Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 54(4), 438–447(2013)

Jin Liu, Yuxi Chen, Xu Chen, Jianli Ding, Kaushik Roy Chowdhury, Qiping Hu, Shenling Wang

A Cooperative Evolution for QoS-driven IoT Service Composition

DOI 10.7305/automatika.54-4.417
UDK 004.738.5; 519.254
IFAC 1.1.9; 2.8.4; 4.0.5

Original scientific paper

To facilitate the automation process in the Internet of Things, the research issue of distinguishing prospective services out of many “similar” services, and identifying needed services w.r.t the criteria of Quality of Service (QoS), becomes very important. To address this aim, we propose heuristic optimization, as a robust and efficient approach for solving complex real world problems. Accordingly, this paper devises a cooperative evolution approach for service composition under the restrictions of QoS. A series of effective strategies are presented for this problem, which include an enhanced local best first strategy and a global best strategy that introduces perturbations. Simulation traces collected from real measurements are used for evaluating the proposed algorithms under different service composition scales that indicate that the proposed cooperative evolution approach conducts highly efficient search with stability and rapid convergence. The proposed algorithm also makes a well-designed trade-off between the population diversity and the selection pressure when the service compositions occur on a large scale.

Key words: Cooperative evolution, IOT service composition, Quality of service

Kooperativna evolucija za kvalitetno pružanje usluga u paradigmi Interneta stvari. Kako bi se automatizirali procesi u internetu stvati, nužno je rezlikovati bitne usluge u moru sličnih kao i identificirati potrebne usluge u pogledu kvalitete usluge (QoS). Kako bi doskočili ovome problemu prdlaže se heuristička optimizacija kao robustan i efikasan način rješavajne kompleksnih problema. Nadalje, u članku je predložen postupak kooperativne evolucije za slaganje usluga uz ograničenja u pogledu kvalutete usluge. Predstavljen je niz efektivnih strategija za spomenuti problem uključujući strategije najboljeg prvog i najboljeg globalnog koje unose perturbacije u polazni problem. Simulacijski rezultati kao i stvarni podatci su korišteni u svrhu evaluacije prodloženog algoritma kako bi se osigurala efikasna pretraga uz stabilnost i brzu konvergenciju. Predloženi algoritam također vodi računa o odnosu između različitosti populacije i selekcijskog pritiska kada je potrebno osigurati slaganje usluga na velikoj skali.

Ključne riječi: kooperativna evolucija, Internet stvari pružanje usluge, kvaliteta usluge

1 INTRODUCTION

As network applications develop, the ability of identifying a physical or virtual entity that exists and moves in space and time makes the vision of the Internet of Things (IOT for short) possible, where the virtual world of information technology integrates seamlessly with the real world of things [1, 2]. In this sense, a "thing" is not restricted to material objects but can apply to virtual entities, activities and events that are connected to each other and identified either by identification numbers, names and/or location addresses [3]. The meaning of "thing" implies the ability of such intelligent subjects, devices or actors for "sensing" objects in the real world and operating on the data obtained from them without obstacles [4]. Identified "things" can be integrated into the dynamic relationship field of event processes and combined into a composite function system in accordance with the laws of its development.

IOT requires the characteristics of automation and intelligentialization to organize resources [1, 5]. Intelligentialization means the ability to understand, process information, learn and make decisions as people can do, so the existing resources can be identified and the relationships between them may be established. Software is the common fabric for achieving this automation and intelligentialization. Current software research focuses on service oriented computing (SOC) for developing a loosely organized and coherent application [6, 7]. Moreover, when "things" provided by a single service are not sufficient to satisfy the user’s requirement, the service composition (SC) techniques enable a combined application out of a large collection of individual but interlinked "thing" in the environment.

To facilitate the automation and adaption of SC, a key research issue is how to make IOT SC smart enough to distinguish prospective services out of many "similar" services and identify needed services that are essential to meet the constraints posed by the Quality of Service (QoS) requirements. Specifically related to SOC, the concept of QoS covers non-functional properties such as price, availability, reliability, and reputation [8]. These properties may be expressed either from a stand-alone Web service or a composed Web service. Furthermore, to embed intelligence in the IoT and achieve the application goals, the integration of atomic services into a combined one must occur, with the help of the SC following cooperation strategies and selecting the desirable services. A general purpose IoT may also involve a cooperative evolution of the operational strategy for optimal QoS based functioning of the network, which again brings changes in service selection and service composition.

In most cases, SC can be abstracted into a multi-objective optimization problem (MOOP) [9]. This is because non-functional criteria are presented in the QoS issue, and there is usually more than one criterion to be considered in a QoS model [10, 11]. Thus, the issue of cooperation evolution is also considered as a MOOP for QoS optimization to deal with uncertain changes in SC.

Recently, the research community has looked into the performance of evolutionary algorithms (EAs), for MOOPs have attracted significant attention within the EA community [12]. The challenge in this emerging area is mainly due to the reason that traditional EAs generally provide only one optimal solution. Similar to EAs, particle swarm optimization (PSO), inspired by the simulation of social behaviors of biological population such as fishes and birds, is also an iterative and population-based optimization technique [13]. As it is easy-to-implement and has robust adaptability, PSO is potential way to frame the QoS optimization w. r. t. cooperation evolution [14-21].

