Accurate QoS-based Service Selection Algorithm for Service Composition

Jianxin Liao, Yang Liu, Xiaomin Zhu, Jingyu Wang,Qi Qi
State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
EB Information Technology Co. Ltd., Beijing, China
E-mail: {liaojianxin, liuyang_5, zhuxiaomin, wangjingyu, qiqi}@ebupt.com

***Abstract*—The most important thing of service composition (SC) is to select optimal candidate service instances compliant with non-functional requirements (e.g. QoS and load balance constraints). Particle swarm optimization (PSO) is known as an effective and efficient algorithm, which is widely used in this process. However, the premature convergence and diversity loss of PSO may result in suboptimal solutions. In this paper, we propose an accurate sub-swarms particle swarm optimization (ASPSO) algorithm which locates optimal solutions by using sub-swarms searching grid cells in which the density of feasible solutions is high. Simulation results demonstrate that the proposed algorithm improves the accuracy of the standard PSO algorithm in service composition.**

***Index Terms*—service composition, service selection, clustering method, particle swarm optimization**

I. INTRODUCTION

Services are implemented and distributed independently, duplicated service instances (SIs) are inevitable. Duplicated implies that SIs with the same function are implemented by different developers. Consequently, for a composite application, multiple SIs with different non-functional parameters exist for each service. The selection of optimal service instance for each service can barely guarantee the optimal QoS parameters of the whole composite application. Therefore, it is a challenge to select optimal SIs compliant with the requirements of whole composition request.

Particle swarm optimization (PSO) is known as an effective and efficient algorithm which can support multi-constraint multi-objective problems, whatever linear or nonlinear. In previous literatures [1-3], PSO has been proposed to solve the service selection problem properly. However, PSO's premature convergence and diversity loss may result in suboptimal solutions regardless of true optimal solutions. Sub-Swarm techniques have been proposed to improve the accuracy of PSO. However, these techniques all concentrate on the improvement of particles' neighbor topology. They neglect the reason of premature convergence, i.e. suboptimal solutions near true optimal solutions form traps what pull particles to converge to them. Inspired by this phenomenon, we propose an accurate sub-swarms particle swarm optimization (ASPSO) algorithm which locates optimal solutions by using sub-swarms searching grid cells in which the density of feasible solutions is high. It compares suboptimal solutions found in diverse sub-swarms which also avoid diversity loss in PSO. Furthermore, it uses an exterior file to record feasible solutions found in the whole solution space. Simulation results reveal that the accuracy of proposed algorithm is higher than the ones compared in searching the optimal solution of service selection problem.

The rest of paper is organized as follows. Section II presents the related work. The proposed algorithm is described in Section III. Section IV presents simulation results and analyses the performance of the proposed algorithm. Finally, Section V concludes this paper.

II. RELATED WORK

In researches on non-functional properties of service composition, there are some literatures using PSO. Fethallah et al. [4] adopted the local version of PSO to avoid local optima in service composition problems. In the local version of PSO, particles search for the optimum based on their inertias and optima which they have ever reached. This algorithm can be regarded as PSO with sub-swarms consisted of each particle. Although the adopted algorithm emphasizes PSO's exploration, its nature based on randomness and local optima always lead to suboptimal solutions. Liao et al. [5] adopted the lbest PSO with ring topology [6] to find one true optimum from local optima in service composition. In this algorithm, every three adjacent particles constitute a sub-swam to select the best local optimum as their global optimum in the sub-swarm. Therefore, all particles form a ring topology which slows down the convergence procedure of the algorithm. Li proved that this algorithm outperforms most existing PSO algorithms. Although existing sub-swarm mechanisms improve the accuracy of algorithms, they ignore the direct factor of the premature convergence of PSO. Sub-optima around the true optimum are traps to all particles. In this paper, we propose a novel PSO algorithm which locates the optimal solution by using multiple sub-swarms to isolate and search candidate areas of sub-optima. Simulation results show that the proposed algorithm is more accurate than compared algorithms.

