Data-intensive Service Mashup Based on Game Theory and Hybrid Fireworks Optimization Algorithm in the Cloud

Wanchun Yang
School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
School of Sciences, Shandong Jiaotong University, Jinan 250357, China
E-mail:yangwch1982@126.com

Chenxi Zhang* and Bin Mu
School of Software Engineering, Tongji University, Shanghai 201804, China
E-mail: chenxizhang10@126.com, binmu@tongji.edu.cn
*Corresponding author

Keywords: cloud computing, data-intensive, mashup, hybrid fireworks optimization algorithm, game theory, service correlation

Received: June 2, 2015

End users can create kinds of mashups which combine various data-intensive services to form new services. The challenging issue of data-intensive service mashup is how to find service from a great deal of candidate services while satisfying SLAs. In this paper, Service-Level Agreement (SLA) consists of two parts, which are SLA-Q and SLA-T. SLA-Q (SLA-T) indicates the end-to-end QoS (transactional) requirements. SLA-aware service mashup problem is known as NP-hard, which takes a significant amount of time to find optimal solutions. The service correlation also exists in data-intensive service mashup problem. In this paper, the service correlation includes the functional correlation and QoS correlation. For efficiently solving the data-intensive service mashup problem with service correlation, we propose an approach GTHFOA-DSMSC (Data-intensive Service Mashup with Service Correlation based on Game Theory and Hybrid Fireworks Optimization Algorithm) which evolves a set of solutions to the Pareto optimal front. The experimental tests demonstrate the effectiveness of the algorithm.

Povzetek: Razvit je nov algoritem za reševanje NP težkega problema prepletanja storitev v oblaku.

1 Introduction

With the rapid development of cloud computing, the number of data-intensive services has increased dramatically. Data-intensive services are defined as the services whose inputs are large data sets. Users can create various mashups which combine data-intensive services to form new value-added services [1-2]. Data-intensive service mashup has become an important type of application in the field of Big Data. The challenging problem of data-intensive service mashup is how to find service from a large number of candidate services while satisfying SLAs. A service-level agreement (SLA) is defined upon a mashup as its end-to-end requirements [3]. A large number of research works such as [4-6], solve the SLA-aware service selection problem by leveraging linear programming. However, integer linear programming is only suitable for small size problems and suffers from high computational costs.

In order to solve the problem of high computational costs, numerous approaches have been studied. Alrifai et al. [7] proposed a hybrid solution that combines global optimization with local selection. The approach first adopted mixed integer programming to get the optimal decomposition of end-to-end QoS constraints, and then performed efficient local selection to get the best services which satisfy the local QoS constraints. Compared with the integer linear programming, the hybrid solution performs better in the time efficiency while achieving close-to-optimal results. The skyline technique has been used to reduce the number of candidate services. Instead of considering all services of each service class, a large number of efficient algorithms speed up the service selection process and discover the optimal composite service from a reduced solution space [8-11].

A large number of research works leverage heuristic algorithms to solve the service selection problem. In [12], an approach based on genetic algorithm was presented where the solution is encoded as a chromosome. However, there are some shortcomings in classical genetic algorithm, such as premature phenomena. To overcome the flaws of genetic algorithm, a number of improved genetic algorithms have been used to find the sub-optimal solution [13-18]. In [18], an improved genetic algorithm was proposed for SLA-aware service selection problem which results with higher fitness values. The simulated annealing and harmony search were used as mutation operator in

the improved algorithm. Compared with genetic algorithm, other heuristic algorithms can also achieve better optimality. Wang et al.[19] combined an approximation approach with artificial bee colony to solve the QoS-aware service selection problem. In [20], an improved immune optimization algorithm based on PSO was presented for QoS-aware service selection with end-to-end QoS constraints. In [21], an effective service selection approach with global QoS constraints based on particle swarm optimization algorithm was proposed.

To our best knowledge, there are only a small number of works which concern SLA from transactional risk. In [22], a survey of SLA assurance was conducted in the cloud. Haddad et al. [23] proposed a series of construction and processing rules to obtain a transactional mashup. However, the approach cannot gain global optimality and cannot support SLA-aware service selection. Wu et al.[24] combined the transaction properties into QoS-aware service selection and presented an approach based on ant colony algorithm.Compared with [23], the approach shows better performance in efficiency.