Following the research aims discussed above, this paper proposes a cooperative evolution algorithm based on PSO for QoS-driven web service composition. It takes several algorithm characteristics into consideration, such as (i) proportional evolutionary optimization, (ii) an improved local best first strategy for candidate service selection, (iii) a perturbing global best strategy along the global best particle, and (iv) a self-adaptive adjusting mechanism of learning rates. The benefits of these factors are as follows: The proportional evolutionary optimization ensures the diversity of population. The improved local best first strategy for the candidate service selection can lead to convergence in the global minima/maxima. The perturbing global best strategy along the global best particle can avoid trapping the result in a local optimization. The self-adaptive adjusting mechanism of learning rates makes a trade-off between the convergence speed and the final optimized result. We undertake a performance comparison with several well-known algorithms to illustrate the promising nature of our approach, and the fast convergence in reaching an optimized result.

The rest paper is organized as follows: Section 2 presents preliminary knowledge of the proposed work, i.e., QoS computation and the model of web service composition. Section 3 describes the proposed cooperative evolution approach on the basis of0 PSO for QoS-driven web service composition. Section 4 describes the performance evaluation of the proposed approach as opposed to two traditional optimization algorithms, i.e., the canonical PSO (CPSO) [21] and the improved discrete immune algorithm based on CPSO (IDIPSO) [22]. Finally, Section 5 concludes this paper.

2 PRIMITIVITY

2.1 QoS of Service Composition

The execution path on a service composition implements a task flow for application goals, where a task is the basic unit that a composite web service can use [23]. The task flow is essentially a task sequence with an initial task and a terminal task. Since a task flow can be implemented by different service compositions, it is necessary to find a desirable one from several potential candidates, according to the results of the QoS computation [10].

In general, the structure of Web service composition can be divided as four basic patterns: sequential, cycle, parallel and branch structure, as described in Fig. 1 (a), (b), (c) and (d) respectively [22].

1. For the sequential pattern, tasks are executed in a sequential order;
2. For the cycle pattern, at least one task should be executed more than once;
3. For the parallel pattern, the parallel tasks that were executed simultaneously go to the next task, until all of these parallel tasks are accomplished;
4. For the branch pattern, it selects only one task from a set of optional ones and goes to the next step.

The difference between the parallel and branch structure is very distinct. For example, all the tasks $t_1, \ldots, t_n$ have to be completed before the execution of task t_{n+1} in a parallel structure shown in figure 1 (c). In the case of figure 1 (d), a branch goes through only one of the tasks $t_1, \ldots$ and t_n from task t_0 to t_{n+1}.

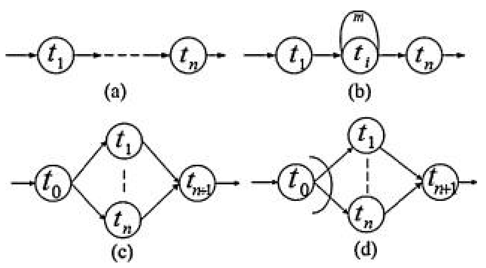


Fig. 1: Basic patterns of web service composition

The QoS of a specific service composition can be deduced from the QoS computation of the above mentioned basic patterns because the composite task is constructed in accordance with these basic patterns. Since it is impossible to calculate all non-functional attributes, some predominant attributes are selected to characterize the QoS. Without loss of generality, non-functional attributes Cost (*C*), Availability (*A*), Time (*T*) and Reliability (*R*) are taken into consideration in QoS computation. Accordingly, the basic pattern of service composition can be evaluated as formulae (1)-(4).

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

$$A = \begin{cases} \prod_{i=1}^{n} A_i, & (a) \\ \prod_{i=1}^{n} A_i, & (b) \\ \min(A_i), & (c) \\ \max(A_i) & (d) \end{cases} \tag{2}$$

$$T = \begin{cases} \sum_{i=1}^{n} T_i, & (a) \\ \sum_{i=1}^{n} T_i, & (b) \\ \max(T_i), & (c) \\ \min(T_i), & (d) \end{cases} \tag{3}$$

$$R = \begin{cases} \prod_{i=1}^{n} R_i, & (a) \\ \prod_{i=1}^{n} R_i, & (b) \\ \prod_{i=1}^{n} R_i, & (c) \\ \max(A_i) & (d) \end{cases} \tag{4}$$

The attribute values are normalized to improve the accuracy of QoS estimation, which may be influenced by various measurement metrics to these attributes. In these formulae, the maximal attribute values and the minimal attribute from qualified services are defined as$Q_i^{\max}$ and $Q_i^{\min}$, respectively. While Q_i is an attribute value of a service before the normalization, $\tilde{Q}_i$ refers to the normalized attribute value.