III. ACCURATE SUB-SWARMS PSO

Although previous research [1-3] adopt PSO to search for optimal solutions in SC problems, the premature convergence of PSO may result in suboptimal solutions in practical use. To

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

illustrate the impact of this feature, we inspect a solution space of MCOSCP. As shown in Fig. 1, points represent feasible solutions of a MCOSCP problem. Better the solution is, higher the value of point is. The value of points is indicated by the size and color of them. In Fig. 1, feasible solutions gather in almost four clusters. In addition, there may be better solutions in a cluster where crowded by more feasible solutions. But when a swarm establishes a suboptimal solution as its global optimum in the later stage of PSO, the suboptimal solution will draw all particles in. In problems full of suboptimal solutions just like MCOSCP, this feature of PSO could pull particles far away from true optima.

To improve the accuracy of PSO, we propose a novel sub-swarms algorithm named accurate sub-swarms particle swarm optimization (ASPSO). In the initialization stage, there is one swarm called the main-swarm in the ASPSO. With the increasing iteration number, sub-swarms are generated around the dense areas of feasible solutions and isolated from the main-swarm. In this process, the main-swarm still searches the remaining area in the solution space until the algorithm stops when the maximum iteration number is reached. ASPSO takes advantage of accurate sub-swarms construction method and competition mechanism to locate optima.

The ASPSO utilizes the feature of solution space to establish sub-swarms in areas where densities of feasible solutions are high. Inspired by serial and parallel niching technique, ASPSO isolates main-swarm and sub-swarms updating all particles in each iteration. Below, we specify procedures of ASPSO different from the standard PSO.

Grid cell file (GCF) and feasible solution file (FSF) are total new modules in ASPSO. In ASPSO, sub-swarms construction is based on the knowledge of the dense areas of feasible solutions. Hence, ASPSO use a FSF to store coordinates of feasible solutions which particles ever reach and grid cell identifiers which feasible solutions belong to. To locate the dense areas of feasible solutions, ASPSO adopts a simple clustering method which requires dividing each dimension of solution space to grids. Thus grid cells are high dimensional units with the same length with particles. There are two terms in one record of GCF: grid cell identifier and feasible solution amount. Grid cell identifier is a vector with the same dimension as particles. Maximum value of its each dimension is determined by grid cell factor. And the feasible solution amount could increase when the algorithm finds a new feasible solution in this grid cell.

In the input of ASPSO, SwarmSize is the total number of particles used in the algorithm. ParticleScope is the value range of solutions in all dimensions which is determined by the service composition problem. GridCellFactor is a multi-dimensional value which determines to how many grids each dimension of the solution space is divided. Each grid cell's range can be acquired by calculation of ParticleScope, GridCellFactor and grid cell identifier. T_{max} is the total iteration time of the algorithm. The sub-swarms update period (SUP) is the period in which sub-swarms are updated. When the iteration time is one, all particles are initialized. Then iteration time increases until the maximum iteration is reached. In each

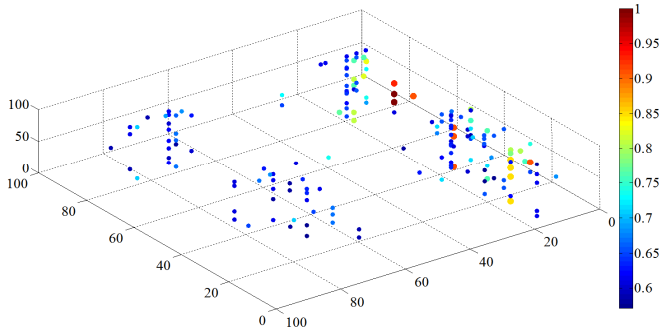

Fig. 1. Optimal and feasible solutions in a MCOSCP solution space

iteration, the algorithm first checks whether iteration time reaches SUP. If so, the algorithm first finds dense areas of feasible solutions, and then updates sub-swarms based on them. Then, the algorithm updates particles' velocities and positions. A solution with the best fitness value is selected from global optima of all swarms as the final result for the service composition.

