Sky-MCSP-R: an efficient graph-based Web service composition approach

Pengjiao Sun¹, Pengcheng Zhang¹³, Wenrui Li², Xuejun Guo¹, Jun Feng¹
¹College of Computer and Information, Hohai University, Nanjing, P.R.China 210098
²School of Mathematics & Information Technology, Nanjing Xiaozhuang University, P.R.China, 211171
³State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, P.R.China, 210093
E-mail: pchzhang@hhu.edu.cn

Abstract—Aiming at optimizing Web service composition which satisfies user's multiple QoS constraints, an efficient graph-based Web service composition approach, named Skyline-improved Multi Constraint Shortest Path-Relax (Sky-MCSP-R), is proposed. Firstly, the approach selects Skyline services from candidate service spaces, thus it can construct the model of Web service composition directly on these high quality candidate services, reducing the whole number of nodes of the model. Secondly, the approach uses McSP-K algorithm which uses over-constraint mechanism to compose basic services, and reduces the constraint intensity so to make algorithm MCSP-K produce as many feasible solutions as possible. Thirdly, the approach uses Relax algorithm to optimize the solutions. Experimental results show that the approach improves the efficiency of Web service composition and keeps a high optimization rate, reducing no solution phenomena. The results also indicate that our approach is superior to previous approaches.

Index Terms—Quality of Service(QoS); Web service composition; MCSP-K; Skyline; over-constraint mechanism

I. INTRODUCTION

With the development of Web services technology, it has become an inevitable trend to compose complex service applications from individual (atomic) services to satisfy user's multiple QoS constraints. Normally, there are a lot of Web services that have the same function but with different QoS (Quality Of Service) to be published and shared on the Internet [4], scholars did many researches on how to choose appropriate services from these individual ones to compose a large-grained service of high quality. The commonly used service optimization approaches currently include evolutionary algorithms which is represented by genetic algorithm [12] and graph search algorithms which is represented by directed graph [5][6][3][15]. Although genetic algorithm could solve the global optimal problem, it likely leads to poor convergence problem and produces unstable outcomes [8]. Furthermore, time complexity is also unacceptable when the data is large scale. Genetic algorithm is also relatively matured and has not too much space for improving. Web service composition approaches based on the graph has advantages due to the concise model, the ability to handle multiple route choices, and it only need to run once in order to find the optimal composition. Consequently, it is frequently used in practice [7]. The searching efficiency of graph-based search algorithms could be acceptable when the scale of candidate services is small. when the scale continues to increase, the number of nodes and edges of graph would also increase exponentially, and even come up space explosion problem. However, there is large space for the the improvement of searching efficiency. Therefore, this paper will focus on the improvements of optimizing Web service composition based on graph.

In order to improve the efficiency of Web service composition approach based on graph, this paper applies the latest research result to the graph approach and to get a new Web service composition approach, named Skyline-improved Multi Constraint Shortest Path-Relax (Sky-MCSP-R). The approach divides the Web service composition process into three phases, it firstly selects Skyline services from candidate service space, then the improved algorithm MCSP-K is adopt to obtain as many feasible solutions as possible, finally the optimal solution is obtained by using algorithm Relax.

The remaining sections are organized as follows: section II reviews and summarizes some related work. Section III introduces the Web service composition model based on graph. Section IV proposes the composition method in three phases, resulting in three subsections A, B, C respectively, and also provides corresponding algorithm for each phase. Section V validates the proposed approach and compare to traditional approach with a series of simulation experiments. Section VI summarizes the paper and put a list of future work.

II. RELATED WORK

Related work could be mainly divided into three categories: the first category is the research of Web service composition method based on the graph; the second category focuses on how to improve composition efficiency by using new technologies; the third category is the research of Web service composition with QoS constraints. All the three categories are closely related to our proposed approach.