A numeric QoS attribute can take either a positive or a negative value. In the former case, taking attributes availability and reliability for example, a higher attribute value indicates better QoS quality, as described in (5).

$$\tilde{Q}_i = \begin{cases} \frac{Q_i - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1 & Q_i^{\max} - Q_i^{\min} = 0 \end{cases} \tag{5}$$

In the latter case, taking attributes cost and response time for example, a negative value exhibits the opposite effect on QoS, as described in the normalized formula (6).

$$\tilde{Q}_i = \begin{cases} \frac{Q_i^{\max} - Q_i}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1 & Q_i^{\max} - Q_i^{\min} = 0 \end{cases} \tag{6}$$

It is obvious that $\tilde{Q}_i \in [0,1]$ when the attributes of a service take either positive or negative values. For convenience, a normalized QoS value is denoted as *Qi* (not$\tilde{Q}_i$) in the rest of this paper.

2.2 Aggregate QoS-based Optimization of Service Composition

According to the types of QoS attributes, this paper designs a web service selection method that maximizes the QoS performance of the constructed composite service. Unlike Cost (C) and Time (T) that are normally expected to be minimal, availability (A) and Reliability (R) are expected to be maximal. Especially, C_0 and $T_=0$ are denoted as the maximal cost and time for users respectively. Thus, a four-dimensional objective optimization model of QoS can be represented by (7).

$$\begin{cases} F(\min C, \max A, \min T, \max R) \\ s.t. C \leq C_0, \quad T \leq T_0; \end{cases} \tag{7}$$

Further, the QoS performance of a potential service can be calculated uniformly by summing up the product of each normalized attribute value and its corresponding weight, as shown below. To evaluate the multi-dimensional performance of aggregated QoS attribute values, the multiple criteria is employed with a weighted sum model that is a uniform QoS performance evaluation model independent of attributes' units and ranges. Thus, the QoS-driven web service composition can be enhanced as a weighted summation optimization model within the constraints as (8).

$$\begin{cases} \max & F = fitness(WSC) = \sum_{i=1}^{n} \omega_i Q_i \\ & s.t. \quad Q_i \leq Q_i^0, i = 1, ..., n \end{cases} \tag{8}$$

where *n* is the number of the non-functionality attributes (QoS); *Qi* is the *i*-th QoS attribute values; ω_i is the corresponding weight of the *i*-th QoS (*Qi*); and $\sum_{i=1}^{n} \omega_i$=1, 0 <ω_i<1, *i* = 1, …, n. *WSC* is the abbreviation of the Web Service Composition. The global constraints given by users is denoted as Q_i^0.

By now the fitness of a web service composition is calculated according to formula (9).

$$F=\sum_{k=1}^{4}\omega_k Q_k=\omega_1\sum_{i=1}^{m}C_i+\omega_2\prod_{i=1}^{m}A_i+\omega_3\sum_{i=1}^{m}T_i+\omega_4\prod_{i=1}^{m}R_i, \tag{9}$$

where Qi $(k = 1, 2, 3, 4)$ are the QoS attribute values (C, A, T, R) of a service composition respectively; m is the task number; $Q_{ki}(k = 1, \dots, 4; i = 1, \dots, m)$ is the k-th QoS attribute value of a candidate service in the task i.

3 COOPERATIVE EVOLUTION FOR SC

3.1 Genetic Algorithm

Genetic algorithm (GA) is based on the idea of the "survival of the fittest in natural selection" first devised by Darwin. The GA algorithm simulates the biological genetic evolution process [21]. Specifically, GA consists of operations such as selection, reproduction, crossover, and mutation operators. Excellent parents in the previous generation pass on their genes to their offspring. In this way, GA can quickly find optimal solutions during the solution search process. GA evolves a population of candidate solutions. Each solution is usually coded as a binary string called a chromosome. The ?tness of each chromosome is then evaluated using a performance function after the chromosome has been decoded. Upon completion of the evaluation, a biased roulette wheel is used to randomly select pairs of better chromosomes to undergo such genetic operations as crossover and mutation that mimic nature. If the newly produced chromosomes turn out to be stronger than chromosomes from the previous generation, they will replace previous chromosomes. This evolution process continues until the stopping criteria are reached. The steps for applying GA to multi-objective optimization problem are as follows [12]:

i The process parameters are encoded as genes by binary encoding.

ii A set of genes is combined together to form a chromosome, which is used to perform those basic mechanisms in the GA, such as crossover and mutation.

iii Crossover is the operation to exchange some part of two chromosomes to generate new offspring, which is important when exploring the whole search space rapidly.

iv Mutation is applied after crossover to provide a small randomness to the new chromosome.

```
The GA algorithm (pseudocode)
Generate an initial population;
Evaluate fitness of individuals in the population;
DO
    Select parents from the population;
    Recombine (mate(crossover and mutation operations)) parents to
    produce children;
    Evaluate fitness of the children;
    Replace some or all of the population by the children;
    While a satisfactory solution has been found.
OD
```

Fig. 2: The primitive idea of GA algorithm

v To evaluate each individual or chromosome, the encoded process parameters are decoded from the chromosome and are used to predict machining performance measures.

vi The fitness or objective function is a function needed in the optimization process and the selection of the next generation in the GA.

vii After a number of iterations of the GA, optimal results of process parameters are obtained by comparison values of objective functions among all individuals.

