ISSN: 2442-6571

SLA based cloud service composition using genetic algorithm

N. Sasikaladevi

School of Computing, SASTRA University, India
sasikalade@gmail.com

ARTICLE INFO

Article history:
Received May 30, 2016
Revised June 17, 2016
Accepted July 26, 2016

Keywords:
Cloud computing
Quality of Service
Genetic algorithm
MMKP

ABSTRACT

Cloud computing tends to provide high quality on-demand services to the users. Numerous services are evolving today. Functionally similar services are having different non-functional properties such as reliability, availability, accessibility, response time and cost. A single service is inadequate for constructing the business process. Such business process is modeled as composite service. Composite service consists of several atomic services connected by workflow patterns. Selecting services for service composition with the constraints specified in Service Level Agreement is the NP-hard problem. Such a cloud service composition problem is modeled in this paper. Genetic based cloud service composition algorithm (GCSC) is proposed. Proposed algorithm is compared with the existing genetic based cloud service composition algorithm based on average utility rate and convergence time. It is proved that the proposed algorithm provides better performance as compared to the existing cloud service composition algorithm.

I. Introduction

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly provisioned and releases with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models” [1] .The characteristics of cloud computing are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service, multi-tenacity, audit ability and verifiability [2,3,4]. Cloud computing service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Cloud Computing deployment models are public cloud, private cloud, community cloud and hybrid cloud.

More and more cloud services are evolving today to meet the customer requirements. Each cloud service is defined with functional and non-functional properties. Functionally similar service may have different non-functional properties. It is not possible to define complex business process as a simple atomic service. Those business processes are defined by the composition of several atomic services. Atomic services are connected by using different workflow patterns. The task of connecting different atomic services to form the business process is known as Service Composition (SC). Whenever user submits the request, he also mentions his Quality of Service (QoS) requirements in Service Level Agreement (SLA). Each service has different QoS characteristics satisfying the QoS requirement on composite service is a tedious task. There is a need to compose the service with the QoS requirements mentioned by the consumer in the SLA. This QoS enforced service composition is the optimization problem. Picking up the appropriate services from the service pool and connecting together by using different workflow patterns by satisfying the functional requirement and QoS requirements based on the SLA is a complex problem. It is considered as a NP-Hard problem[5].

In this paper, the SLA based cloud service composition problem is solved by using genetic approach. A genetic based service composition algorithm is proposed. And it is implemented and compared with the existing service composition algorithm. It is proved that the proposed algorithm provides better performance in terms of time and optimality as compared to well known service composition algorithm. Rest of the paper is organized as; section 2 provides the details of existing service composition frameworks. Section 3 elucidates the SLA based service composition problem. Section 4 provides the proposed algorithm. Proposed algorithm is compared with existing service composition algorithm in Section 5. Section 6 provides the conclusion and future enhancements.

II. Related Works

Cloud service composition problem is handled differently by different researchers. Service competition is treated as an optimization problem [6,7] Traditional optimization methods such as backtracking and branch-and-bound methods are adopted to provide solution for this problem. Service composition is considered as a Multi dimensional multi choice knapsack problem [8]. Parallel form of branch and bound solution is provided by Preve [9]. SC is solved by using linear programming by [10,11]. Fast service compsotion methods are proposed by Liu by using two phase method. Liu proposed also proposed three layer hierarchical models for cloud service composition [12]. Three different algorithms for service composition is proposed by Qi et.al [13] for skyline service.

Numerous services are evolving today. Wide range of functionally similar services is available. Hence service composition problem need to search the optimal solution in the large search space. So, it can also be considered as a combinatorial optimization problem [14]. Neural network based solution is proposed by [15,16]. Jungman et. alproposed a solution based on markov decision process [17] . Ludwig et.al [18] proposed genetic algorithm solution for the SC problem. Yang proposed a solution based on game theory. Particle swarm optimization based solution is proposed by Wang [20]. Memetic algorithm for SC problem is proposed by Jula [21]. Machine language based techniques are proposed by different researchers. Bao et.al proposed solution for SC using Finite State [22].