Many approaches regard the service mashup problem as a single objective optimization problem. To obtain multiple Pareto-optimal solutions, a few researches adopt multi-objective genetic algorithms to solve the problem [25-26]. In [25], a multi-objective optimization framework for SLA-aware service selection was presented. By leveraging multi-objective genetic algorithm, the framework can produce a set of Pareto-optimal solutions effectively. In [26], the authors improved a multi-objective optimization algorithm which applies background knowledge to find QoS-optimized service selection. In [27], an effective multi-objective approach was presented to solve QoS-aware service selection with conflicting objectives and diverse constraints on quality matrices. In [28], a hybrid multi-objective particle swarm optimization algorithm was proposed for SLA-aware service selection problem. Fireworks optimization algorithm (FOA) [29-30] is a relatively new heuristic method inspired by the phenomenon of fireworks explosion. As far as we know, there is still no research on the application of fireworks optimization algorithm in multi-objective service selection problem.

The contributions which distinguish our work from the above researches can be summarized as follows: 1. the problem of SLA-aware data-intensive service mashup with service correlation is formulated; 2. the SLA is divided into two aspects which are SLA-Q and SLA-T; 3. the service correlation is composed of two parts which are functional correlation and QoS correlation; 4. an approach GTHFOA-DSMSC (Data-intensive Service Mashup with Service Correlation based on Game Theory and Hybrid Fireworks Optimization Algorithm) which evolves a set of solutions to the Pareto optimal front is presented.

The remainder of this paper is organized as follows: Section 2 briefly presents the framework for data-intensive service mashup. Section 3 introduces the multi-objective optimization model, while Section 4 presents an approach based on Game Theory and Hybrid Fireworks Optimization Algorithm. The analysis of simulated results is done in Section 5. Finally, Section 6 concludes our work.

2 Framework for data-intensive service mashup

Figure 1 demonstrates the data-intensive service mashup process in the cloud. The framework is composed of three main components: 1) Planner component; 2) Generator component; 3) Execution Engine.

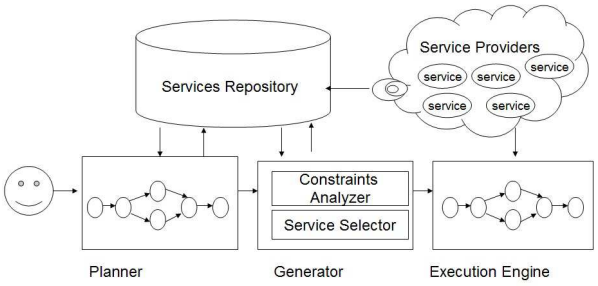

Figure 1: Framework for data-intensive service mashup.

2.1 Planner component

The component receives the request to generate an abstract mashup. An abstract mashup specifies the execution sequences among the activities. Each activity is an abstract service(AS) which corresponds to a candidate services set.

2.2 Generator component

Given a certain abstract mashup, in addition to a set of concrete services that can implement the activities from the cloud provider, the generator component can decide on what concrete services to include in the mashup. This component consists of two sub-components:1) Constraints Analyzer; 2) Service Selector.

Constraints Analyzer component receives the end-to-end constraints and generates a score based on constraints. Service Selector component selects services to construct the concrete mashup. The concrete mashup is composed of a set of concrete services from the candidate services. For an abstract mashup with n abstract services and l candidate services in each abstract service, there are l^n concrete mashups to be evaluated. A concrete service is represented by a tuple denoted as $< N, I, O, T, Q >$. Following is the detail description of the tuple.

- N is the name of a service.
- I=$\{I_1,...,I_n\}$ is a set of inputs which are required when performing the service.

– O={O_1,...,O_n} is a set of outputs that will be acquired after completing the service.

– $Transactional\ properties(T)$ guarantee the failure atomicity during execution.

– $QoS(Q)$ presents the quality of service which can be used to assess a service.

2.3 Execution engine

The concrete mashup is sent to execution engine to be executed. The execution engine is responsible for coordinating the execution of the components in the most effective way.

3 Multi-objective optimization model

3.1 QoS attributes

QoS attributes are introduced to describe non-functional properties of data-intensive services. They are used to differentiate the services providing the same functionality during the service selection process. In this paper, four most popular QoS attributes are considered: execution cost(C), response time(T), availability(A) and reliability(R).

– Execution cost : the cost that a service requester has to pay for the service invocation.

– Response time : the time interval between when the service is invoked and when the result is obtained.

– Availability : the probability that the service is accessible.

– Reliability : the probability that a request is correctly responded.

