Genetic Approach for Service Selection Problem in Composite Web Service

N.Sasikaladevi
Research Scholar,
Dept. of Computer science
St.Joseph's College,
Trichy, TN, India

L.Arockiam
Associate Professor,
Dept. of Computer science
St.Joseph's College,
Trichy, TN, India

ABSTRACT
Services are the basic amass that aims to support the building of business application in a more flexible and interoperable manner for enterprise collaboration. Satisfying the needs of service consumer and to become accustomed to changing needs, service composition is performed to compose the various capabilities of available services. With the proliferation of services presenting similar functionalities around the web, the task of service selection for service composition is intricate. It is vital to provide systematic methodology for selecting required web services according to their non-functional characteristics or quality of service (QoS). Various heuristic and meta-heuristic algorithms are evolving to solve the QoS based service selection problem. One of the meta-heuristic algorithms is genetic algorithm. In this paper, the genetic algorithm is developed to maximize the non-functional Characteristic called the reliability of the composite web service and the performance of the developed algorithm is calculated.

General Terms
Service Oriented Architecture, Service Selection Algorithms

Keywords
Genetic Algorithm, MMKP, Fitness

1. INTRODUCTION
The Service Oriented Architecture (SOA) is a "software architecture that represents software functionality as services over the network". Web Services are the predominant implementation platform for SOA and it uses a set of standards, SOAP, UDDI, WSDL, which enable a lithe way for applications to intermingle with each other over networks. Simple Object Access Protocol (SOAP) is a standard protocol that allows network communication between services. The easiest way to publish a web service is to use a SOAP container. When a software component is published as a web service, any SOAP-enabled client that knows the network address of the web service can send a SOAP request and get a SOAP response. To get the message information, SOAP-enabled clients read a WSDL file that describes the web service. Once the Web Services Description Languages (WSDL) file is read, the client can start sending SOAP messages to the web service. WSDL describes what a web service can essentially do, where it resides, and how to invoke it. Universal Description Discovery and Integration (UDDI) is a standard that allows information about businesses and services to be electronically published, queried and stored. Published information is stored into one or more UDDI registries, which can be accessed through SOAP.

All these standards are XML-based, which allows applications to intermingle with each other over networks, regardless of what languages and platforms they are using. The two features, self-description and language-platform-independence, differentiate web services from other distributed computing technologies, like Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM). Research in web services includes many demanding areas starting from service publication to service mining. The most imperative among them is web service composition. Web service composition is needed when a client's complex request cannot be answered by single service, but by combining or composing various functionalities of available services or more than one services. Service composition involves three different issues. The first, called selection of service is fretful with selecting suitable services to composite that satisfy the user requirement. The second, called composition synthesis is concerned with synthesizing a specification of how to coordinate the component services to fulfill the client request. The third issue, called as service orchestration is concerned with achieving the synchronization among services by executing the specification produced by the composition synthesis.

This paper presents various service selection algorithm and techniques available for composite web services. The selection of web service is based on the non-functional characteristics called reliability. Service reliability estimation method is proposed. The genetic based service selection algorithm is developed. The developed algorithm is compared with the heuristic algorithm based on time complexity.

2. RELATED WORK
Yi Xia et.all [1] derived a QoS-Aware Web Service Selection Algorithm Based on Clustering. This algorithm is based on the service clustering which can cluster a lot of atomic services of each task into a few classes according to their QoS properties. With the help of service clustering, this algorithm is capable to reduce the execution time and promise the near-optimal result as well. Finally, three strategies are provided for re-selecting atomic services in dynamic environment. In experiment, they studied the performance of QSSAC

algorithm, and its feasibility had been demonstrated by simulation.

Maolin Tang et.all [2] proposed a hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. They propose a new hybrid genetic algorithm for the optimal web service selection problem. The hybrid genetic algorithm had been implemented and evaluated. The evaluation results have shown that the hybrid genetic algorithm outperforms other two existing genetic algorithms when the number of web services and the number of constraints are large.