Table 1 summarizes the related works on cloud service composition problem. In the service composition, satisfying the requirement of the consumer is important. All these methods are developed to improve the Quality of Service of the composed service. Among the available solution, only Yang 2012 [19] considered the SLA for service composition. In this paper, service composition problem is modeled as an optimization problem and it uses SLA for setting the QoS requirement for the composite service

Table 1. Comparison of various cloud service composition approaches

Authors	**Method adopted**	**QoS factors Considered**	**Tool used**	**Data Set used**
Wang et al.[16]	Neural Network back propagation algorithm	Response time, Cost, Throughput, Reputation, Availability, Reliability.	Matlab 7.6	WS-DREAM, RG
Ye et al. [24]	Genetic algorithm	Response time, Cost, Availability, Reputation		
Zhu et al. [25]	Mixed Integer programming		Visual C#.NET	
Liu et al. [12]	State transition matrix	Cost		WS-DREAM
Worm et al. [26]	Dynamic programming	Response time, Cost, Availability		
Ludwig [18]	Genetic algorithm	Response time, Cost, Availability, Reliability	Java	
Bao and Dou [22]	Finite state machine	Response time, Cost, Reliability, Availability, Reputation		
Sundareswaran et al. [27]	B+ tree index structure		C	
Chunqing et al. [28]	service conflict detection	Cost	Java, Cauldron, Chaff solver	
Xiaona et al. [29]	Trust based algorithm	Availability, Durability		Seekda
Zhang et al. [30]	Artificial neural network		JavaScript, RESTful	

Authors	Method adopted	QoS factors Considered	Tool used	Data Set used
Qi and Bouguettaya[13]	Graph based algorithms	Response time, Cost	Java	Using Synthetic Generator
Wang et al. [20]	Particle swarm optimization		Matlab 7.6, Lp-Solve 5.5	QWS, Synthetic Generator
Jula et al. [21]	Memetic algorithm	Response time, Cost	Visual C#.NET	WS-DREAM
Zhao et al. [24]	Genetic based negative selection algorithm	Response time, Cost, Availability, Reliability		
Dou et al. [30]	K-mean clustering algorithm	Cost, Latency, Reputation		
Karim et al. [31]	QoS mapping	Cost, Response Time, Security ,Reputation ,Availability, Reliability, Durability, Data Control		
Zibin et al. [32]	Ranking using similarity prediction	Response Time, Throughput	Planet-lab, Axis2	tpds 2012

III. SLA based Cloud Service Composition Problem

SLA is a legal document between the service provider and service consumer. SLA is used to ensure the Quality of service the consumer needs from the provider. It defines the functional requirements and non-functional requirements such as reliability, availability, Successability, accessibility etc. It also includes the tolerable response time and cost of the service [33].

SLA based cloud service composition problem is as follows: A business process involves several services and these services are connected by defined workflow patterns. In cloud computing paradigm, functionally similar services are available. Services are selected from the pool of services based on QoS requirements specified in the SLA is a SLA based Cloud Service Composition Problem (SCSC). SCSC problem is an optimization problem and it is NP-Hard. It is formulated as a multi-dimensional multi choice knapsack problem (MMKP)[34].

The reliability of a composite web service is decided by the reliability of individual services and their composition relationships. If every service involved in the composite web service is selected based on their reliability, other QoS parameters have an impact on it. When the reliability is maximized, QoS parameters such as Cost, response time and user preferences are also to be controlled. To construct an optimal composite web service, in addition to meet the reliability and QoS requirements, there is a need to define an objective function for optimization.

The multi-dimensional multi choice knapsack (MMKP) is reduces to solve the cloud service composition problem. This is modeled as linear integer problem:

- Service Classes: Business process is represented by a Collection of services. The collection of services for the business process is known as service classes.
- Service Candidate: A service candidate is a set of services with the same functional property but with different non-functional properties
- Utility Value: Utility value is calculated for each service candidate in each service class. There are 3 QoS attributes are considered. All these QoS attributes are to be maximized. The utility value for service candidate j from the service class i is calculated as,

$$U_{ij} = \sum_{l=1}^{3} \frac{Q_{ij}^{l} - \mu_l}{\sigma_l} * w_l \text{ where, } \sum_{l=1}^{2} w_l = 1 \quad (1)$$

W_l is calculated from SLA. W_1 refers to the weight factor for reliability, W_2 is the weight factor for Availability and W_3 is the weight factor for Successability. μ is the mean QoS for all the service candidates in a service class. σ is the standard devition for all service candidates in a service class. In the utility calculation 3 QoS attributes are weighted based on the user requirement specified in the SLA.

The notations used in this model are defined as follows:

- *S* is a service class. *A* service class is a collection of individual web services with a common functionality but different non-functional properties.
- *U* is the estimated utility vector of Service class *S*.
- *Q* is the QoS constraints (Response time and Cost) of the service Class *S*.
- *Qc* is the maximum allowed QoS Constraints for a business process.

The composite SSP is modeled as MMKP in the following way:

- The steps in composite web service represent the classes in MMKP.
- Every service in a service class has many candidates. Hence, every candidate represents an item in the class.
- The utility of candidates represents the profit of the item in MMKP.
- The response time and cost of the candidates represent the constraints for a MMKP.
- The objective is to maximize the total utility of the composite service under the constraints *Qc*.