Figure 2 illustrates the pseudocode of the GA algorithm.

3.2 Particle Swarm and Local Stochastic Strategy

Inspired by human decision-making behavior, Boyd and Richerson found that humans make decisions based on two kinds of information: their own experiences and other people's experiences [24, 25]. They proposed two different methods of individual learning and the transmission of culture, i.e., PSO. Moreover, Eberhart and Kennedy observed birds or fish looking for food and proposed a PSO to describe the behavior of a swarm of birds that did not initially know where to find food [13]. When an individual bird knows the direction for food, it transmits that information to its group. In this way, it corrects the direction of the other birds and allows them to advance toward the food.

Although the tranditioal PSO has already been applied successfully in many application areas [14-17], it makes slow progress in keeping the balance between exploration and exploitation in dynamic environment is slow [18-20]. This is because that the tranditioal PSO cannot adapt to the changing environment and converge to an optimum in an early iteration [14]. Moreover, the tranditioal PSO is also susceptible to becoming stuck at a local-optimal solution region [15]. Accordingly, an enhanced PSO (also known as canonical PSO, CPSO) with inertia weights was introduced by Shi and Eberhart, that can be represented with (10) and (11) [21]:

```
The CPSO algorithm (pseudocode)
1: Initialize a population array of particles with random
   positions and velocities on N dimensions in the search
   space.
2: loop
3:   For each particle, evaluate the desired optimization
     fitness function in N-dimensional space;
4:   Compare particle's fitness evaluation with its pBest. If
     current value is better than pBest, the set pBest equal
     to the current value, and if pBest is better than gBest,
     then set gBest equal to the current value in
     N-dimensional space.
5:   Change the velocity and position of the particle
     according to the equation (10) and (11);
6:   If a criterion is met (usually a sufficiently good fitness
     or a maximum number of iterations), exit loop.
8: End loop
```

Fig. 3: The primitive idea of CPSO

$$v_i^d(t+1) = \\ w \times v_i^d(t) + c_1 \times rand_1 \times (pBest_i^d(t) - x_i^d(t)) + \\ c_2 \times rand_2 \times (gBest_i^d(t) - x_i^d(t)) \tag{10}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1), 1 \leq i \leq n, 1 \leq d \leq N \tag{11}$$

where x is a volume-less particle (a point) in the N-dimensional search space; n is the total number of particles; $x_i = (x_i^1, ..., x_i^N)$ is the current position of the i-th particle; $v_i = (v_i^1, ..., v_i^N)$ is the velocity of the i-th particle; $pBest_i = (pBest_i^1, ..., pBest_i^N)$ is the best position of the i-th particle by the current iteraction; $gBest_i = (gBest_i^1, ..., gBest_i^N)$ is the best position of total particle swarm. $rand_1$ and $rand_2$ are the random numbers uniformly distributed in $(0, 1)$; c_1 and c_2 are the acceleration coefficients that determine the relative cognition and social influence on the particle's velocity. The parameter w gradually reduces as the nubmber of the generation interation increases. In addition, V_{max} is employed to place a limit on the particle's velocity. To be specific, a triple-valued signum function $sign(x)$ will be applied to $|V_i^d|$ in terms of $V_i^d = sign(V_i^d) * V_{\max}^d$, if $|V_i^d| > V_{\max}^d$. The formula (10) and formula (11) simulate how an individual is influenced by others from its society. The pseudo-code for particle swarm optimization algorithm is illustrated in Fig. 3. By regulating the parameters w and V_{max}, CPSO can demonstrate its ability in goal search [14].

3.3 Cooperative Evolution

PSO and GA have multiple points of similarities and differences. They are both commonly used in optimization strategies based on random searches, the random production of initial solutions and use of fitness values as evaluation indicators [26]. The distinction between PSO and GA is that PSO lacks crossover and mutation functions, but GA shares information through chromosomes. PSO can memorize the global best result at runtime and affects the movement of other particles easily to get a quick convergence and ensure that the optimization falls into at least a local optimal solution. This paper combines the strengths of both from PSO and GA by integrating these two algorithms.

The proposed cooperative evolution algorithm for QoS-driven web service composition is initialized with a population of random solutions and then it searches for optima by traveling the search space. During this travel, an evolution of this solution is performed by integrating PSO and GA. The cooperative evolution algorithm can be defined as the following steps.

Step 1. Initialization It initializes a population of particles with random positions and velocities in the *n*-dimensional SC problem space.

Step 2. Evaluation It evaluates the desired optimization fitness function in *n* variables for each particle.

Step 3. Setting pBest and gBest It sets *pBest* and the objective value for each particle, which is equal to its current position and objective value. It then sets *gBest* and the objective value for each particle, which is equal to its position and objective value of the best initial particle.

Step 4. Updating the velocity and position It updates the velocity and position of each particle according to (10) and (11).