Ping Wang et. All [3] proposed an evidence-based scheme for web service selection. Their model effectively enables trickery detection by means of existing bodies of verification, and therefore excludes the fraudulent evidence of malevolent evaluators from the selection process. In addition, a quality index is proposed to help third party examine the body of evidence and make the outranking result more reliable. Importantly, the quality index is based not only on the confidence degree of the evidence, but also on the support degree, and therefore discovers the effects of intentional negative assessments. The validity of the approach is demonstrated numerically by means of two service selection.

Qibo Sun et.all [4] derived a QoS-aware Service Selection Approach. This work proposes a QoS-aware Service Selection Approach (QSSA) with particle throng optimization and fuzzy logic control to support fast and dynamic service selection and assist users in selecting the most suitable services. The core of QSSA is decomposing global QoS constrains to local constraints and then selecting a local optimization with local selection. Experimental results demonstrate that QSSA can obtain the most suitable composite service with low cost.

Shangguang Wang et.all [5] proposed a Web Service selection based on QoS estimation. In this paper, they propose a WS Selection Approach based on QoS Estimation (WSSAQE). The aim of WSSAQE is to perform accurate QoS estimation, and then assuage the deviations between requiring and receiving QoS in WS selection. Experimental results show that their proposed WSSAQE is effective and efficient. Moreover, it significantly improves the QoS-based WS selection process.

3. SERVICE SELECTION AS AN OPTIMZATION PROBLEM

The optimization algorithms can be estranged into two categories: deterministic algorithms and stochastic algorithms. Deterministic algorithms go after a meticulous procedure and its path and values of both design variables and the functions are repeatable. For example, hill-climbing is a deterministic algorithm, and for the same initial point, they will follow the same path whether you execute the program when ever. On the other hand, the stochastic algorithms forever have some randomness. Genetic algorithms are a fine example, the strings or solutions in the population will be dissimilar each time you run a program since the algorithms use some pseudorandom numbers, though the final results may be no huge difference, but the paths of each individual are not precisely repeatable. Furthermore, there is a third type of algorithm which is a combination or hybrid of deterministic and stochastic algorithms. For example, hill-climbing with a random restart is a fine example. The basic initiative is to use the deterministic algorithm, but start with different initial points.

This has persuaded advantages over a simple hill-climbing technique which may be trapped in a local peak. Nevertheless, since Heuristics is a solution approach by trial-and-error to produce acceptable solutions to a complex problem in a reasonably practical time. The difficulty of the problem of interest builds it unfeasible to search every possible solution or combination, the aim is to find good, feasible solutions in an acceptable timescale. There is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why if it does work. The idea is that an resourceful but realistic algorithm that will work most of the time and be able to produce fine quality solutions. Among the found quality solutions, it is expected that some of them are almost optimal, though there is no guarantee for such optimality [6].

There is a random component in this hybrid algorithm; we often categorize it as a sort of stochastic algorithm in the optimization literature. The majority conventional algorithms are deterministic. For example, the Simplex method in linear programming is deterministic. Some deterministic optimization algorithms used the incline information; they are called gradient-based algorithms. For example, the well-known Newton-Raphson algorithm is gradient-based as it uses the function values and their derivatives, and it works extremely well for smooth uni modal problems. Nevertheless, if there is some discontinuity in the objective function, it does not work well. In this case, a non-gradient algorithm is preferred. Non-gradient-based or gradient-free algorithms do not use any derivative, but only the function values. Hooke-Jeeves pattern search and Nelder-Mead downhill simplex are examples of gradient-free algorithms.

For stochastic algorithms, we have in universal two types: heuristic and meta heuristic, though their difference is small. Loosely speaking, heuristic means 'to find' or 'to discover by trial and error'. Quality solutions to a hard optimization problem can be found in a sensible amount of time, but there is no assurance that optimal solutions are reached. It is anticipated that these algorithms work most of the time, but not all the time. This is frequently fine enough when we do not inescapably desire the finest solutions but rather good solutions which are simply reachable. Auxiliary development over the heuristic algorithms is the professed meta heuristic algorithms. Here meta- means 'beyond' or 'higher level', and they generally perform better than simple heuristics. In addition, all meta heuristic algorithms use definite tradeoff of randomization and local search. It is significance to pointing out that no agreed definitions of heuristics and meta heuristics exist in literature, some use 'heuristics' and 'meta heuristics'

interchangeably. However, recent trends tend to name all stochastic algorithms with randomization and local search as meta heuristic. Here we will also use this convention. Ran domination provides a good way to move away from local search to the search on the global scale. Therefore, almost all meta heuristic algorithms intend to be suitable for global optimization [6].