The Problem is formulated as:

$$Max\ z = \sum_{i=1}^{n} \sum\nolimits_{j \in S_i} U_{ij} X_{ij} \tag{2}$$

Subject to $\sum_{i=1}^{n} \sum_{j \in S_i} Q_{ijk} x_{ij} \le Q_c$, where, $k = 1, 2$

$$\sum_{i=1}^{n} x_{ij} = 1,\ i = 1, 2, ..., n,\ j \in S_i \tag{3}$$

$x_{ij} \in \{0, 1\},\ i = 1, ..., n,\ j \in S_i$

The Cloud service composition problem is stated as follows: Given *n* service classes with each service class *i* containing *j* items. The j^{th} service candidate of service class *i* has Utility U_{ij}. Each service candidate has 2 QoS constraints denoted as Q_{ij}. The knapsack has capacity Q_c. The goal is to select one service candidate from each service class to maximize the sum of their utilities and to keep the total constraints not more than the corresponding capacity.

IV. Genetic based Cloud Service Composition Algorithm (GCSC)

A genetic algorithm (GA) is a probabilistic search algorithm used to solve various combinatorial optimization problems. It is based on the principle of "Survival of Fittest" and developed by Holland. Fittest individual will survive and reproduce whereas individual with less fit will be eliminated. GA simulates this process. Individuals in the population are referred as chromosomes and it forms the possible solution to the optimization problem. Fitness of the individual are calculated. Fittest individuals will reproduce using crossover operation. The result is the new offspring; which inherits the characteristics form their parents. Mutation operation is used to alter genes in the offspring. Offspring replaces the individual with lesser fitness in the population [35][36].

A. Fitness function

GCSC is developed to solve the SLA based cloud service composition problem. Population is coded using value encoding. Every individual (Service candidate) in the population is coded using structure. It contains the information about the service class to which it belongs to, its utility value, constraints such as response time and cost. Fitness function for GCSC is defined based on the objective and constraints of the problem.

$$f(x) = \sum_{i=1}^{n} \sum_{j \in S_i} U_{ij} X_{ij} \tag{4}$$

where U_{ij} is the utility value of jth service candidate in the ith service class. The f(x) is the maximization function. Constraints are specified as,

$$\sum_{i=1}^{n} \sum_{j \in S_i} Q_{ijk} x_{ij} \le Q_c \text{ , where, } k = 1, 2 \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ i = 1, 2, ..., n,\ j \in S_i$$

where Q_{ijk} is the QoS constraints value of jth service candidate in the ith service class. Q_c is the vector which contains the maximum tolerable response time and maximum allowed cost for the composite cloud service. If X_{ij} =1 then the jth service candidate in the ith service class is selected for inclusion in the solution, else jth service candidate in the ith service class is not selected for inclusion in the solution. Algorithm 1 shows the fitness function for GCSC.

Algorithm1: Fitness function $f(Pop(C))$

```
Calculate fitness= $\sum_{i=1}^{n} \sum_{j=1}^{m} U_{Gi_{Ij}}$ ,
where U- Utility value $G_i$ -Service Class, $I_j$- Service Candidate

//checking the constraints
Summed the constraints consumed by selected service candidates of each service class.

        if (Constraints consumed <= required constraints) then
                $Pop(C)$ is feasible
        else
                $Pop(C)$ is not feasible
        return fitness
```

B. Genetic operations

Selecting the parents for reproduction is the important task. Selection process reflects the convergence rate. GCSC problem should converge too quickly. Hence, selection operation is designed such a way that it improves the convergence rate. In this GCSC, a truncation based selection is coded. Population is sorted based on the fitness value. Algorithm 2 shows the selection function.

Algorithm 2: Truncation Selection function $\gamma(Pop(C))$

```
Input: population $Pop(\tau)$, the truncated threshold $T \in [0,1]$
Output: population after selection $Pop(\tau)'$

Sort the population $Pop(\tau)$ based on the fitness with worst fitness at the first position

For i=1 to N do
        $r$ =Random$\{[(1-T)N], \ldots N\}$
                $Pop(\tau)' = Pop(r)$
return $Pop(\tau)'$
```

In crossover operations, two parents from the current population are combined to produce offspring. Random value is selected between 1 to n-1 where n is the number of service classes. Single point cross over is used to create offspring. Then, the offspring's are checked against the feasible constraints. If the offspring are not feasible, then it is rejected. Algorithm 3 shows the crossover operation.