Step 5. Evolution of particles Moreover, to keep the feasibility of the particles, It adopts a constriction factor χ *to* restrict the velocity of particle evolution with a similar manner in [14]. The new modified factor χ can be defined as in (12).

$$\chi = \frac{2}{|-2 - \tau - \sqrt{\tau^2 + \tau}|}, \tag{12}$$

where τ is the age of the infeasible particle. And the new modified position of the particle is deduced according to formula (13).

$$x_i^{k+1} = x_i^k + \chi v_i^{k+1}. \tag{13}$$

If the particle evolution is feasible, cooperative evolution for SC implements the parameter χ to adjust the position and velocity of particles according to (13), where the value of τ increases with the increasing number of failed trials in keeping the feasibility of the particle evolution.

Step 6. Evaluation Cooperative evolution for SC calculates the fitness values of all particles and estimates their performance.

Step 7. Updating pBest and gBest For each particle, cooperative evolution for SC compares its current objective value with the objective value of its $pBest$. If the current value is better, then the SC updates $pBest$ and its objective value with the current position and objective value. After that, the cooperative evolution for SC determines the best particle of the current swarm with the best objective value. If the objective value is better than the objective value of $gBest$, SC updates $gBest$ and its objective value with the position, and objective value of the current best particle.

Step 8. Ranking Cooperative evolution for SC ranks individuals or particles according to their objective value. After that it returns a column vector that contains the corresponding individual fitness value.

Step 9. Selection The selection is an operator to choose two parent strings for generating new strings or offsprings. In the selection operation, a string with a high fitness value has more chance to be selected as one of its parents than a string with a low fitness value. Unlike GA where parent strings are selected by random choice, the parent strings in coperative evolution are not totally selected with a random manner. The fitness value of each string is used to select parent strings, akin to a roulette wheel selection that is a popular and well-known selection approach.

Step 10. Crossover Crossover is an operator to generate new strings or offsprings from parent strings. Our approach randomly selects several positions to exchange element and gains two new offsprings.

Step 11. Mutation Mutation is an operator to change elements in a string which is generated by a crossover operator. The mutation operator can be regarded as a transition from a current solution to its neighborhood solution in local search algorithms. It means to randomly select a position for changing elements.

Step 12. Elitist strategy It removes the worst string from the current population, and adds the best string in the previous population to the current one.

Table 1: The setting of parameter values for the proposed CE algorithm

The description of parameter	Value
The total number of training (epochs)	500
Population size	50
PSO algorithm part	
Inertia weight (W)	0.8
Learning factor (c_1, c_2)	$c_1 = c_2 = 1.472$
The maximum velocity of each particle (V_{max})	5
GA algorithm part	
Crossover rate (P_c)	0.65
Mutation rate (P_m)	0.05

4 PERFORMANCE EVALUATION AND ANALYSIS

To indicate the effectiveness and efficiency of the proposed approach, experimental cases with different scales were randomly generated to compare the performance with CPSO [21], and the recently proposed improved discrete immune algorithm based on CPSO (IDIPSO)[22]. These simulations were carried out using MATLAB 7.0 and a personal computer with an Intel Core2 Duo 2.10 GHz CPU.

4.1 Experiment Setting

Table 1 exhibites the common parameters for CPSO, Cooperative Evolution (abbreviated as CE) and IDIPSO. Each service is associated with four QoS attribute values: Cost (*C*), Availability (*A*), Time (*T*) and Reliability (*R*). Since our approach assigns the attribute Cost a higher weight than other attributes, the initial weights of the QoS attributes *C*, *A*, *T*, *R* are set as $\omega_C = 0.4$, $\omega_A = \omega_T = \omega_R = 0.2$ respectively. For convenience of comparison, the experiment takes four group with the task numbers of 20, 40, 60 and 80 for service composition operations respectively. In these cases, each task contains 15 candidate services. Accordingly, the search spaces for the potential service composition cases are 1520, 1540, 1560 and 1580, respectively. The QoS attribute values of candidate services are randomly generated from the interval $[0, 1]$. The same initial population is adopted for impartial comparisons among three algorithms in 30 rounds of independent computation.

4.2 Cooperative Evolution

The results of performance comparion in the experiments with CPSO, IDIPSO and CE are illustrated in table 2 that summarizes the mean best fitness (mean for short) and standard deviation (STD for short) of the optimal fitness values.