Firstly, Web service composition approaches based on graph abstracts services in the candidate service space and relationships between services as a directed graph, thus constructs a graph model of Web service composition, graph algorithm is then used to calculate the optimal combination plan. The graph algorithm applied to service composition mainly include
the dynamic programming algorithm, the Dijkstra algorithm, the Bellman-Ford algorithm, the MCSP-K algorithm, and so on. [5] regards each service class as a stage, regards services in the service class as nodes of the directed graph, and nodes in adjacent stage are linked together, regards invocation cost of a adjacent service as weight of a directed graph, the minimum total cost can be calculated from initial node to final node by using algorithm dynamic programming. Because clearly classify stages according to the service class, this method is only suitable in single path situation, not satisfy the demand of user’s multiple path selection. [6] regards input/output information as nodes of the directed graph, regards service from input information to output information as an edge the directed graph. Then QoS value of a service could be regard as weight, thus the minimum total QoS values could be calculated from the initial information to destination information by using algorithm Dijkstra. The method can deal with multi-path selection, but because the input and output information of each service are based on specific situation, the method is not so practical. [3] uses algorithm Bellman - Ford, the algorithm is similar to algorithm Dijkstra, but weights of services can be negative, so can be used to solve composition of negative correlation between services. The method does not take user’s constraints into account. Algorithm MCSP-K in [15] considers user’s multiple QoS constraints, and the graph model using by MCSP-K supports multiple path selection. Time complexity of these composition methods can be accepted in smaller candidate services scale, as the scale continues to increase, nodes and edges of the directed graph would also increase exponentially, which affect the efficiency of composition. Considering the superiority algorithm MCSP-K possesses comparing with other graph algorithms, this paper deeply studied MCSP-K, which is the basis of composition approach Sky-MCSP-R.

Secondly, in recent years, many experts try new theory and techniques to improve the efficiency of Web service composition. Three representative approaches are studied: division to solve, global constraint decomposition and reduction of candidate service space. In [1], the authors use method of division to solve, and the main idea of the approach is “divide and conquer” which is introduced in. The composite service is decomposed into several parts according to certain rules, then several parts are solved respectively to improve the efficiency of composition. The decomposition technique is vital to service composition in this process, and it will influence quality of the final solution. In [13], the corresponding authors propose a novel method which decompose global constraints into local constraints on the basis of global optimization method, then do service selection by using local optimization method which could select services that satisfied users’ end-to-end QoS constraints with as short as possible. However, because algorithm for decomposition also costs much time, the choice of the decomposition algorithm is a challenge. The main idea of [2] is using the Skyline method to reduce the candidate service space to speed up the solving efficiency of service composition scheme. Linear programming algorithm is also used in [2] to compose services. However, in this approach users QoS constraints have not been taken into account. Different to them, Skyline technology was applied to service composition based on graph in this paper, which serves as the first phase of approach Sky-MCSP-R.

Finally, because it is difficult to introduce QoS constraints into the Web service composition approaches based on graph, a lot of approaches in literature have not taken user’s QoS constraints into account, such as [5][6][3]. In [15], the authors puts forward a Web composition approach which satisfies user’s multiple QoS constraints. However when the number of constraints increases, composition efficiency would be affected a lot, furthermore it also likely lead to over-constraint condition which will lead to no solution phenomenon. In [11], the authors introduces a over-constraint mechanism which divide user’s constraints into hard constraints and soft constraints so as to relax constraints accordingly, reduce no solution phenomenon. Our approach absorb some main ideas from the thought of [11]. The service composition approach in this paper takes user’s QoS constraints into account. However, compare to [11], the problem of over constraint is resolved to a certain extent, and consequently our approach can reduce no solution phenomenon.

III. WEB SERVICE COMPOSITION MODEL BASED ON GRAPH

A large-grained service that satisfies users’ demand could be composed of several small-grained services. Each small-grained service corresponds to one task that accomplish a function of the large-grained service. There are many candidate services on the Internet that could accomplish one specific task. That is to say, these candidate services have the same functional property. However having the same functional property does not imply having the same non-functional properties. Commonly used non-functional properties include delay, cost, reliability, credibility, availability. We call these non-functional properties as quality of service (QoS) [16].

Each QoS attribute value \( q_i(1 \leq i \leq l) \) could be normalized [10] to a value \( q_i \) in interval [0, 1]. The higher the value of service has, the greater value it has.