Algorithm 3: Crossover function δ($Pop(C, p1, p2)$)

```
Set any random value between 1, n-1 to cross over point $c_p$
For i=0 to $c_p$ do
{
        $O_1$=$Pop(C)_{p1}$
        $O_2$=$Pop(C)_{p2}$
}
For i=$c_p$ +1  to $n$ do
{
        $O_1$=$Pop(C)_{p2}$
        $O_2$=$Pop(C)_{p1}$
}
Calculate $f(O)$ and check the feasibility
```

Mutation function is based on the random seed. One service class is selected randomly from the population. Selected service candidate from the population is deselected. Some other service candidate is selected randomly. Algorithm 4 shows the mutation operation. Then, the fitness is calculated for the new population and feasibility constraints are checked. If the population is feasible, then the mutation function returns the selected population. The import component in designing the mutation operation is the mutation rate. Mutation rate highly affects the solution and convergence rate.

Algorithm 4: Mutation function β ($Pop(C)$)

```
Set the $m_r = 0.02$
Set the $m_l = Pop_{size} * m_r$
For i=0 to $m_l$ do
{
        Select one item $Pop_r$ form $Pop$ randomely
        Select one service class $S_r$ randomly from the $Pop_r$
        If $Pop(C)_{S_r}$ is selected then selected service candidates in $Pop(C)_{S_r}$ are deselected
        Else $Pop(C)_{S_r}$ is selected and one service candidate in each service class $Pop(C)_{S_r}$ is selected randomly
}
Calculate the $f(Pop(C))$ and Check the constraints
```

Too high mutation rate increases the probability of searching more areas in search space, however, prevents population to converge to any optimum solution. Alternatively, too low mutation rate may produce premature convergence. In other words; too high mutation rate reduces the search ability of GA. In GCSC mutation rate is set to 0.02 based on trial and error method.

The GCSC algorithm is shown in Algorithm 5. Generation counter is initialized to 0. Initial population is created using random seed. Algorithm5 terminates if it reaches the maximum number of generations. Another counter called No-Improvement is also defined in GCSC. In the each iteration, the fitness is calculated for the current population and if the fitness value is equal to the fitness of the previous solution, then No-Improvement counter is incremented. If there is no chance in the fitness value for continuous 10 generations, then the algorithm terminates even though the number of generations is less than the maximum generation.

Algorithm 5: Genetic based Cloud Service Composition (GCSC)

```
Initialize $C = 0$;
Generate initial population $Pop(C) = \{K_1, K_2, \ldots, K_n\}$ where, $K_i \in \{0,1\}$
Calculate initial fitness for $Pop(C) = \{f(K_1), f(K_2), \ldots f(K_n)\}$;
Search for the $K'$ with $f(K') \geq f(K_i)$ for $i = 1,2..n$
Initial fitness is $f(K')$
While ($C \leq MAXGEN$)
{
        Select $P_1, P_2$ by applying selection function $\gamma(Pop(C))$
        Uniform cross over on $P_1, P_2$ by applying the uniform crossover function
$\delta(Pop(C)$
        Mutate using $\beta$ ($Pop(C)$) to find offspring $O$ and calculate $f(O)$
      Improve the $O$ using mutate function for $l$ times.
        If $f(O) = any\ K \in Pop(C)$ then discard $C$; continue while;
      Calculate $f(O)$
      Search $K^{''} \in Pop(C)$ subject to
$f(K'') \leq f(K)$ for all $K \in Pop(C)$ $K^{''}$ and replace by $O$
        If ($f(O) > f(K^{''})$ then $K' = O$
         $C++;$
}
Return $K'$ and fitness value as $f(K')$
```

V. Experiment

In order to evaluate the GCSC, the simulation tool is created using Visual Studio.NET. GCSC algorithm is implemented using the language VC++. GCSC is evaluated based on optimal solution and time taken to provide the solution. Experiments are carried out in the System with Pentium Processor with 2.0 GB RAM, Windows 7 Operating System. Data are analyzed using Matlab R2010. Two different datasets are used for the experiments. One is QWS dataset. Another one is randomly generated dataset. In the experiments, GCSC is validated based on the optimality. GCSC is compared with the well known cloud service composition proposed by Wang [20] based on the optimal solution and time taken for convergence.

A. *GCSC Convergence time and Utility value*

Datasets are generated randomly using uniform distribution. Composite services with minimum number of atomic services 10 and maximum number of atomic services 30 are considered. Service classes refer to the group of functionally similar services. 100 to 1000 functionally similar services are considered for each and every service class. These services are refereed as service candidates. GCSC is analyzed based on the convergence time with varying number of service classes and service candidates. Fig 1 shows the convergence time of GCSC with different set of service candidates. It shows that convergence time increases slightly when the service candidates are increases. The convergence time slightly when the number of service classes is increases.