3.2 Normalization of attribute value

Due to the diverse measurement metrics of QoS attributes, attribute values should be normalized. For positive attributes, higher value indicates better quality (e.g. availability and reliability), which are normalized as equation (1). For negative attributes, lower value indicates better quality (e.g. cost and response time), which are normalized as equation (2).Q_i^{max} andQ_i^{min} are the maximal and minimal attribute values among all services, respectively. $Q_i^{'}$ refers to the attribute value of Q_i after normalization.

$$Q_i^{'} = \begin{cases} \frac{Q_i - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & Q_i^{max} - Q_i^{min} > 0 \\ 1 & Q_i^{max} - Q_i^{min} = 0 \end{cases} \quad (1)$$

$$Q_i^{'} = \begin{cases} \frac{Q_i^{max} - Q_i}{Q_i^{max} - Q_i^{min}} & Q_i^{max} - Q_i^{min} > 0 \\ 1 & Q_i^{max} - Q_i^{min} = 0 \end{cases} \quad (2)$$

3.3 QoS computation of mashup

A mashup can be constructed from several services in different structures. There are four basic structures: sequential, parallel, branch, and loop structures. Figure 2 shows the four structures.

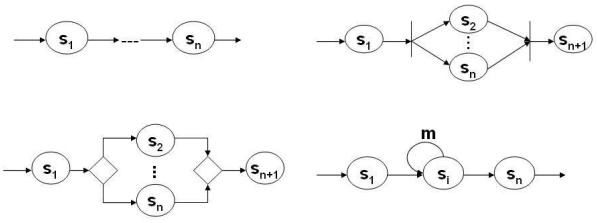

Figure 2: Four basic structures.

This paper computes the QoS of mashup according to the equations in table 1.For an additive property (e.g. cost and response time), we should compute the value through add operation. For a multiplicative property (e.g. availability and reliability), the value should be determined by multiply operation.

	Sequential	Parallel	branch	Loop
C	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} p_i c_i$	$k*c$
T	$\sum_{i=1}^{n} t_i$	$\max[t_i]$	$\sum_{i=1}^{n} p_i t_i$	$k*t$
A	$\prod_{i=1}^{n} a_i$	$\prod_{i=1}^{n} a_i$	$\prod_{i=1}^{n} p_i a_i$	a^k
R	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} p_i r_i$	r^k

Table 1: QoS aggregation functions.

3.4 Transactional properties

The transactional properties of services that we consider in this paper are pivot, compensatable, retriable and their combination. To obtain a transactional mashup, the rules are proposed in [23]. The transactional property of mashup $Tp(M) \in Tpset$, $Tpset$={p,c,r,cr}.

– $TP(M) = p$. Once all services of mashup execute successfully, the effects cannot be undone.

– $TP(M) = c$. Mashup is able to recover its effects even if it is executed successfully.

– $TP(M) = r$. Mashup will execute repeatedly until it is successful.

– $TP(M) = cr$. Mashup is both compensatable and retriable.

Table 2 and 3 represent the rules, where row heading indicates the transactional property of the first service, column heading indicates the transactional property of the second service. The value in each table cell represents the transactional property of mashup. $''-''$ denotes the mashup does not satisfy atomic consistency.

	p	c	r	cr
p	-	-	p	p
c	p	c	p	c
r	-	-	r	r
cr	p	c	r	cr

Table 2: Transactional rules for sequential construct.

	p	c	r	cr
p	-	-	-	p
c	-	c	-	c
r	-	-	r	r
cr	p	c	r	cr

Table 3: Transactional rules for parallel construct.

3.5 Service level agreement (SLA)

We assume that the end user has more SLA requirements with regard to the QoS values and transactional properties of the requested mashup. For a given abstract mashup, we consider a selection as a feasible selection, if it contains exactly one service for each service class and satisfies the end-to-end QoS (transactional) requirements.

3.6 Service correlation

- Functional Correlation: some concrete services are functional dependent on each other. The functional correlation $FC(S_{im}, S_{jn})$ indicates if the abstract service S_i selects the m^{th} concrete service, then the n^{th} concrete service should be selected for abstract service S_j .
- QoS Correlation: the QoS values delivered by a service in a mashup may vary according to the other services selected. $QC_a(S_{im}, S_{jn})$ indicates a QoS correlation between S_{im} and S_{jn}, regarding a QoS attribute a.

3.7 Multi-objective optimization model

In this paper, we take T, C and R as three objective functions for the sake of simplicity. A model of multi-objective service mashup can be formalized as follows:

Minimum(T(M),C(M),-R(M))
$s.t.$
$(1)A(M) > A_0$
$(2)T(M) < T_0$
$(3)C(M) < C_0$
$(4)Tp(M) \in$ {p,c,r,cr}
(5)Correlation constraints are satisfied.

Where T(M),C(M),A(M) and R(M) represent QoS attributes of mashup. A_0 ,T_0 and C_0 are the constraints to availability, time and cost respectively. The goal is to make the objective functions to be minimized simultaneously.

4 Service mashup based on game theory and hybrid fireworks optimization algorithm

4.1 Game theory

During the game, a problem is divided into several simpler problems according to the number of players. Each player seeks the best strategy in order to improve its objective criterion. As soon as no players can improve its objective value by adjusting its own best strategy, the goal is reached.