Supposing there is a QoS attribute whose weight was set to \( w_i \) by user, then the total QoS value of an individual service \( s_i(1 \leq i \leq n) \) could be represented as:

\[
Q(s_i) = \sum_{j=1}^{l} w_j q_j(s_i)
\]

The total QoS value \( S \) could be linearizedly represented as:

\[
Q(S) = \sum_{i=1}^{n} \sum_{j=1}^{l} w_j q_j(s_i)
\]

\[\mathcal{C} = \{C_1, C_2, \cdots, C_m\}, \quad 1 \leq m \leq l \text{ represents user’s global QoS constraint set, then:}\]

\[
\sum_{i=1}^{n} q_j(s_i) \leq C_k \quad \forall C_k \in \mathcal{C}
\]
Web service composition model based on graph abstracts each candidate service as node of a directed graph, abstracts the precursor and successor relationship between services as edges of a directed graph. Figure 1 is such a directed acyclic graph (DAG). Each rounded rectangle box in Figure 1 stands for a candidate service class $S_i(i \geq 1)$ which could accomplish a specific service task, candidate service in service class are represented by oval $V_i(i \geq 1)$ in rounded rectangle box, the directed edges stand for the predecessor and successor relationships between services. If there is a directed edge between $S_i$ and $S_{i+1}$, task $S_{i+1}$ is optional when task $S_i$ is completed. The initial node $V_s$ and final node $V_d$ stand for two virtual services.

The service composition problem with multiple QoS constraints can be converted into a problem which looks for the optimal path of service composition from initial node to final node. The path has a higher QoS value calculated by using formula (2), and meets formula (3). Because there are multiple group of service tasks could be chosen to accomplish a specific function of a composite service, the Web service composition model based on graph could easily handle multiple path selection. In order to improve the efficiency of service composition, the proposed service composition method Sky-MCSP-R would reduce the node size of the graph model by removing nodes which is impossible to appear in the optimal path of service composition. Then by using composition algorithm MCSP-K with the over-constraint mechanism introduced, we hope to obtain as many feasible solutions as possible. Finally we obtain the optimal solution from these feasible solutions.

IV. WEB SERVICE COMPOSITION APPROACH SKY-MCSP-R

Sky-MCSP-R is a Web service composition approach in three phases based on graph. The first phase is to select Skyline services from candidate service space, thus obtain some high quality candidate services for each service class, and reduce the node scale of DAG. The second phase is to compose services by using algorithm MCSP-K whose over-constraint mechanism is introduced in to obtain feasible solutions. The third stage is to use algorithm Relax to get the optimal solution. Sky-MCSP-R could improve the solving speed on the basis of keeping a high optimization rate and reducing the no solution phenomenon.

A. Selecting Skyline services

The concept of Skyline could be derived from database [9], scholars have applied it to Web service composition in recent years. The Skyline method in [14] runs through the whole process of service composition, leads to poor flexibility. In [2], the authors make use of the Skyline method only for screening services. Service composition approaches can adapt different method according to its demand, and thus has more flexibility. This paper also uses the Skyline method in screening, and the selected service is called Skyline service, which is defined as follows.

Definition 1: (Skyline service). Assuming that service $P$ is one of candidate services in a service class $S_i$, there is no other candidate service $Q$ can dominate service $P(Q < P)$. That is to say, there does not existing another candidate service $Q$ whose QoS attribute values are all less than or equal to service $P$’s, and thus service $P$ is called one of the Skyline services of service class $S_i$.

We can prove that a non-Skyline service could not be one of composition services in optimal solution, and services in the optimal solution must be a Skyline service [14]. In [2], the authors do not provide the detail algorithm on how to get Skyline services from the candidate service space. Considering the characteristic of Web service composition, we design the Skyline algorithm shown in Algorithm 1.

Algorithm 1 Skyline Services (Skyline)

Require: $S_i, V_k$
Ensure: set of Skyline services $L_i$

for each candidate service $V_k \in S_i$ do
    generate the graph node $V_k$;
    put $V_k$ in ArrayList $L_i$
    for each candidate service $V_j$ ahead of $V_k$ in $L_i$, $k \geq j$ do
        if $V_j < V_k$ then
            delete $V_k$ from $L_i$;
            break;
        else if $V_k < V_j$ then
            delete $V_j$ from $L_i$;
        end if
    end for
end for