Most of the meta heuristic algorithms are nature-inspired as they have been developed based on some abstraction of nature. Nature has evolved over millions of years and has found perfect solutions to almost all the problems she met. We can thus learn the success of problem-solving from nature and develop nature-inspired heuristic and/or meta heuristic algorithms. More specifically, some nature-inspired algorithms are inspired by Darwin's evolutionary theory. Consequently, they are said to be biology-inspired or simply bio-inspired.

Two major components of any meta heuristic algorithms are: selection of the best solutions and randomization. The selection of the best ensures that the solutions will converge to the optimality, while the randomization avoids the solutions being trapped at local optima and, at the same, increase the diversity of the solutions. The good combination of these two components will usually ensure that the global optimality is achievable. Meta heuristic algorithms can be classified in many ways. One way is to classify them as: population-based and trajectory-based. For example, genetic algorithms are population-based as they use a set of strings, so is the particle swarm optimization (PSO) which uses multiple agents or particles. PSO is also referred to as agent-based algorithms[7]

3.1 Service Selection Problem in MMKP Form

The service selection problem is formulated as Multidimensional Multi choice Knapsack Problem (MMKP) form. Composite web service consists of number of atomic services. Numerous web services are evolving today. Web services with same functionality from different vendors are available now. Choosing the best service based on reliability rate is a simple task. But cost of services varies and it is depends on the service provider. Choose one service from each group based on the reliability rate and the total cost of these services should be less than or equal to the cost defined in Service level agreement (SLA). This is the optimization problem formulated in MMKP form. For a composite service that has N service classes ($S_1, S_2 \ldots S_n$) in a work flow plan and with m QoS constraints, we map the service selection problem to a 0-1 multidimensional multichoice knapsack problem (MMKP) [8,9]. MMKP is defined as follows.

Suppose there are N object groups, each has $l_i\ (1 \le i \le N)$ objects. Each object has a profit $p_{ij}$and required resources $r_{ij} = (r_{ij}^1, r_{ij}^2, \ldots, r_{ij}^m)$. The amount of resources available in the knapsack is $R = (R_1, R_2, \ldots, R_m)$. MMKP is to select exactly on object from each object group to be placed in the knapsack so that the total profit is maximized while the total resources used are less that the available resources.

The QoS service selection problem is to select one service candidate from each service class to construct a composite service that meets users' QoS constraints and maximizes the totol utility[10,11,12,13]. The QoS service selection problem is mapped to MMKP as follows.

1. Each service class is mapped to an object group in MMKP.
2. Each atomic service in a service class is mapped to an object in a group in MMKP.
3. The utility a candidate produces is mapped to the profit of the object.
4. The users' QoS constraints are considered as the resource available in the knapsack.

Mathematically, the service selection problem is formulated as follows:

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

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

$$\sum_{j \in S_i} x_{ij} = 1$$

where $x_{ij} \in \{0,1\} i = 1,2 \ldots . n, j \in S_i$

where x_{ij} is set to 1 if atomic service j is selected for class S_i and 0 otherwise. $q_{ij} = [q_{ij}^1, \ldots q_{ij}^n]$ is the QoS resource needs of each atomic service j for class S_i; the sum of all resources used by all service must be less that the overall constraints Q_c. The MMKP problem has been shown to be NP-complete [9]. We may solve MMKP by finding optimal results or use heuristic algorithms to reduce the time complexity.