4.2 Fireworks optimization algorithm

Fireworks Optimization Algorithm (FOA), a novel heuristic algorithm, is implemented by simulating the fireworks explosion. At each iteration, the algorithm selects some quality locations as fireworks, which generate many sparks to search the local area. The algorithm continues until optimal location is found, or the termination condition is satisfied. The number of sparks and the amplitude generated by each firework are respectively defined such that:

$$s_i = m \times \frac{f_{max} - f(x_i) + \alpha}{\sum_{i=1}^{n}(f_{max} - f(x_i)) + \alpha} \tag{3}$$

$$A_i = A \times \frac{f(x_i) - f_{min} + \alpha}{\sum_{i=1}^{n}(f(x_i) - f_{min}) + \alpha} \tag{4}$$

Where m and A are control parameters, n is the size of the population, f_{max} and f_{min} indicate the maximum and minimum object values among the n fireworks respectively, and α is a small constant. To avoid overwhelming effects of splendid fireworks, bounds are defined for s_i as follows:

$$s_i^{'} = \begin{cases} round(\beta \times m) & \text{if } s_i < \beta m \\ round(\delta \times m) & \text{if } s_i > \delta m \\ round(s_i) & otherwise \end{cases} \tag{5}$$

Where β and δ are constant parameters. The location of each spark x_j generated by x_i can be calculated by equation (6):

$$x_j^d = x_i^d + A_i \times rand(-1, 1) \tag{6}$$

If the obtained location falls out of the search area, we should map it to the search area as follows:

$$x_j^d = x_{min}^d + x_j^d mod(x_{max}^d - x_{min}^d) \quad (7)$$

In the algorithm, the current best location is always selected as a firework of the next explosion iteration. Afterwards, n-1 locations are selected according to their distances to other locations.

4.3 Fitness assignment

In this paper, we adopt the notion of domination value to compute the fitness. A solution x_i is said to dominate a solution x_j in three cases:

- The solution x_i is feasible, and x_j violates some constraints.
- Both x_i and x_j are feasible, and x_i dominates x_j in terms of their object values.
- Both x_i and x_j violate constraints, and x_i dominates x_j in terms of their SLA violations.

In the GTHFOA-DSMSC, the fitness value of x_i is determined:

$$f(x_i) = \sum_{x_j > x_i} s(x_j) + \frac{1}{d(x_i)} \quad (8)$$

Where $d(x_i)$ is the distance from x_i to its nearest solution. The strength value $s(x_j)$ is computed according to the number of other solutions it dominates.

$$s(x_j) = |\{x_k \in P \cup NP | x_j > x_k\}| \quad (9)$$

4.4 Coding strategy

As illustrated in figure 3, the firework is encoded as an integer array of n elements: AS_1,AS_2, ..., AS_n and the value of AS_i ranges from 1 to m . Where n is the number of activities in the abstract mashup and m is the number of candidate services for each of the activity.

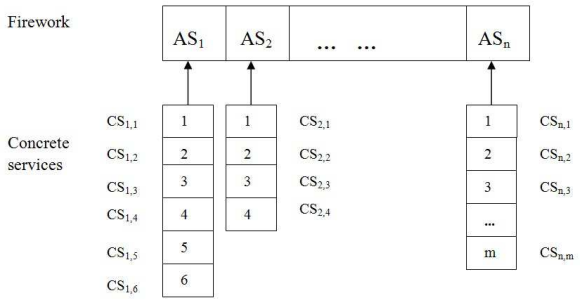

Figure 3: Coding strategy.

4.5 Crossover and mutation operators

The crossover and mutation operators are incorporated into the FOA to improve the performance. If crossover occurs at certain position, solutions in pairs swap their values at that position and the resulting solutions are used as offspring. The mutation operator is done by randomly selecting a position in a parent solution and randomly choosing a new concrete service to replace the one at that position.

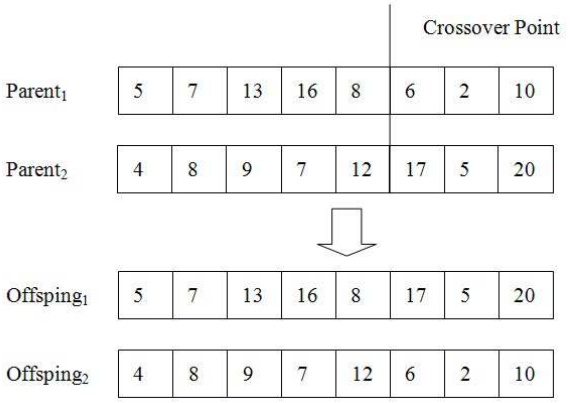

Figure 4: Crossover operator.