Algorithm 1. Initialization function of ASPSO

```
1  input: SwarmSize, ParticleScope, GridCellFactor;
2  for each particle in the main-swarm do
3     initialize positions and velocities randomly;
4     restrict positions and velocities according to the ParticleScope;
5     initialize particles' local optima to be their corresponding positions;
6  end for
7  select global optimum to be the one with best fitness value in local optima;
8  Initialize GCF according to the ParticleScope and GridCellFactor;
9  for position of each particle do
10    FSF.check_insert(position);
11 end for
12 for feasible solution amount of each grid cell in GCF do
13       Increase feasible solution amount according to the number of feasbile
solutions found in this grid cell;
14 end for
15 output: MainSwarm,GCF, FSF;
```

In the initialization function as shown in Algorithm 1, all particles belong to the main-swarm. First, the algorithm initializes particles scattering randomly in the solution space. Particles' velocities are set the same with their positions. Second, positions and velocities are restricted according to the range of solution space. Position bounds of each particle are the same with the range of solution space. The extreme velocity is plus or minus difference between maximum and minimum value of solution space. Local optima of particles are defined as their positions at this time. The global optimum is selected from local optima based on their fitness value in line 7. Then, the GCF is initialized to get ready for the division of feasible solutions in line 8. Finally, each particle's position is evaluated to decide whether it is a feasible solution in line 9-11. Feasible solutions will be inserted into the FSF for the clustering method. Meanwhile, feasible solution amounts of corresponding grid cells in which feasible solutions reside will increase accordingly in line 12-14.

If iteration time occurs to be the sub-swarms update period (SUP), ASPSO first evaluates dense areas of feasible solutions and updates sub-swarms before updating velocities and

positions of particles. ASPSO adopts a simple clustering method to generate sub-swarms from grid cells in which densities of feasible solutions are higher than surroundings. The dense areas evaluation and sub-swarms update process is presented in Algorithm 2-3.

Algorithm 2. Find_dense_areas function of ASPSO

```
1  input: ParticleScope, GCF, FSF;
2  for each grid cell in GCF do
3    if FeaSolAmount/AveDense >1 then
4      GridCellCheck.insert(grid cell);
5    end if
6  end for
7  for each GridCell(k) in GridCellCheck do
8    if GridCell(k) is unchecked then
9      mark GridCell(k) checked by GridCell k;
10     initialize TempCheck = GridCell(k);
11     for each unchecked GridCell(h) in GridCellCheck (h>k) do
12       if GridCell(h) is adjacent to any grid cell in TempCheck then
13         TempCheck.insert( GridCell(h) );
14         mark GridCell(h) checked as a neighbor of GridCell(k);
15         h=h+1;
16       end if
17     end for
18       calculate the number of feasible solutions and the total max and min
           ranges of all grid cells in TempCheck; merge these two terms to
           one record and insert it to CheckResult;
19   end if
20 end for
21 output: CheckResult;
```

In line 3 of Algorithm 2, FeaSolAmount means feasible solution amount found in this grid cell of GCF; AveDense is equal to total feasible solution amount divided by total grid cell amount. GridCellCheck in line 4 stores identifiers and corresponding feasible solution amounts of grid cells in which densities of feasible solutions found are higher than the average level. Line 7-20 is a simple clustering method which finds dense regions of feasible solutions using grid cells. For each unchecked GridCell(k) in GridCellCheck, the algorithm checks every GridCell(h) (h>k). If GridCell(h) is a neighbor adjacent to GridCell(k), it will be added to TempCheck. If there is none unchecked neighbor of GridCell(k) in GridCheck, TempCheck stores all direct and indirect adjacent grid cells of GridCell(k) including itself. Then in line 18, feasible solution amount and range of these cells are added to CheckResult. If all grid cells are checked, this function will return CheckResult.

Algorithm 3. Update_sub-swarms function of ASPSO

```
1  input: SwarmSize, ParticleScope, MainSwarm, FSF, CheckResult,GCF;
2  if CheckResult != null then
3    select the optimal solution as main-swam's new global optimum from
       existing sub-swarms' global optima and main-swarm's global
       optimum;
4      retrieve all particles and reconstruct sub-swarms according to
CheckResult;
5    update the range of main-swarm;
6    initialize positions and velocities of sub-swarm particles randomly;
7    restrict positions and velocities according to the sub-swarms' range;
8    initialize local and global optima of sub-swarm particles;
9    for position of each particle in newly generated sub-swarms do
10     FSF.check_insert(position);
11   endfor
12   for feasible solution amount of each grid cell in GCF do
13     Increase feasible solution amount according to the number of feasbile
solutions found in this grid cell;
14   end for
15 endif
16 output: MainSwarm, SubSwarms, FSF,GCF;
```