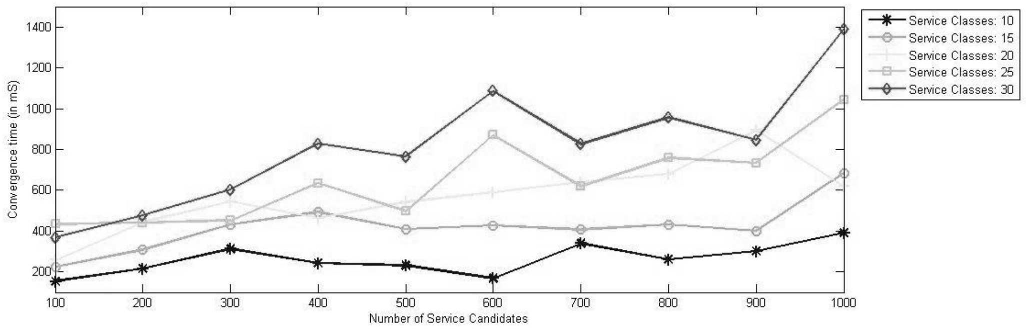

Fig. 1. Convergence time of GCSC with respect to number of service candidates

GCSC is analyzed based on the utility rate achieved. Fig. 2 and 3 shows the utility rate of GCSC with respect to different service candidates. Average utility rate with service classes 10 is higher than the average utility rate with service classes 30. GCSC provides the utility rate in the range of 95 to 99.

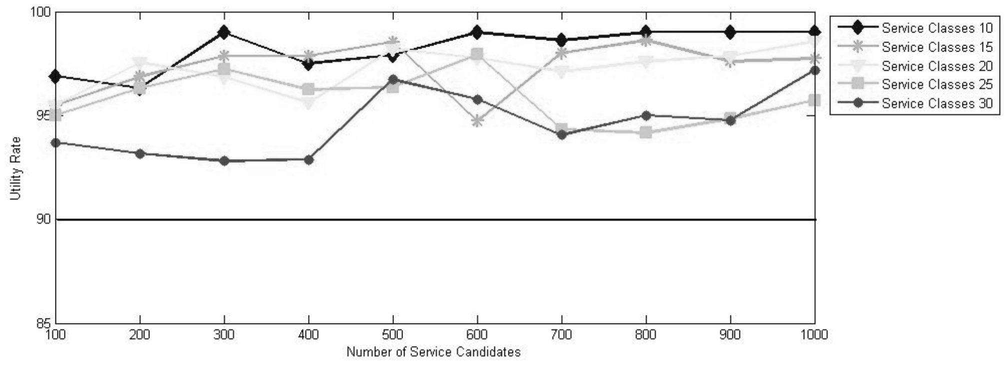

Fig. 2. Utility rate of GCSC with respect to number of service candidates

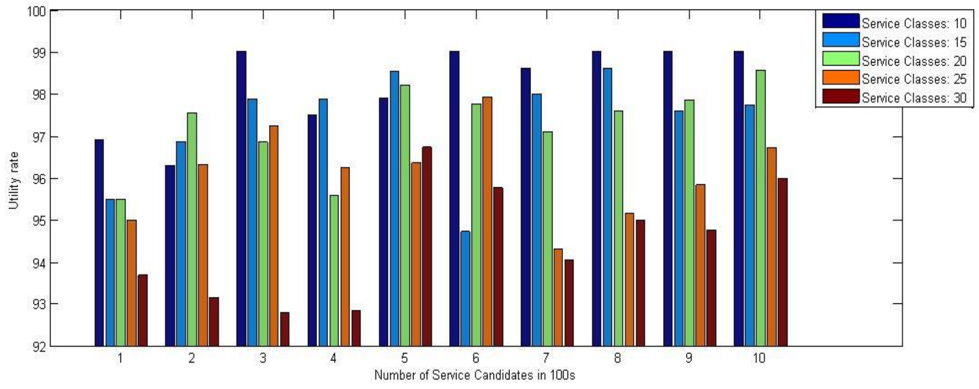

Fig. 3. Comparison of Utility rate of GCSC with respect to number of service candidates

Efficiency of the genetic algorithms is depends on the number of generations require to converge. Fig.4 shows the number of generations require for convergence. GCSC takes minimum number of generations for convergence. Maximum number of generations required for convergence for the service classes with 10 to 30 are 40.