Algorithm Skyline scans the candidate service $V_k$ in the service class $S_i$ in turn and generates the graph node $V_k$ correspondingly, then puts $V_k$ in ArrayList $L_i$ (rows 1 ~ 3). If service $V_k$ is in the domination of one service $V_j$ ahead in $S_i$, that is to say, $V_k$ is not one of the Skyline services, the graph node $V_k$ which is corresponding to service $V_k$ should be deleted (rows 5 ~ 7). If service $V_j$ dominates service $V_k$, i.e., service $V_j$ is not the Skyline service, we can remove graph node $V_j$.
corresponding to the service \( V_j \) (rows 8 ~ 9). Thus, the non-Skyline service in candidate service space is eliminated, we can construct the model of Web service composition based on graph directly on the basis of high quality candidate services.

B. Seeking feasible solutions

In order to satisfy user’s multiple QoS constraints and support multiple path selection, this paper selects algorithm MCSP-K to compose basic services. The efficiency of service composition is improved after Skyline service selection in phase 1. However in some cases, no solution phenomenon would come up. This phenomenon belongs to over-constraint problem. The over-constraint problem in the Web service composition refers to the no solution phenomenon which is that we cannot find a group of composite services which satisfies user’s demand causing by too many or tight constraints [11]. When the candidate service space is narrowed after using Skyline method, over-constraint problem is more likely to appear. To solve the problem, this paper introduces an over-constraint mechanism in algorithm MCSP-K by learning ideas in [11]. We divide user’s constraints into hard constraints and soft constraints. Hard constraints are constraints which must be satisfied in the process of service composition, and they are hard QoS constraints the user offer and meet formula (3). Soft constraints reflect user’s expectations in some QoS attributes in the process of service composition, and they are optional constraints and not necessarily meet formula (3), but whether they are satisfied affects scores of the composition path. Differently from [11], this paper only considers hard constraints when running algorithm MCSP-K so that we can improve the composition more efficiency. Consequently we get algorithm MCSPi improved from MCSP-K, and the new algorithm is shown as Algorithm 2.

In order to find out a high quality composition path, each node reserves multiple paths which satisfy hard constraints \( C_{hard} \) from initial node \( V_i \) to its own’s (rows 8 ~ 9 ). To reduce the number of each node’s reserved paths, we just keep the first \( K \) paths whose cost are lower and which are more likely to be the optimal solution (rows 10 ~ 11). The feasible solutions are obtained in the final node \( V_d \).

C. Achieving optimal solution

We can get the optimal solution from feasible solutions which were obtained from the composition algorithm MCSPi by using algorithm Relax. In order to guarantee the performance of optimal solution, we now consider soft constraints \( C_{soft} \) in phase 3. Assuming that we can find \( n \) paths in the final node \( V_d \) after running algorithm MCSPi, namely \( n \) feasible solutions were obtained in phase 2. Computing QoS value of each of the \( n \) paths again, we can add a corresponding value to each path according to the number of soft constraints the path meets. That is to say corresponding constants \( C' \) to formula (2) is added:

\[
Q(S) = \sum_{i=1}^{n} \sum_{j=1}^{l} w_{ij} q_{ij}(s_i) + C' \tag{4}
\]

Algorithm 2 improved MCSP - K (MCSPi)

**Require:** \( G = (V,E), V_i, C_{hard} \)

**Ensure:** paths of \( V_d \)

1. for each node \( V_i \in G \) in topological order do
2. for each node \( V_j \) linked with \( V_i \) do
3. if \( V_i == V_j \), then
4. \( Q(V_i \rightarrow V_j) = Q(V_i \rightarrow V_j); \)
5. else
6. \( Q(V_i \rightarrow V_j) = Q(V_i \rightarrow V_j) + Q(V_i \rightarrow V_j); \)
7. end if
8. if \( Q(V_i \rightarrow V_j) \) satisfies \( C_{hard} \) then
9. add \( P(V_i \rightarrow V_j) \) to node \( V_j \);
10. for each path \( p \) in \( V_j \) do
11. if \( Q(p) < Q(V_s \rightarrow V_j) \) and \( C_{hard}(p) \leq C_{hard}(V_s \rightarrow V_j) \) then
12. remove \( p \) from \( V_j \);
13. end if
14. end for
15. if \( size(paths \ of \ V_j) > K \) then
16. keep the first \( K \) paths whose cost are lower;
17. end if
18. end if
19. end for
20. end for