4.6 Non-dominated archive controller

The function of the archive controller is to determine whether a solution should be inserted into the external archive. The size of external archive may increase quickly, and thus it is required to limit the size of archive. If the external archive is empty, then the current solution is added into the archive. If the new solution is dominated by an individual within the archive, then the solution is discarded. If none of the individuals included in the archive dominates the new solution, then the solution is inserted in the archive. If there are individuals in the archive which are dominated by the new solution, then the individuals are removed from the external archive. Lastly, when the external archive reaches the size limit, the approach proposed by [31] is invoked.

4.7 Hypervolume (HV)

The HV of a solution set w signifies the hypervolume in the objective space that is dominated by w . In Figure 5, the solution w_5 is dominated by the solution set {w_1, w_2, w_3 ,w_4}. In this paper, we combine the non-dominated fronts of several algorithms into a maximum front w_{max} . The HV ratio of w is calculated by equation (10):

$$HVRatio = \frac{HV(w)}{HV(w_{max})} \quad (10)$$

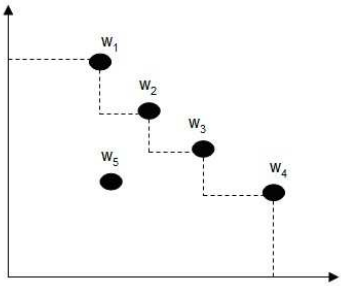

Figure 5: The hypervolume of the solution set $\{w_1, w_2, w_3, w_4\}$.

4.8 Algorithm design of GTHFOA-DSMSC

The algorithm description is as follows:

step1 Randomly generate a population P of n feasible solutions; create the empty external archive, and select non-dominated solutions from P to update the archive. Each player optimizes only his object.

step2 If the termination criteria is satisfied, goto Step5; else continue;

step3 Compute s_i and A_i for each individual x_i in P according to equations (3) (4)and (5); generate sparks of x_i according to equations (6) and (7); compute fitness for all sparks according to equations (8) and (9); select n solutions from the fireworks and sparks, use the crossover and mutation operations on the n solutions. Each player obtains his best solution respectively. Update the archive based on the new solutions.

step4 Update P by including the best solution and other $n-1$ ones selected based on their distance to other locations, goto Step2.

step5 Get the solutions and stop the algorithm.

5 Experiment and analysis

In this section, we conduct experiments to assess the performance of the proposed algorithm on a PC with Pentium 2.0GHz processor, 4.0GB of RAM and Windows7. In the experiment, the QoS attributes of the services are randomly generated expect for the cost, which is partially anti-correlated to the other QoS attributes. The algorithms optimize three objectives and have to meet constraints. The transactional property of each service is selected from $\{p,c,r,cr\}$ randomly. The percentage Θ indicates the strength of end-to-end QoS constraints. For positive QoS attributes, Θ is calculated by equation (11). For negative QoS attributes, Θ is calculated by equation (12).

$$\Theta = \frac{c_i - q_i^{min}}{q_i^{max} - q_i^{min}} \tag{11}$$

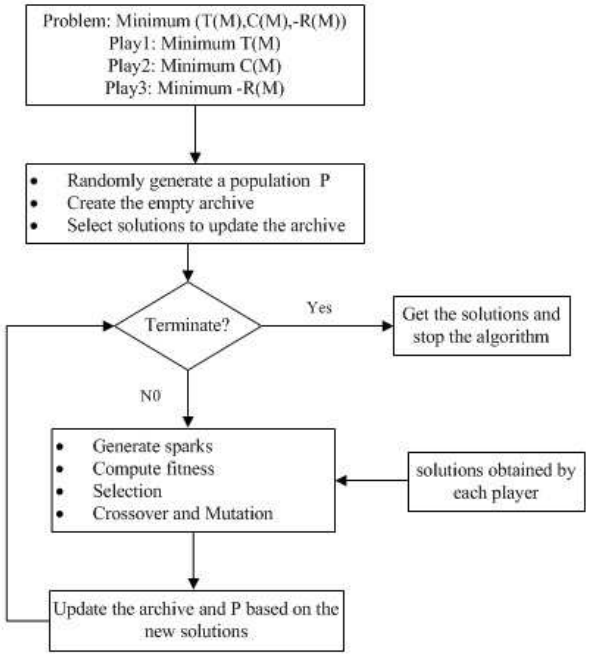

Figure 6: The flowchart of GTHFOA-DSMSC.

$$\Theta = \frac{q_i^{max} - c_i}{q_i^{max} - q_i^{min}} \tag{12}$$

Where c_i is the i^{th} QoS constraint, q_i^{max} and q_i^{min} indicate the maximum and minimum aggregated values of the i^{th} QoS attribute of the mashup. For all the algorithms, the upper limit of the archive size is set to 20. The population size is set to 30 for the GTHFOA, 200 for the NSGA-II [32] and GDE [33]. We evaluate every test case 100 times and limit the runtime of each algorithm to 30s.