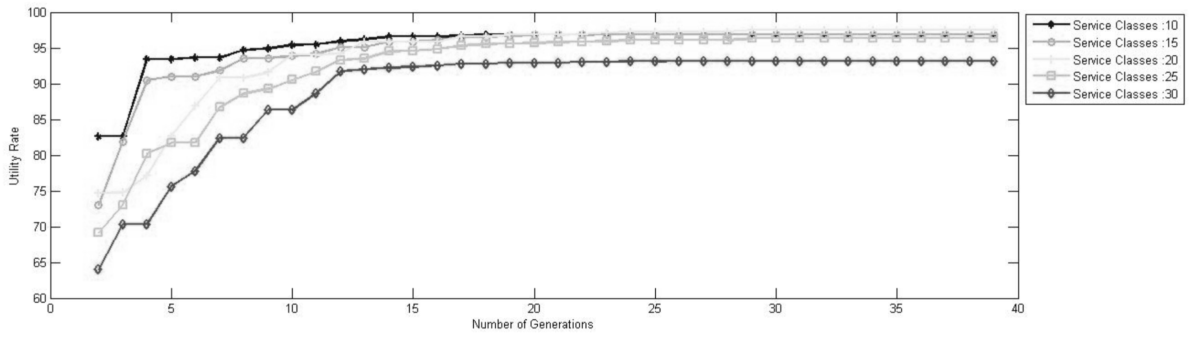

Fig. 4. Convergence time for service classes

B. *Performance Comparison between GCSC and FCSC*

GCSC is compared with the well known cloud service composition proposed by Wang based on the optimal solution and time taken for convergence [20]. Wang proposed the Fast cloud service composition algorithm (FCSC) based on particle swarm optimization. Fig. 5 shows GCSC is faster than FCSC with varying service candidates and service classes. It is proved that time GCSC converge quickly as compared to FCSC.

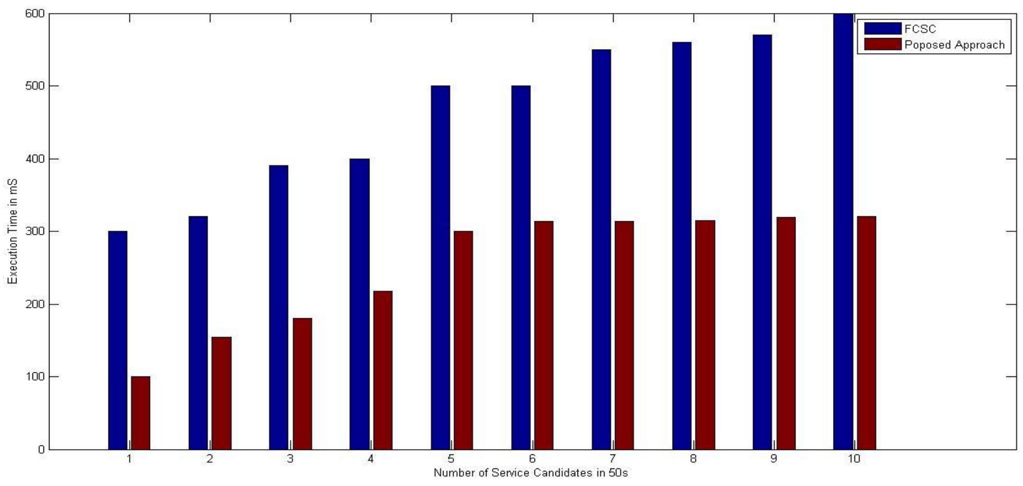

Fig. 5. Comparison of FCSC with GCSC based on Execution time

Fig. 6 shows that GCSC provides the better utility as compared to FCSC. It is true for varying service candidates and service classes. GCSC provides utility in the range of 96-99. FCSC provide utility in the range of 91-97.