4. A POPULATION BASED META-HEURISTIC ALGORITHM-GENETIC APPROACH

Genetic algorithm (GA) works on the Darwin's principle of natural selection. The theoretical foundations of GAs were originally developed by Holland. GAs is based on the evolutionary process of biological organisms in nature. During the course of evolution, natural populations progress according to the principle of natural selection and "survival of the fittest". Individuals which are more triumphant in adapting to their environment will have a better chance of surviving and reproducing, whilst individual which are less fit will be eliminated.

Genetic Algorithm hoists a couple of significant features. First it is a stochastic algorithm; randomness as an indispensable role in genetic algorithms. Both selection and reproduction needs random procedures. A second very significant point is

that genetic algorithms at all times reflect on a population of solutions. Keeping in memory more than a single solution at each iteration presents a lot of benefits. The algorithm can recombine diverse solutions to get better ones and so, it can use the benefits of hodgepodge. A population base algorithm is also very acquiescent for parallelization. The robustness of the algorithm should also be revealed as somewhat essential for the algorithm success. Robustness refers to the ability to perform consistently

4.1 A Genetic Algorithm

An algorithm is a sequence of steps for solving a problem. A genetic algorithm is a problem solving method that utilizes genetics as its model of problem solving. It's a search technique to discover approximate solutions to optimization and search problems. Fundamentally, an optimization problem seems really simple. One be acquainted with the form of all possible solutions corresponding to a precise question. The set of all the solutions that convene this form make up the search space. The problem consists in finding out the solution that fits the best, i.e. the one with the most payoffs, from all the possible solutions. If it's promising to quickly itemize all the solutions, the problem does not hoist much difficulty. But, when the search space becomes huge, enumeration is soon no longer feasible merely because it would acquire far surplus time. In this it's required to apply a specific technique to find the optimal solution. Genetic Algorithms offers one of these methods. Almost they all work in a similar way, become accustom with the simple genetics to algorithmic mechanisms.

GA touches a population of possible solutions. Each solution is symbolized through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a Genetic Algorithm. A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes, and are used to perform mutations and recombinations over solutions of the problem. Suitable depiction and reproduction operators are really something determinant, as the performance of the GA is extremely ward on it.

Commonly, it can be awfully hard to find a representation, which respects the structure of the search space and reproduction operators, which are consistent and appropriate according to the properties of the problems.

Selection is believed to be able to evaluate each individual in the population. Selection is finished by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it characterizes. The fitness should match up to a valuation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms pact with the problems that maximize the fitness function. But, if the problem consists in minimizing a cost function, the variation is reasonably easy. Moreover the cost function can be changed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they consider individuals with low evaluation functions as better.

Once the reproduction and the fitness function have been accurately defined, a Genetic Algorithm is evolved according to the same basic structure. It commences by generating an initial population of chromosomes. This first population must volunteer a wide diversity of genetic materials. The gene pool should be as huge as possible so that any solution of the search space can be provoked. Generally, the initial population is generated randomly. Then, the genetic algorithm loops over an iteration process to make the population evolve.

Each iteration consists of the following steps:

• SELECTION: The first step consists in selecting individuals for reproduction. This selection is done randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.
• REPRODUCTION: In the second step, children are breed by the selected individuals. For generating new chromosomes, the algorithm can utilize both recombination and mutation.
• EVALUATION: Then the fitness of the new chromosomes is weighed up.
• REPLACEMENT: During the last step, individuals from the old population are killed and replaced by the new ones.

The algorithms stopped when the population congregates toward the optimal solution.
The basic genetic algorithm is as follows:

- [start] Genetic random population of n chromosomes (suitable solutions for the problem)
- [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
- [New population] Create a new population by repeating following steps until the New population is complete
- [Selection] select two parent chromosomes from a population according to their fitness.
- [crossover] With a crossover probability, cross over the parents to form new offspring. If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome)
- [Accepting] Place new offspring in the new population
- [Replace] Use new generated population for a further sum of the algorithm.
- [Test] If the end condition is satisfied, stop, and return the best solution in current population.

- [Loop] Go to step2 for fitness evaluation.