5.1 Performance vs problem size

We investigate the performance of GTHFOA with the problem size. In figure 7, the number of abstract services is fixed 8. The number of concrete service candidates increases from 40 to 120 with the step 20. In figure 8, the number of abstract services increases from 4 to 12 with the step 2. The number of concrete service candidates is fixed 100. As the figures illustrate, GTHFOA is able to achieve above 90% in average. NSGA-II and GDE have decreasing HV ratio with the increase of the problem size. Through the experiment, we conclude GTHFOA can efficiently escape from the local optima and guide the search towards the Pareto fronts. The exploration strategies used by the other heuristic algorithms are not sufficient for achieving a good approximation of the Pareto-front in the solution space.

5.2 Performance vs strength of constraints

We investigate the performance of GTHFOA with the strength of constraints. In figure9, the strength of con-

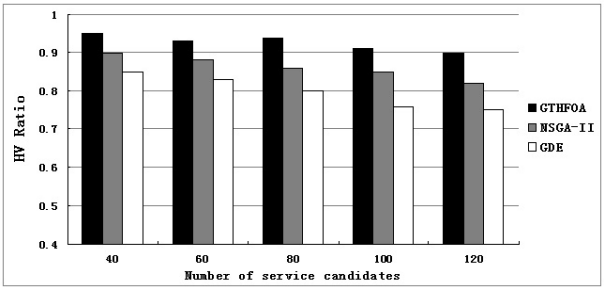

Figure 7: Performance vs service candidates.

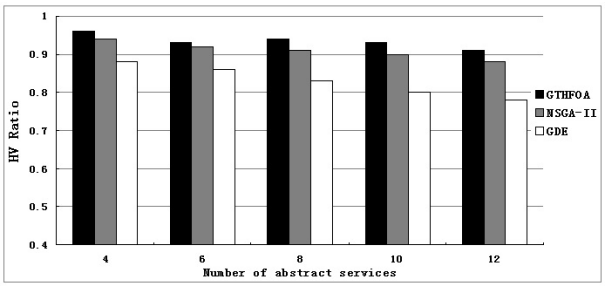

Figure 8: Performance vs abstract services.

straints increases from 0.2 to 0.6 with the step 0.1. With the increase of the strength of constrains, the existence probability for a feasible solution which satisfy all constraints declines. In figure9, GTHFOA has a constant HV ratio above 90%. If the strength of constraints is fixed 0.6, GTHFOA can gain HV ratio 91 %, while NSGA-II 85% and GDE 68%. As can be seen, GTHFOA shows better performance over NSGA-II and GDE. GTHFOA has a higher probability of reaching the Pareto-optimal front than other optimization algorithms.

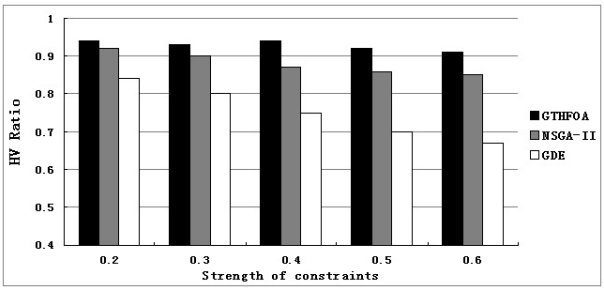

Figure 9: Performance vs strength of constraints.

5.3 Performance vs convergence

This part shows the convergence of GTHFOA. We fix the number of abstract services 8 and the number of service candidates 100. The strength of constraints is fixed 0.4. In figure 10, the number of iterations increases from 0 to 400 with the step 100. As can be seen, the convergence of GTHFOA is approximately 95%, while NSGA-II 90% and GDE 75%. The performance of GTHFOA overwhelms the other two algorithms. The GDE has the lowest performance among the algorithms, which indicates that its search capability is very limited. By comparison, we find GTHFOA converges fast and can converge to a higher value.

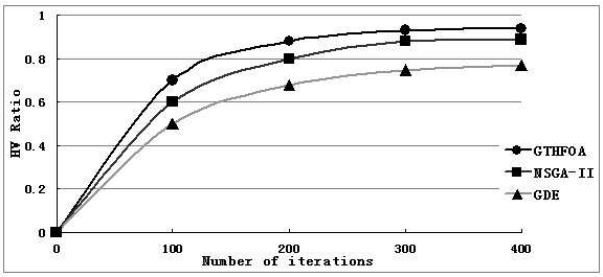

Figure 10: Performance vs convergence.