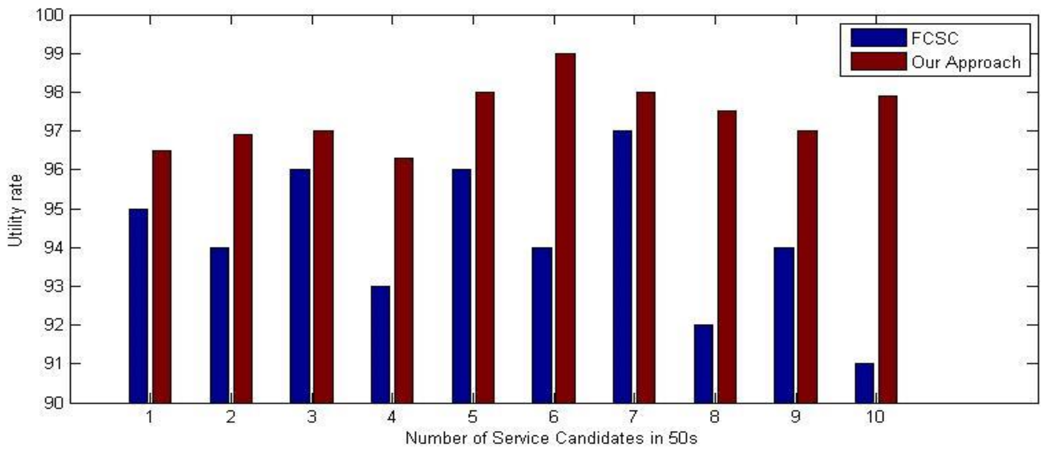

Fig. 6. Comparison of FCSC with GCSC based on Utility rate

VI. Conclusion

More and more cloud services are evolving today to meet the customer requirements. Each cloud service is defined with functional and non-functional properties. Functionally similar service may have different non-functional properties. Those business processes are defined by the composition of several atomic services. In this paper, SLA based cloud service composition problem is modeled. Genetic based cloud service composition algorithm is proposed. It is compared with fast cloud service composition algorithm FCSC. It is proved that GCSC provides better utility rate as compared to FCSC. Execution time for GCSC is 2 times lesser than FCSC. Business process consists of several atomic services. These services are connected by using different workflow pattern. In future, GCSC is further improved by considering the workflow patterns of the business process.

References

[1] Peter Mell, T. G,*The NIST Definition of Cloud Computing*. NIOS Technology, U.S. Department of Commerce, 2011.

[2] Hamdaqa, M., & Tahvildari, L. Cloud Computing Uncovered: A Research Landscape. *Advances in Computers,* 86, pp. 41–85,*Elsevier,* 2012.

[3] Yakimenko, O. A., Slegers, N. J., Bourakov, E. A., Hewgley, C. W., Bordetsky, A. B.,Jensen, R. P., Robinson, A. B., Malone, J. R., & Heidt, P. E., Mobile system for precise aero delivery with global reach network capability. *IEEE International Conference on Control and Automation*, 2009. ICCA 2009, pp. 1394–1398, 2009.

[4] Wischik, D., Handley, M., & Braun, M. B, The resource pooling principle, *SIGCOMM Computer Communication Review*, 38, 47–52, 2008

[5] Fei, T., Yuanjun, L., Lida, X., & Lin, Z, FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. *IEEE Transactions on Industrial Informatics*, 9, 2023–2033, 2013

[6] Anselmi, J., Ardagna, D., & Cremonesi, P. A QoS-based selection approach of autonomic grid services. *In Proceedings of the 2007 workshop on service-oriented computing performance. Aspects, issues, and approaches*.pp. 1–8 . Monterey, California, USA: ACM, 2007

[7] Yu, T., & Lin, K.-J. (2005). Service selection algorithms for composing complex services with multiple qos constraints. *In Proceedings of the Third International Conference on Service-Oriented Computing*, pp. 130–143. Amsterdam, The Netherlands: Springer-Verlag, 2005

[8] Kofler, K., ul Haq, I., & Schikuta, E, A parallel branch and bound algorithm for workflow QoS optimization. *International Conference on Parallel Processing*, 2009. pp. 478–485.

[9] Preve, N. P, *Grid Computing: Towards a Global Interconnected Infrastructure.* Springer, 2011

[10] Korte, B., & Vygen, J, *Linear Programming* . Berlin Heidelberg: Combinatorial Optimization Springer, pp. 51-71, 2012

[11] Vanderbei, R. J,*). Linear Programming: Foundations and Extensions*. Springer London Limited, 2008.

[12] Liu, M., Wang, M. R., Shen, W. M., Luo, N., & Yan, J. W, A quality of service (QoS)-aware execution plan selection approach for a service composition process. *Future Generation Computer Systems-the International Journal of Grid Computing and science*, 28, pp.1080–1089, 2013.

[13] Qi, Y., & Bouguettaya, A, Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25, pp.776–789, 2013.

[14] Knuth, D. E, *The Art of Computer Programming*. Volume. 4 A: Combinatorial Algorithms, Part 1, Pearson Education India, 2011.

[15] Shrme, A. E, Hybrid intelligent technique for automatic communication signals recognition using bees algorithm and MLP neural networks based on the efficient features. *Expert Systems with Applications*, 38, pp.6000–6006, 2011.

[16] Wang, J.-Z., Wang, J.-J., Zhang, Z.-G., & Guo, S.-P, Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, 38, pp.14346–14355, 2011.