According to Table 2, it is not difficult to find that the mean of the best CE case is much better than those of IDIPSO and CPSO. The values of the test statistics among

Table 2: Performance comparison among PSO, IDIPSO and CE with various scales of SC cases

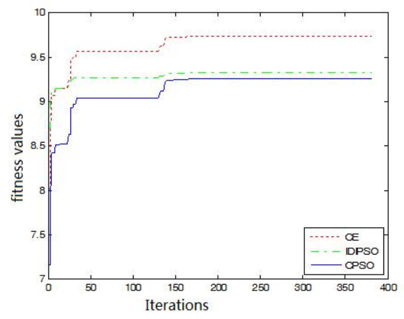
Comparision items	Tasks	The scale of experiment tasks			
		20	40	60	80
Mean	CPSO	7.1536	18.0763	28.3013	37.4307
	IDIPSO	8.0103	18.5778	26.9378	36.2268
	CE	8.6983	20.2341	30.8330	40.4683
STD	CPSO	0.2431	0.4813	0.7536	0.9967
	IDIPSO	0.0630	0.1259	0.1826	0.2455
	CE	0.1973	0.4143	0.6314	0.8287

CE, CPSO and IDIPSO are given as the *t*-test† columns in this Table. We observe that the mean results from CE, CPSO and IDIPSO have significant difference reaching to at least 6%. Moreover, the differences of the best mean values for CE, CPSO and IDIPSO increase with the increasing task numbers. For example, the pairwise differences of mean from CE and IDIPSO are 0.6880, 1.6563, 3.8952 and 4.2415 for SC with the task scale 20, 40, 60 and 80 respectively. At the same time, the mean value of CPSO divided by CE are 1.5447, 2.1578, 2.5317 and 3.0376, with respect to these SC cases. Regarding STD, CE is bettern than CPSO but worse than IDIPSO because CE is a stochastic algorithm. Especially for the small scale SC with 20 tasks, CE locates the optimal solutions in independent runs. Compared with particle swarm optimization and a genetic algorithm variant for IOT service selection and composition, these results indicate that CE with our proposed enhanced strategy performs well in the search aspect, and has excellent convergence and stability.

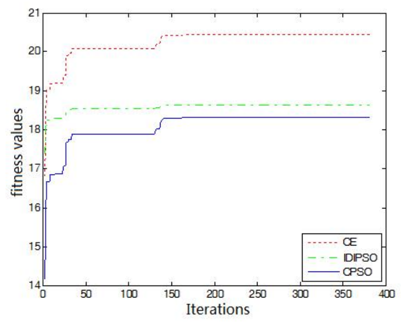
4.3 Online performance comparison among CPSO, IDIPSO and CE

Besides the analysis of the final static computational performance in Section 4.2, Fig. 4 graphically demonstrates the online average evolutionary performance comparison among CPSO, IDIPSO and CE in terms of the online average evolutionary performance of solving IOT SC issues. In Fig. 4, the abscissa represents the evolutionary generations and the vertical axis indicates the plot of the average best fitness values.

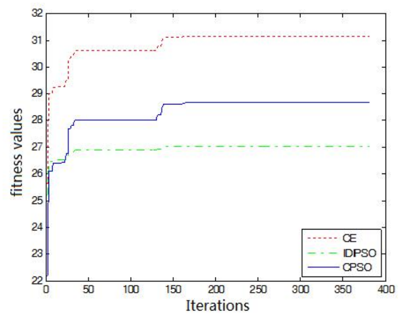
According to Table 2, the online evolutionary behavior comparison in Fig. 4 indicates that CE outperforms IDIPSO and CPSO greatly in the global exploration ability for promising solutions and the fine exploitation ability for the best global solution. Moreover, Fig. 4 also implies that the average best fitness values of CE surpasses that of the other two algorithms quickly in SC with different scales, which demonstrates the excellent search ability of CE for the promising solutions.



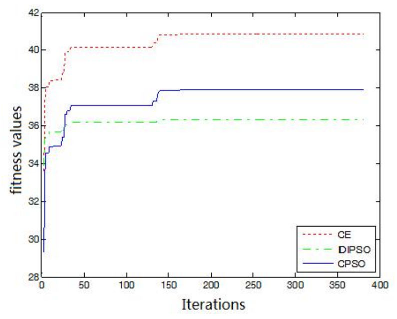
(a) SC with the scale of 20



(b) SC with the scale of 40



(c) SC with the scale of 60



(d) SC with the scale of 80

Fig. 4: Online evolutionary behavior comparison among PSO, IDIPSO and CE

5 CONCLUSIONS AND DISCUSSIONS

To address the issues of QoS-driven IOT service composition, this paper proposes a cooperative evolution algorithm based on particle swarm optimization. The proposed approach contributes the following: the improved local best first strategy guarantees that the component weights in the local fitness of a task and the fitness of SC are equivalent. The relative importance of QoS attributes of a IOT SC is as the same as the size of potential candidate services. Accordingly, the local improvement of a IOT service can correctly guide the algorithm search, when a better candidate service is selected to accomplish the corresponding task. The perturbance in the global best strategy effectively overcomes the insufficiency in dealing with the situation that all the individuals learn from the same global best solution. Our approach also considers the population diversity and selection pressure simultaneously. All these strategies guarantee the balance between coarse-grained exploration and fine-grained exploitation search. The experimental comparisons with CPSO and IDIPSO illustrate the excellent performance of CE for SC in the search convergence and the performance stability.

The proposed PSO based cooperative evolution algorithm for QoS-driven service composition is characterized by its ability to intelligently adapt over time. To begin with, unlike traditional service composition in the virtual information domain, service composition in IOT applications needs to process real-time data collected from electronic devices that are tightly coupled with the physical world. Further, the resource restrictions of SC operations are more prominent in IOT applications compared with conventional Web applications, such as the efficiency of distributed communication and computation and limitation of storage.