5.4 Performance vs service correlation

This part shows the performance of GTHFOA with service correlation. The approach GTHFOA-withoutQC does not consider the service correlation. We fix the number of abstract services 8 and the number of service candidates 100. The strength of constraints is fixed 0.4. We evaluate the ratio of the results of the two approaches. The ratio can be calculated by

$$ratio = \frac{GTHFOA - withoutQC}{GTHFOA} \tag{13}$$

In figure 11, the number of service correlation increases from 20 to100 with the step 20. As can be seen, GTHFOA-withoutQC does not consider the correlation, so it cannot reach the Pareto-optimal front. With the number of service correlation increases, the ratio declines. That is because the QoS deviation happened during the service selection.

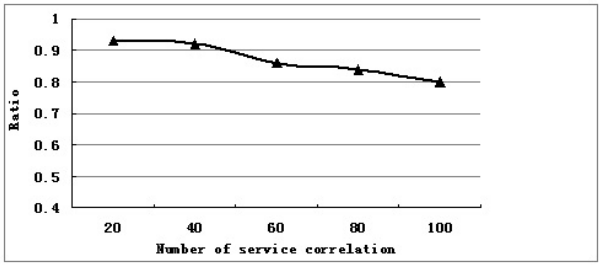

Figure 11: Performance vs service correlation.

6 Conclusion

This paper solves the data-intensive service mashup from QoS and transactional dimensions. The Service-Level

Agreement (SLA) consists of two parts, which are SLA-Q and SLA-T. In this paper, we consider the service correlation which includes the functional correlation and QoS correlation. In order to solve the service mashup problem efficiently, we propose an approach based on game theory and hybrid fireworks optimization algorithm which evolves a set of solutions to the Pareto optimal front. In our future work, we will further improve the performance of multi-objective evolution algorithm to solve the multi-objective service selection problem. The problem of runtime service process reconfiguration is also left for future research.

Acknowledgement

The research is supported by the Fundamental Research Funds for the Central Universities, the Natural Science Foundation of Shandong Jiaotong University (Z201342) and National Key Basic Research Program of China (2010CB328106).

References

[1] A. Bouguettaya, S. Nepal, W. Sherchan, X. Zhou, J. Wu, S. Chen, L. Liu, H. Wang and X. Liu(2010). End-to-End Service Support for Mashups,*IEEE Transactions on Service Computing*, 3(3), pp. 250-263.

[2] A. Ngu, M. Carlson, Q. Sheng and Hye-young Paik(2010). Semantic-Based Mashup of Composite Applications,*IEEE Transactions on Service Computing*, 3(1), pp. 2-15.

[3] F.H. Zulkernine and P. Martin(2011). An Adaptive and Intelligent SLA Negotiation System for Web Services,*IEEE Transactions on Service Computing*, 4(1), pp. 1939-1374.

[4] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam and H. Chang(2004). QoS-aware Middleware for Web Services Composition,*IEEE Transactions on Software Engineering*, 30(5), pp. 311-327.

[5] D. Ardagna and B. Pernici(2007). Adaptive Service Composition in Flexible Processes,*IEEE Transactions on Software Engineering*, 33(6), pp. 369-384.

[6] T.Yu, Y.Zhang and K.J.Lin(2007). Efficient Algorithms for Web service selection with End-to-End QoS Constraints,*ACM Transactions on the Web*, 1(1), article 6.

[7] M. Alrifai, T. Risse and W. Nejdl(2012). A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints, *ACM Transactions on the Web* , 6(2), article 7.

[8] M. Alrifai, D. Skoutas and T. Risse(2010). Selecting Skyline Services for QoS-based Web Service Composition, *International Conference on World Wide Web*, Raleigh, North Carolina, pp. 11-20.

[9] K. Benouaret, D. Benslimane and A. Hadjali (2011). On the Use of Fuzzy Dominance for Computing Service Skyline Based on QoS,*IEEE International Conference on Web Services*, Washington,DC, pp. 540-547.

[10] K. Benouaret, D. Benslimane and A. Hadjali(2012). WS-Sky: An Efficient and Flexible Framework for QoS-Aware Web Service Selection, *IEEE International Conference on Services Computing*, Honolulu, HI, pp.146-153.

[11] Q. Yu and A. Bouguettaya(2013). Efficient Service Skyline Computation for Composite Service Selection, *IEEE Transactions on Knowledge and Data Engineering*, 25(4), pp. 776-789.

[12] G. Canfora, M. Penta, R.Esposito and M. Villani (2005). An Approach for QoS-aware Service Composition based on Genetic Algorithms,*7th annual conference on Genetic and evolutionary computation*, Washington,DC, pp.1069-1075.