[17] Jungmann, A., & Kleinjohann, B, Towards the Application of Reinforcement Learning Techniques for Quality-Based Service Selection in Automated Service Composition. *International Conference on Services Computing (SCC)*, 2012 IEEE, pp. 701–702, 2012

[18] Ludwig, S. A, Clonal selection based genetic algorithm for workflow service selection. *IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1–7, 2012.

[19] Yang, Y., Mi, Z., & Sun, J, Game theory based IaaS services composition in cloud computing environment. *Advances in Information Sciences and Service Sciences*, 4, pp.238–246, 2012.

[20] Wang, S. G., Sun, Q. B., Zou, H., & Yang, F. C, Particle swarm optimization with skyline operator for fast cloud-based web service composition. *Mobile Networks & Applications*, 18, pp.116–121, 2013.

[21] Jula, A., Othman, Z., & Sundararajan, E, A hybrid imperialist competitivegravitational attraction search algorithm to optimize cloud service composition. *IEEE Workshop on Memetic Computing (MC)*, 2013, pp. 37–43, 2013

[22] Bao, H. H., & Dou, W. C, A QoS-aware service selection method for cloud service composition. In *2012 IEEE 26th international parallel and distributed processing symposium workshops & Phd Forum* pp. 2254–2261. New York: IEEE, 2012.

[23] Wang, J.-Z., Wang, J.-J., Zhang, Z.-G., & Guo, S.-P, Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, 38, pp.14346–14355, 2011.

[24] Ye, Z., Zhou, X., & Bouguettaya, A, Genetic algorithm based QoS-aware service compositions in cloud computing. In J. Yu, M. Kim, & R. Unland (Eds.). *Database Systems for Advanced Applications* ,6588, pp. 321–334. Berlin Heidelberg: Springer, 2011.

[25] Zhu, Y., Li, W., Luo, J., & Zheng, X, A novel two-phase approach for QoSaware service composition based on history records. *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2012, pp. 1–8.

[26] Worm, D., Zivkovic, M., van den Berg, H., & van der Mei, R, Revenue maximization with quality assurance for composite web services. *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2012, pp. 1–9.

[27] Sundareswaran, S., Squicciarini, A., & Lin, D, A Brokerage-Based Approach for Cloud Service Selection. *IEEE 5th International Conference on Cloud Computing (CLOUD*), 2012, pp. 558–565.

[28] Chunqing, C., Shixing, Y., Guopeng, Z., Bu Sung, L., & Singhal, S, A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises. *IEEE 5th International Conference on Cloud Computing (CLOUD)*, 2012, pp. 883–890.

[29] Xiaona, W., Bixin, L., Rui, S., Cuicui, L., & Shanshan, Q, Trust-Based Service Composition and Optimization. *In Asia-Pacific Software Engineering Conference (APSEC*), 2012,Vol. 1, pp. 67–72.

[30] Zhang, X. Y., & Dou, W. C, Preference-aware QoS evaluation for cloud web service composition based on artificial neural networks. In F. L. Wang, Z. G. Gong, X. F. Luo, & J. S. Lei (Eds.). *Web Information Systems and Mining*, 6318, pp. 410–417. Berlin: Springer-Verlag Berlin, 2010.

[31] Karim, R., Chen, D., & Miri, A, An end-to-end QoS mapping approach for cloud service selection. *IEEE Ninth World Congress on Services (SERVICES)*, 2013, pp. 341–348.

[32] Zibin, Z., Yilei, Z., & Lyu, M. R, Distributed QoS Evaluation for Real-World Web Services. *IEEE International Conference on Web Services (ICWS)*, 2010, pp. 83–90.

[33] Xin Zhou, Liwei shen, Xin Peng, Wenyun Zhao, “Towards SLA constrained service composition: An Approach based on a fuzzy linguistic preference model and an evolutionary algorithm”, *Information Science, Elsevier*, 316, 2015, pp.370-396

[34] Tao Yu, Yue Zhang, Kwei Jay Lin, “Efficient algorithms for web services selection with end-to-end QoS constraints”, *ACM transactions on the web*, vol. 1 issue 1 2007

[35] Tan, Khor and Lee, *Multi objective Evolutionary Algorithms and Applications* , Springer-Verlag London Limited 2005

[36] Sivanandam · S.N.Deepa,*Introduction to Genetic Algorithms*, Springer-Verlag Berlin Heidelberg 2008

[37] Wang, Qibo Sun, Hua Zou, Fangchun Yang, Particle Swarm Optimization with Skyline Operator for Fast Cloud-based Web Service Composition, *Mobile Network Applications* 18, pp.116–121, 2013.