Experiments in Section 4 indicate the feasibility and performance of the proposed CE algorithm of QoS-driven IOT service composition. For the proposed CE, we observe that the means of its best values are an improvement over those of IDIPSO and CPSO. The experiments in Section 4 also imply higher efficiency of the proposed algorithm for finding potential SC solutions. Due practical limitations, the experimental data with different scales were generated according to partial data traces collected from real scenarios. For the future, further experiments with larger scales will be designed and carried out to verify intelligent adaptation of the proposed SC algorithm.

Acknowledgments This work was supported by the grants of the National Basic Research Program of China (2011CB707101), the National Natural Science Foundation of China (61070013, U1135005, 61272275 & 61171014), the NSFC Research Fund for International Young Scientists (61250110541), the Programme of Introducing Talents of Discipline to Universities (No. B07037), Guangxi Key Laboratory of Trusted Software (No. kx201421), National Fundamental Survey Project of Procreative Experiment (2011-5-2) from the State Bureau of Surveying and Mapping Geographic Information, Open Research Fund supported by State Key Lab. of Software Engineering, the Fundamental Research Funds for the Central Universities, and the Open Research Fund of Key Laboratory of Disaster Reduction and Emergency Response Engineering of the Ministry of Civil Affairs (LDRERE20120503).

REFERENCES

[1] Luigi Atzori, "*The Internet of Things: A survey,*" *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Liu, X. Li, L. Feng, "Resource space view tour mechanism," *Concurrency and Computation Practice & Experience*, vol. 20, no. 7,pp. 863-883, 2008

[3] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, Imrich Chlamta, "Internet of things: vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no.7, pp. 1497–1516, 2012.

[4] J. Liu, J. Zhou, J. F. Wang, F. Zhang, F. Liu, "Irregular community discovery for cloud service improvement," *Journal of Supercomputing*, vol. 61, no. 2, pp.317-336, 2012

[5] X. Luo, Z. Xu, J. Yu, X. Chen, "Building Association Link Network for Semantic Link on Web Resources", *IEEE Transactions on Automation Science and Engineering*, 8(3): 482-494, 2011.

[6] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, D. Savio, "Interacting with the SOA-based Internet of Things: discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223-235, 2010.

[7] Jan S. Rellermeyer, Michael Duller, Ken Gilmer, Damianos Maragkos, Dimitrios Papageorgiou, Gustavo Alonso, "The software fabric for the Internet of Things", *in Proceedings book of 1st International Conference on the Internet of Things*, Springer, pp.87-104, 2008.

[8] Angus F.M. Huang, Ci-Wei Lan, Stephen J.H. Yang, "An optimal QoS-based web service selection scheme," *Information Sciences*, vol. 179. no.19, pp. 3309-3322, 2009.

[9] S. Dustdar, W. Schreiner, "A survey on web services composition," *International Journal of Web and Grid Services*, vol. 1, no. 1, pp. 1-30, 2005.

[10] B. Srivastava, J. Koehler, "Web service composition-current solutions and open problems," *in Proceedings books of International Workshop on Planning for Web Services*, pp. 28-35, 2003.

[11] J. Hoffmann, P. Bertoli, M. Pistore, "Web service composition as planning, revisited: in between background theories and initial state uncertainty," *in Proceedings book of the National Conference on Artificial Intelligence*, vol.22, no.2, pp.1013-1018, 2007.

[12] L. Xing, Y. Chen, H. Cai, "An intelligent genetic algorithm designed for global optimization of multi-minima functions," *Applied Mathematics and Computation*, vol. 178, no. 2, pp. 355–371, 2006

[13] Fevrier Valdez, Patricia Melin, Oscar Castillo, "An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms," *Applied Soft Computing*, vol.11, no.2, pp.2625–2632, 2011.

[14] A. Sahu, S. K. Panigrahi, S. Pattnaik, "Fast Convergence Particle Swarm Optimization for Functions Optimization," *Procedia Technology*, vol. 4, pp. 319-324, 2012.

[15] J. Sun, X. Wu, V. Palade, W. Fang, C. H. Lai, W. Xu, "Convergence analysis and improvements of quantum-behaved particle swarm optimization," *Information Sciences*, vol. 193, pp. 81-103, 2012

[16] D. C. Shen, X. W. Xia, "A local search particle swarm optimization with dual species conservation for multimodal optimization", *in Proceedings book of International Conference on Information Computing and Applications*, Springer, pp. 389-396, 2012.

[17] W. Chu, X. Gao, S. Sorooshian, "Handling boundary constraints for particle swarm optimization in high-dimensional search space," *Information Sciences*, vol.181, no.20, pp.4569-4581, 2011.

[18] H. Huang, H. Qin, Z. Hao, A. Lim, "Example-based learning particle swarm optimization for continuous optimization," *Information Sciences*, vol. 182, no. 1, pp.125-138, 2012.