[13] Y. Ma and C. Zhang(2008). Quick Convergence of Genetic Algorithm for QoS-driven Web Service Selection, *Computer Networks*, 52(5), pp. 1093-1104.

[14] Y. Syu, Y. FanJiang, J. Kuo and S. Ma(2012). A Genetic Algorithm with Prioritized Objective Functions for Service Composition, *International Conference on Advanced Information Networking and Applications Workshops*, Fukuoka, pp. 932-937.

[15] M. Chen and S. Ludwig(2012). Fuzzy-guided Genetic Algorithm applied to the Web Service Selection Problem,*IEEE World Congress on Computational Intelligence*, Brisbane, QLD, pp. 1-8.

[16] Y. Yu, H. Ma and M. Zhang(2013). An Adaptive Genetic Programming Approach to QoS-aware Web Services Composition, *IEEE World Congress on Evolutionary Computation*,Cancun,Mexico, pp. 1740-1747.

[17] F. Lecue and N. Mehandjiev(2011). Seeking Quality of Web Service Composition in a Semantic Dimension, *IEEE Transactions on Knowledge and Data Engineering*,23(6), pp. 942-959.

[18] A.E.Yilmaz and P. Karagoz (2014). Improved Genetic Algorithm based Approach for QoS Aware Web Service Composition, *IEEE International Conference on Web Services*,Anchorage, AK , pp. 463-470.

[19] X. Wang, Z. Wang and X. Xu(2013). An Improved Artificial Bee Colony Approach to QoS-Aware Service Selection, *IEEE International Conference on Web Services*, Santa Clara, CA, pp. 395-402.

[20] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng and Y. Fan(2012). An Improved Discrete Immune Optimization Algorithm based on PSO for QoS-driven Web

Service Composition, *Applied Soft Computing*, 12(8), pp. 2208-2216.

[21] G. Kang, J. Liu, M. Tang and Y. Xu (2012). An Effective Dynamic Web Service Selection Strategy with Global Optimal QoS Based on Particle Swarm Optimization Algorithm, *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Shanghai, China, pp.2280-2285.

[22] L. Sun, J. Singh and O. Hussain(2012).Service Level Agreement (SLA) Assurance for Cloud Services: A Survey from a Transactional Risk Perspective, *10th International Conference on Advances in Mobile Computing & Multimedia*, Bali, Indonesia, pp. 263-266.

[23] J.E. Haddad and G. Ramirez(2010). TQoS: Transactional and QoS-aware Selection Algorithm for Automatic Web Service Composition, *IEEE Transactions on Service Computing*, 3(1), pp. 73-85.

[24] Q. Wu and Q. Zhu(2013). Transactional and QoS-aware dynamic service composition based on ant colony optimization, *Future Generation Computer Systems*, 29(5), pp. 1112-1119.

[25] H.Wada, J. Suzuki, Y.Yamano and K. Oba(2012). E^3: A Multiobjective Optimization Framework for SLA-Aware Service Composition, *IEEE Transactions on Service Computing*, 5(3), pp. 358-372.

[26] F. Wagner, B. Klopper, F. Ishikawa and S. Honiden(2012). Towards Robust Service Compositions in the Context of Functionally Diverse Services,*International Conference on World Wide Web*, Lyon, France, pp. 969-978.

[27] A. Moustafa and M. Zhang(2013). Multi-Objective Service Composition Using Reinforcement Learning,*International Conference on Service-Oriented Computing*, Berlin, Germany, pp.298-312.

[28] H. Yin, C. Zhang, B. Zhang, R. Sun and T. Liu(2014). A Multi-Objective Discrete Particle Swarm Optimization Algorithm for SLA-Aware Service Composition Problem, *Acta Electronica Sinica*, 42(10), pp. 1983-1990.

[29] Y. Tan and Y. Zhu (2010). Fireworks Algorithm for Optimization, *International Conference on Swarm Intelligence*, Beijing, China, pp.355-364.

[30] Y. Pei, S. Zheng, Y. Tan and H. Takagi(2012). An Empirical Study on Influence of Approximation Approaches on Enhancing Fireworks Algorithm,*IEEE International Conference on Systems, Man, and Cybernetics*,Seoul, Korea, pp. 1322-1327.

[31] J.D.Knowles and D.W.Corne (2000). Approximating the Nondominated Front using the Pareto Archived Evolution Strategy,*Evolutionary Computation*, 8(2), pp. 149-172.

[32] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan (2002). A Fast and Elitist Multiobjective Genetic Algorithm:NSGA-II,*IEEE Transactions on Evolutionary Computation*, 6(2), 182-197.

[33] S. Kukkonen and J. Lampinen(2005). GDE3: The Third Evolution Step of Generalized Differential Evolution, *IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, pp. 443-450.