

elitist learning. That is, CSP and the corresponding CSPLPT have common advantages of local search mechanisms of CSA and PSO.

(2) Several other heuristic techniques are also proposed to pay more attentions either to convergence or to diversification. These heuristics are all proposed to compensate the possible disadvantages based on analysis. Experiments are conducted to verify the necessity and effectiveness of the proposed heuristics.

The proposed algorithm CSPLPT is originated for web service selection problem and it is also possible for a service-oriented environment. Two recent swarm algorithms, a genetic algorithm variant CoDiGA and a PSO variant iPSOA, are used to validate the performance of CSPLPT with various scales of composite web service benchmarks. Experimental comparisons indicate that CSPLPT performs much better than CoDiGA and iPSOA in terms of QoS performance for web service selection.

Acknowledgment. This work is partially supported by National Natural Science Foundation of China (61105127, 11171040). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] C. Petrie and C. Bussler, Service agents and virtual enterprises: A survey, *IEEE Internet Computing*, vol.7, no.4, pp.68-78, 2003.

[2] S.-Y. Hwang, E.-P. Lim, C.-H. Lee and C.-H. Chen, Dynamic web service selection for reliable web service composition, *IEEE Transactions on Services Computing*, vol.1, no.2, pp.104-116, 2008.

[3] L. J. Zhang, Editorial: Modern services engineering, *IEEE Transactions on Services Computing*, vol.2, no.4, p.276, 2009.

[4] W.-L. Lin, C.-C. Lo, K.-M. Chao and M. Younas, Consumer-centric QoS-aware selection of web services, *Journal of Computer and System Sciences*, vol.74, pp.211-231, 2008.

[5] L. F. Ai, M. L. Tang and C. Fidge, Partitioning composite web services for decentralized execution using a genetic algorithm, *Future Generation Computer Systems*, vol.27, pp.157-172, 2011.

[6] N. C. Mendonca, J. A. F. Silva and R. O. Anido, Client-side selection of replicated web services: An empirical assessment, *The Journal of Systems and Software*, vol.81, pp.1346-1363, 2008.

[7] L. Z. Zeng, B. Benatallah, A. H. H. Ngu et al., QoS-aware middleware for web services composition, *IEEE Transactions on Software Engineering*, vol.30, no.5, pp.311-327, 2004.

[8] P. Wang, K.-M. Chao and C.-C. Lo, On optimal decision for QoS-aware composite service selection, *Expert Systems with Applications*, vol.37, pp.440-449, 2010.

[9] Y.-S. Luo, Y. Qi, D. Hou et al., A novel heuristic algorithm for QoS-aware end-to-end service composition, *Computer Communications*, vol.34, no.9, pp.1137-1144, 2011.

[10] R. Mohanty, V. Ravi and M. R. Patra, Web-services classification using intelligent techniques, *Expert Systems with Applications*, vol.37, pp.5484-5490, 2010.

[11] W.-L. Lin, C.-C. Lo, K.-M. Chao and N. Godwin, Multi-group QoS consensus for web services, *Journal of Computer and System Sciences*, vol.77, no.2, pp.223-243, 2011.

[12] D. A. Menascé, E. Casalicchio and V. Dubey, On optimal service selection in service oriented architectures, *Performance Evaluation*, vol.67, pp.659-675, 2010.

[13] W. B. Wang, Q. B. Sun, F. C. Yang and X. C. Zhao, An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication, *International Journal of Computational Intelligence Systems*, vol.4(s), pp.18-30, 2010.

[14] W. J. Niu, G. Li, Z. J. Zhao, H. Tang and Z. Z. Shi, Multi-granularity context model for dynamic web service composition, *Journal of Network and Computer Applications*, vol.34, pp.312-326, 2011.

[15] C. C. Chang and L.-W. Sung, A computer maintenance expert system based on web services, *ICIC Express Letters*, vol.3, no.4(B), pp.1209-1214, 2009.

[16] A. F. M. Huang, C.-W. Lan and S. J. H. Yang, An optimal QoS-based web service selection scheme, *Information Sciences*, vol.179, pp.3309-3322, 2009.