Then the optimal path is the path whose \( Q(S) \) value is highest. The corresponding algorithm is shown in Algorithm 3. The algorithm examines all feasible solutions, scans all paths

Algorithm 3 Relax

**Require:** \( V_d, C_{soft}, level \)

**Ensure:** optimal solution \( p \)

1. for each path \( p \in V_d \) do
2. compare \( C_1 \) with \( C_{soft} \) to get number \( n \) which satisfies \( C_{soft} \);
3. if \( n/(size \ of \ C_{soft}) > \frac{2}{3} \) then
4. \( C' = level + 0.5; \)
5. else if \( \frac{1}{3} \leq n/(size \ of \ C_{soft}) \leq \frac{2}{3} \) then
6. \( C' = level; \)
7. else
8. \( n/(size \ of \ C_{soft}) < \frac{1}{3} \)
9. \( C' = level - 0.5; \)
10. end if
11. recalculate the \( Q(S) \) value of the path using formula (6);
12. end for
13. return the highest \( Q(S) \) value path;

in turn, and calculates the number of soft constraints each path meets (rows 1 ~ 2). According to the relative size of the number \( n \) and soft constraint size, the constant value \( C' \) (rows 3 ~ 10) is determined. Then formula (4) is used to recalculate the \( Q(S) \) value of the path (11 row). Path with the
highest $Q(S)$ value is the optimal path, namely the optimal solution (13 row).

**Time Complexity Analysis.** Supposing that a composition service needs $N$ service classes, each service class contains $l$ candidate services. In the worst case, time complexity of algorithm Skyline is $O(Nl^2)$, time complexity of algorithm MCSPi is $O(Kl^2 + Kn^{l−1}) = O(KlN^{l−1})$, and time complexity of algorithm Relax is $K^2$. Consequently, the total time complexity of approach Sky-MCSP-R is $O(KlN^{l−1})$, and it is the same with original service composition approach MCSP-K. But for Sky-MCSP-R, after selecting Skyline services in the first phase, the number of candidate services in each service class must be less than $l$. Consequently, execution time of service composition must be promoted, and we will use simulation experiments to prove it in the next section.

**V. Experimental Analysis**

**Experimental setup.** In order to validate the efficiency and effectiveness of the proposed method Sky-MCSP-R, we will make a series of simulation tests and compare the composition efficiency and optimization rate between Sky-MCSP-R and MCSP-K [15] by using test cases as shown in Table I. 25 test cases are divided into 5 test groups, and each group has 5 test cases. For example, test cases of test group 2 are from 6 to 10. Test cases in the same test group have the same No. Candidates (number of candidate services) but different No. Service Class (number of Service Classes). For example, No. Candidates of each test case in Test Group 4 are all 40, but the No. Service Class ranges from 10 to 50. Test cases in different test group have different No. Candidates but with the same No. Service Class. For example, the No. Candidates of Test Case 13 in Test Group 3 and Test Case 23 in Test Group 5 are different, but the No. Service Class are both 30.

Because there are no standard test data sets of Web service composition, test data sets in the experiments are produced by the random function of Java program. We choose 5 QoS properties including delay, cost, reliability, credibility and availability whose value are all in the range [0, 100]. The QoS model we use are shown like formula (3) (4) (5), formula (3) is used to compute QoS value of each service, formula (4) is used to compute QoS value of each composite service, and the QoS value of composite service must satisfy formula (5). The experimental platform is on myeclipse, and environment of hardware are Intel(R) Pentium(R) CPU P6200 @ 2.13GHz, RAM 2G.

**Experimental results.** Table 3 compares the optimization rate between Sky-MCSP-K and MCSP-K in 25 test cases. The optimization rate is the QoS value ratio of our heuristic method with exhaustive method, and it reflects the performance of selected composition scheme of composition method. It is not hard to see that the optimization rate of proposed Sky-MCSP-K is slightly lower than method MCSP-K. Sky-MCSP-R could reach 90% optimization rate on average, compared to the average 91% optimization rate of MCSP-K, reducing the optimization rate by 1%.