After all dense areas of feasible solutions are stored in CheckResult, sub-swarms are generated with a preference for denser and larger areas as shown in Algorithm 3. In each sub-swarms update period, all particles are retrieved to generate new sub-swarms. Sub-swarms are reconstructed according to the feasible solution dense areas at this time. This method could keep the sub-swarms track up to date dense areas of feasible solutions. And the total particle amount is maintained to be SwarmSize. The particle amount assigned to each sub-swarm is proportional to the ratio of its size in the whole solution space. The algorithm brings in none more particles and keeps the same particle density in the solution space with the basic PSO. Constant particle density is sufficient for MCOSCP problems. This mechanism could avoid greedily increasing particles which may induce heavy computational complexity. Before particles are retrieved, the algorithm selects main-swarm's new global optimum from existing sub-swarms' global optima and the main-swarm's global optimum to improve search result in line 3. The first step of a sub-swarm generation is to determine the search range of sub-swarm. The search range is mainly based on the corresponding dense area range in CheckResult. If the size of a dense area is not qualified to be searched by two particles, the algorithm will properly enlarge the search range of the sub-swarms to keep the particle density. Then the algorithm assigns corresponding amount of particles to this sub-swarm. The algorithm constructs sub-swarms with preference for denser and larger areas in CheckResult until remaining particles are not enough for a sub-swarm. In line 5, the range of main-swarm is updated to ensure main-swarm and sub-swarms are isolated. Once sub-swarms are generated, the algorithm initializes positions and velocities of particles randomly within the value of sub-swarms' search ranges in line 6-7. Position bounds are the same with the corresponding sub-swarm's search range. The extreme velocity is plus or minus difference between the maximum and minimum value of corresponding sub-swarm's search range. In line 8, the algorithm initializes local and global optima just like that in main-swarm. Then if a position is a feasible solution, it will be inserted to the FSF in line 9-11. Feasible solution amounts of corresponding grid cells in which feasible solutions reside will increase accordingly in line 12-14.

The secondary step of iteration stage is the update of particles. This process is quite similar with corresponding process presented above except updating positions and velocities. Before the selection of local and global optimum of each particle, its velocity and position must be restricted according to the range of swarm to which it belongs. At the end, positions are inserted to the FSF if they are feasible solutions. Feasible solution amounts of corresponding grid cells in which feasible solutions reside will increase accordingly.

IV. Performance Evaluation

In this section, we evaluate the performance of proposed ASPSO algorithm by comparing it with standard PSO and lbest PSO with ring topology [6]. The lbest PSO with ring topology (LPSO-RT) induces stable niching ability in finding one true optimal solution in the presence of massive suboptimal solutions [5]. Simulation results demonstrate that ASPSO improve the accuracy of PSO in MCOSCP problems.

We use execution time and accuracy rate (the fitness value of the optimal SCP which the algorithm finds divided by the fitness value of the real optimal SCP) as metrics to evaluate the proposed algorithm.

We compare the performance of three algorithms with increasing iterations when the service instance number is 10^3,10^4 and 10^5 respectively. As shown in Fig. 2, the number of service sets and swarm size remain 10 and 20 respectively. In these scenarios, the accuracy of ASPSO surpasses the other two algorithms. When there are few service instances, the accuracy of these three algorithms are close. When there are enormous service instances, the accuracy of ASPSO is almost 5% higher than LPSO-RT, 10% higher than PSO. Moreover, the performance of ASPSO is steadier than other algorithms. Fig. 2 (d-f) show the time consumption of these algorithms. When iterations are less than 200, ASPSO spends at most 2 seconds more than other two algorithms. But with the iterations increase, the execution time of ASPSO rises quickly. This may result from the sub-swarms update consumption. The accuracy of ASPSO increases inconspicuously when the iterations are more than 200. In practical service composition problems, we could use ASPSO with less than 200 iterations. Therefore, ASPSO would induce more accurate results with just 1-2 seconds more execution time.