[17] S. G. Wang, Q. B. Sun and F. C. Yang, Towards web service selection based on QoS estimation, *Int. J. Web and Grid Services*, vol.6, no.4, pp.424-443, 2010.

[18] L. N. De Castro and J. I. Timmis, *Artificial Immune Systems: A New Computational Intelligence Paradigm*, Springer-Verlag, 2002.

[19] T. Burczynski, Information sciences special issue on artificial immune systems, *Information Sciences*, vol.179, pp.1377-1378, 2009.
[20] X. B. Cao, H. Qiao and Y. W. Xu, Negative selection based immune optimization, *Advances in Engineering Software*, vol.38, pp.649-656, 2007.
[21] D. Dasgupta and L. F. Niño, *Immunological Computation: Theory and Applications*, CRC Press, 2008.
[22] E. Hart and J. Timmis, Application areas of AIS: The past, the present and the future, *Applied Soft Computing*, vol.8, no.1, pp.191-201, 2008.
[23] L. N. de Castro and F. J. von Zuben, Learning and optimization using the clonal selection principle, *IEEE Trans. Evolutionary Computation*, vol.6, no.3, pp.239-251, 2002.
[24] M. G. Gong, L. C. Jiao and L. N. Zhang, Baldwinian learning in clonal selection algorithm for optimization, *Information Sciences*, vol.180, pp.1218-1236, 2010.
[25] J. Y. Chen, Q. Z. Lin and Z. Ji, A hybrid immune multiobjective optimization algorithm, *European Journal of Operational Research*, vol.204, pp.294-302, 2010.
[26] Q. Li, L. Y. Sun and L. Bao, Enhanced index tracking based on multi-objective immune algorithm, *Expert Systems with Applications*, vol.38, pp.6101-6106, 2011.
[27] K. Trojanowski and S. T. Wierzchon, Immune-based algorithms for dynamic optimization, *Information Sciences*, vol.179, pp.1495-1515, 2009.
[28] Z.-H. Zhan, J. Zhang, Y. Li et al., Adaptive particle swarm optimization, *IEEE Trans. Syst., Man, Cybern. B, Cybern*, vol.39, no.6, pp.1362-1381, 2009.
[29] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
[30] Y. Ma and C. W. Zhang, Quick convergence of genetic algorithm for QoS-driven web service selection, *Computer Networks*, vol.52, no.5, pp.1093-1104, 2008.
[31] F. Jay and R. Mayer, IEEE standard glossary of software engineering terminology, *IEEE Std 610.12-1990*, 1990.
[32] *Oxford Dictionaries*, Oxford University Press, http://www.askoxford.com/?view=uk.
[33] J. O'sullivan, D. Edmond and A. T. Hofstede, What's in a service? Towards accurate description of non-functional service properties, *Kluwer Academic Publishers Distributed and Parallel Databases*, vol.12, pp.117-133, 2002.
[34] H.-C. Wang, C.-S. Lee and T.-H. Ho, Combining subjective and objective QoS factors for personalized web service selection, *Expert Systems with Applications*, vol.32, pp.571-584, 2007.
[35] D. Skoutas, D. Sacharidis, A. Simitsis and T. Sellis, Ranking and clustering web services using multicriteria dominance relationships, *IEEE Transactions on Services Computing*, vol.3, no.3, pp.163-177, 2010.
[36] J. Kennedy, R. C. Eberhart and Y. H. Shi, *Swarm Intelligence*, Morgan Kaufmann, San Mateo, CA, 2001.
[37] Y. Shi and R. C. Eberhart, Parameter selection in particle swarm optimization, *Evolutionary Programming VII: The 7th International Conference*, pp.591-600, San Diego, CA, USA, 1998.
[38] C. S. Park and S. Park, Efficient execution of composite web services exchanging intensional data, *Information Sciences*, vol.178, no.2, pp.317-339, 2008.
[39] A. Liu, L. Huang and Q. Li, QoS-aware web services composition using transactional composition operator, *Proc. of Int. Conf. Advances in Web-Age Information Management*, pp.217-228, 2006.
[40] X. C. Zhao, A perturbed particle swarm algorithm for numerical optimization, *Applied Soft Computing*, vol.10, no.1, pp.119-124, 2010.