Figure 2 and Figure 3 compare the execution time between Sky-MCSP-R and MCSP-K in 25 test cases. The number of user’s constraints are two ($cost < 300$, $credibility < 400$), no soft constraints are considered. The time ratio is less than 1 in all cases, which illustrates that execution time of Sky-MCSP-R is less than MCSP-K, and consequently Sky-MCSP-R has a higher composition performance. In fact, each service class can screen out 80% Skyline services on average when the number of QoS constraints are five. Eliminating 20% candidate services in candidate service space, and search efficiency is improved. Figure 2 shows test case of test cases in the same Test Group. We can see promotion of efficiency of Sky-MCSP-R increases with the number of service classes when the number of candidate services remain unchanged. When the number of service classes is small, the promotion of efficiency is limited. As the number of service classes increase, the promotion effect is obvious. The most ideal case is that execution time of Sky-MCSP-R is less than 30% of MCSP-K (Test Case = 20). Figure 3 shows test cases in different Test Group. We can see efficiency promotion of Sky-MCSP-R increases with the number of candidate services when number of service classes remain unchanged. But this is not always, sometimes efficiency promotion of Sky-MCSP-R decreases with the number of candidate services (compared Test Case = 9 to Test Case = 14). This is due to the reason that time cost of computing Skyline services also increase as the number of candidate services increase. Consequently, increasing the number of service classes more than increasing the number of candidate services can promote the efficiency of Sky-MCSP-R. But in any case, the time efficiency superiority of Sky-MCSP-R is obvious compared to MCSP-K. Although the optimization rate of MCSP-K is better than Sky-MCSP-R, it is worth to reduce optimization rate by 1% to on the price of significantly improved time efficiency.

In addition, when the number of user’s constraints is two ($cost < 300$, $credibility < 400$), no solution phenomenons arise in both methods. When the number of user’s constraints increase to five, there are 5 no solutions in 25 test cases by using MCSP-K. However, by using method Sky-MCSP-
TABLE I

| Test Case | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Test Group | 1  | 1  | 1  | 1  | 1  | 2  | 2  | 2  | 2  | 3  | 3  | 3  | 3  | 3  | 4  | 4  | 4  | 4  | 4  | 5  | 5  | 5  | 5  | 5  | 5  |
| No. Candidates | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 30 | 30 | 40 | 40 | 40 | 40 | 40 | 50 | 50 | 50 | 50 |
| No. Service Class | 10 | 20 | 30 | 40 | 50 | 10 | 20 | 30 | 40 | 50 | 10 | 20 | 30 | 40 | 50 | 50 | 10 | 20 | 30 | 40 | 50 | 50 | 50 | 50 |

TABLE II

<table>
<thead>
<tr>
<th>Test Case</th>
<th>MCSP-K</th>
<th>Sky-MCSP-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Group</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Optimization rate (%)</td>
<td>35%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Fig. 3. Execution time comparison in different Test Group (Sky-MCSP-R/MCSP-K).

R, five constraints are converted to two hard constraints (cost < 300, credibility < 400) and three soft constraints (time < 300, availability < 350, safety < 300), over-constraint problem does not happen, and no solution phenomenon is reduced.

VI. CONCLUSION AND FUTURE WORK

This paper proposes a new service composition named Sky-MCSP-R. This approach can select 80% Skyline services in high quality from candidate service space, reducing the scale of nodes of graph model. The algorithm MCSPi is used to compose services, and MCSPi can find as many service compositions which satisfies user’s demand as possible. Finally algorithm Relax is used to get the optimal solution. Sky-MCSP-R can effectively solve the optimization problem of Web service composition which satisfies users multiple QoS constraints, achieving a balance between time performance and optimization rate.

In the future, we plan to propose a Skyline service maintenance algorithm. When the number of candidate services increases, maintenance algorithm only expands original Skyline services rather than re-running Skyline algorithm for Skyline services.

VII. ACKNOWLEDGEMENTS

The work is supported by the National Natural Science Foundation of China under Grant (Nos. 61202097 and 61202136 and 61370091), China Postdoctoral Science Foundation (Grant Nos. 2012T50489 and 2011M500897) and Doctoral Fund of Ministry of Education of China (Grant No.20120094120009).

REFERENCES