V. Conclusions

It is a challenge to select optimal service instances considering multiple constraints in service composition. PSO is known as an effective and efficient algorithm for solving service composition problems. However, the premature convergence and diversity loss of PSO may induce particles trapped in local optima. In this paper, we propose an accurate sub-swarms particle swarm optimization algorithm (ASPSO) for the service composition problems. It constructs sub-swarms around areas where densities of feasible solutions are high. A simple clustering method is adopted to distinguish dense areas in the ASPSO algorithm. The algorithm adopted serial and parallel niching technique to isolate main-swarm and sub-swarms searching different areas in the mean time. This mechanism could avoid particles converging to local optima by distributing search units more evenly than standard PSO. Also, this algorithm enhances the accuracy of result by fully searching areas where true optimum may probably exist. In simulations, we test PSO, LPSO-RT and ASPSO. Simulation results demonstrate that ASPSO is more resistant to local optima. ASPSO is more accurate than compared algorithms with acceptable increased time consumption. In the future, we will improve the dense areas identifying method to reduce ASPSO's computation complexity. Moreover, we will research on service selection algorithms for multi-objective service composition.

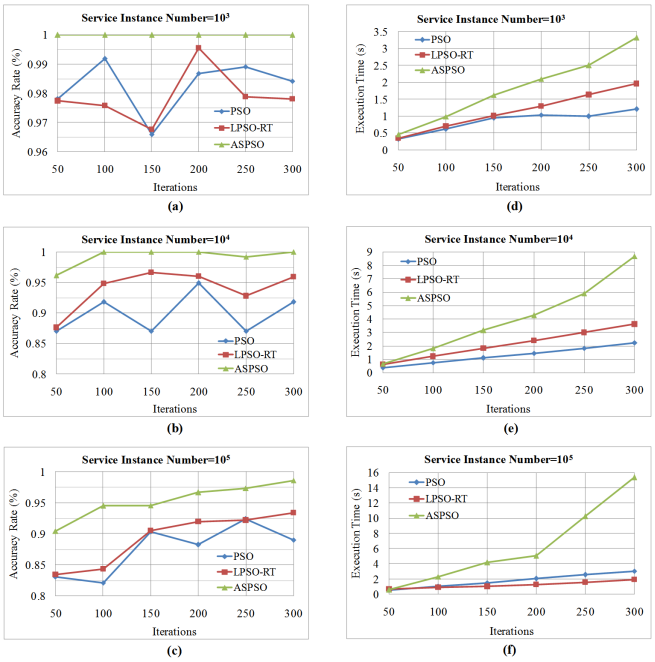

Fig. 2. Performance comparison with increasing iterations.

Acknowledgment

This work was jointly supported by: (1) the National Basic Research Program of China (No. 2013CB329102); (2) National Natural Science Foundation of China (No. 61271019, 61101119, 61121001, 61072057, 60902051); (3) PCSIRT (No. IRT1049).

References

[1] T. Guha and S. A. Ludwig, "Comparison of Service Selection Algorithms for Grid Services: Multiple Objective Particle Swarm Optimization and Constraint Satisfaction Based Service Selection," in Proc. IEEE Int. Conf. of Tools with Artificial Intelligence, 2008.

[2] H. Xia, Y. Chen, Z. Li, H. Gao, and Y. Chen, "Web Service Selection Algorithm Based on Particle Swarm Optimization," in Proc. IEEE Int. Conf. of Dependable, Autonomic and Secure Computing, 2009.

[3] J. Cao, X. Sun, X. Zheng, B. Liu, and B. Mao, "Efficient Multi-objective Services Selection Algorithm Based on Particle Swarm Optimization," in Proc. of IEEE Asia-Pacific Services Computing Conf., 2010.

[4] H. Fethallah, M.A. Chikh, M. Mohammed, and K. Zineb, "QoS-aware service selection based on swarm particle optimization," in Proc. Int. Conf. of Information Technology and e-Services, 2012.

[5] J. Liao, Y. Liu, J. Wang, and X. Zhu, "Service Composition based on Niching Particle Swarm Optimization in Service Overlay Networks," KSII Trans. on Internet and Information Systems, vol. 6, no. 4, pp. 1106-1127, 2012.

[6] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," IEEE Trans. on Evol. Comput., vol. 14, no. 1, pp. 150-169, 2010.