[19] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 182, no. 20, pp. 4515-4538, 2011.

[20] R. Poli, J. Kennedy, T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol.1, no. 1, pp. 33-57, 2007.

[21] Y. Shi, R. Eberhart, "A modified particle swarm optimizer," *in Proceedings book of IEEE World Congress on Computational Intelligence*, pp. 69-73, 1998.

[22] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, Y. Fan, "An Improved Discrete Immune Optimization Algorithm based on PSO for QoS-driven Web Service Composition," *Applied Soft Computing*, vol.12, no.8, pp.2208-2216, 2012.

[23] X. Luo, J. Yu, Q. Li, F. Liu, Z. Xu, "Building Web Knowledge Flows based on Interactive Computing with Semantics", *New Generation Computing*. 28(2): 113-120, 2010.

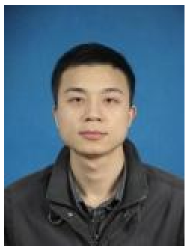
[24] R. Boyd, P. J. Richerson, "Culture and the evolutionary process," University of Chicago Press, Chicago, 1985.

[25] X. Luo, X. Wei, J. Zhang, "Guided Game-Based Learning Using Fuzzy Cognitive Maps", *IEEE Transactions on Learning Technologies*, 3(4): 344-357, 2010.

[26] R. J. Kuo, Y. J. Syu, Z. Y. Chen, F. C. Tien, "Integration of Particle Swarm Optimization and Genetic Algorithm for Dynamic Clustering," *Information Sciences*, vol. 195, pp.124-140, 2012.

Jin Liu is a professor at Computer School, Wuhan University, China. He received his Ph.D. degrees in State Key Lab of Software Engineering from Wuhan University, China, in 2005. He was a visiting scholar at department of Electrical & Computer Engineering, New Jersey Institute of Technology, USA, from Match, 2011 to April, 2012. His areas of research include software modeling on the Internet, service-oriented computing and knowledge processing. He has published more than 40 research papers.

Yuxi Chen is a postgraduate student in the State Key Laboratory of Software Engineering, Computer School, Wuhan University, China. He received his Bachelor degree from the Department of Computer Science at Wuhan University in 2013. His research interests include Software Engineering and interactive collaboration on the Web.



Xu Chen is a lecturer at the State Key Laboratory of Software Engineering of Wuhan University. He received his Bachelor degree from the Department of Computer Science at Wuhan University in 2004 and his Ph.D. degree from the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing at Wuhan University in 2009. His main research interests cover WebGIS, Software Architecture and Knowledge Engineering. As a core member, he has been actively involved in many projects, including National Natural Science Foundation of China, National Basic Research Program of China, National High-tech R&D Program of China, and etc. Xu Chen is the corresponding author of this paper (xuchen@whu.edu.cn).

Jianli Ding is a lecturer in the computer school, Wuhan University, China. He received his M.S. in Computer Science from the Harbin Institute of Technology, Harbin, in 1997, and Ph.D. from the Wuhan University, Wuhan, in 2006. His main research interests include image processing, neural networks and Big Data.

Kaushik R. Chowdhury is Assistant Professor in the Electrical and Computer Engineering Department at Northeastern University, Boston, MA. He received his M.S. in Computer Science from the University of Cincinnati, OH, in 2006, and Ph.D. from the Georgia Institute of Technology, Atlanta, GA in 2009. His M.S. thesis was given the outstanding thesis award jointly by the ECE and CS departments at the University of Cincinnati. He won the best paper award at IEEE ICC conference in 2009, 2012 and 2013, and currently serves on the editorial board of the Elsevier Ad Hoc Networks and Elsevier Computer Communications journals. His expertise and research interests lie in wireless cognitive radio ad hoc networks, energy harvesting sensors, and body area networks. He is a member of the IEEE.

Qiping Hu is a professor in the International School of Software, Wuhan University, China. His research is in the area of GIS Web service.

Shenling Wang is a lecturer of College of Information Science and Technology at Beijing Normal University, Beijing, China. She received her Ph.D. degree from Nanyang Technological University in 2012, Singapore. Dr. Wang's current research interests include Internet of Things, Semantic Link Networks and Computability Theory.

AUTHORS' ADDRESSES

Prof. Jin Liu, Ph.D.
Yuxi Chen, B.Sc.
Xu Chen, Ph.D.
Jianli Ding, Ph.D.
State Key Lab. of Software Engineering, Computer School, Wuhan University, China
email: mailjinliu@yahoo.com, {chenyuxi,xuchen,dingjianli}@whu.edu.cn

Asst. Prof. Kaushik Roy Chowdhury, Ph.D.
Electrical and Computer Engineering Department, Northeastern University, United States
email:krc@ece.neu.edu

Prof. Qiping Hu, Ph.D.
International School of Software,
Wuhan University, China
email:huqp@whu.edu.cn

Shenling Wang, Ph.D.
College of Information Science and Technology,
Beijing Normal University, China
email:wangsl0362@163.com

Received: 2012-11-11
Accepted: 2013-09-05