4.2 Service Selection Algorithm Using Genetic approach

Here, we use Genetic Algorithms to solve the MMKP where one has to maximize the profit of group of objects in a knapsack without exceeding its capacity[14,15,16].

4.2.1 Encoding of the chromosomes

The chromosomes in GAs symbolize the space of candidate solutions. Promising chromosomes encodings are binary, permutation, value, and tree encodings. For the Knapsack problem, we use binary encoding, where every chromosome is a string of bits, 0 or 1. A chromosome can be represented in an array having size equal to the number of the groups (in our example of size 4). Each element from this array indicates whether a group is included in the knapsack ('1') or not ('0'). For example, the following chromosome: 0 1 2 3 indicates that the 1st and the 4th groups are included in the knapsack.. To represent the whole population of chromosomes we apply a three dimensional array (chromosomes [*Size*][*number of groups*][number of items]). *Size* stands for the number of chromosomes in a population. The second dimension represents the groups that may potentially be included in the knapsack. The third dimension represents the items that may potentially be included in the knapsack.

4.2.2 Fitness function

GAs necessitates a fitness function which allocates a score to each chromosome in the current population. Thus, it can calculate how well the solutions are coded and how well they solve the problem. We compute the fitness of each chromosome by summing up the profits of the items that are included in the knapsack, while making sure that the capacity of the knapsack is not exceeded. If the volume of the chromosome is greater than the capacity of the knapsack then one of the bits in the chromosome whose value is '1' is inverted and the chromosome is checked again.

4.2.3 Group Selection

The selection process is based on fitness. Chromosomes that are evaluated with higher values (fitter) will most likely be selected to reproduce, whereas, those with low values will be discarded. The fittest chromosomes may be selected several times, however, the number of chromosomes selected to reproduce is equal to the population size, therefore, keeping the size constant for every generation. This phase has an element of randomness just like the survival of organisms in nature.

The selection method was implemented in order to increase the probability of selecting fitter chromosomes to reproduce more often than chromosomes with lower fitness values. Array of chromosomes was sorted based on their fitness values in ascending order. Thus, the indices of the chromosomes with higher fitness values were at the end of the array *indexes*, and the ones with lower fitness will be towards the beginning of the array. Then, we randomly choose a group from the first group with 5% probability, from the second group with 10% probability, from the third group with 15% probability, from the fourth group with 25% probability and from the last group with 45% probability. Thus, the fitter a chromosome is the more chance it has to be chosen for a parent in the next generation.

4.2.4 Crossover

Crossover is the procedure of combining the bits of one chromosome with those of another. This is used to create an offspring for the next generation that accedes to the traits of both parents. Single point crossover randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring.

4.2.5 Mutation

Mutation is complete to prevent GAs from droping into a local extreme. Mutation changes the new offspring by flipping bits from 1 to 0 or from 0 to 1. Mutation can occur at each bit position in the string with some probability, usually very small (e.g. 0.001).

4.2.6 Elitism

Elitism was used where two of the fittest chromosomes are copied without changes to the new population, so the best solutions found will not be lost.

4.3 PBSSA

The new population based service selection algorithm (PBSSA) is developed to optimize the service selection problem in composite web service environment. PBSSA is presented here.

Step 1:
Initialize the no improvement counter to 0 and fitness loop counter to 0
Selection of Initial generations ώ=(x1,x2…xn) and sort the chromosomes based on the fitness value.
Step 2:
Calculate the maximum fitness of the current generations as δp
Construct the new generation β with two chromosomes with largest fitness
Step 3:
Select p1, a single chromosome from the categories of current generation using random value
Select p2, a single chromosome from the categories of current generation using random value
Do crossover on p1 and p2
Place the new offspring into the next generation γ
Repeat step 3 for population/2 number of times
Step 4:
Replace the current generation β with the new generation γ
Merge sort the current generation β based on the fitness value of chromosome
Do the mutation on the current generation β
Set δ as the maximum fitness of current generation.
If δ <= δp then Increment the no improvement count
Else set no improvement count as 0
If improvement count > ρ then exit

Repeat step 2-4 for the maximum number of generations.

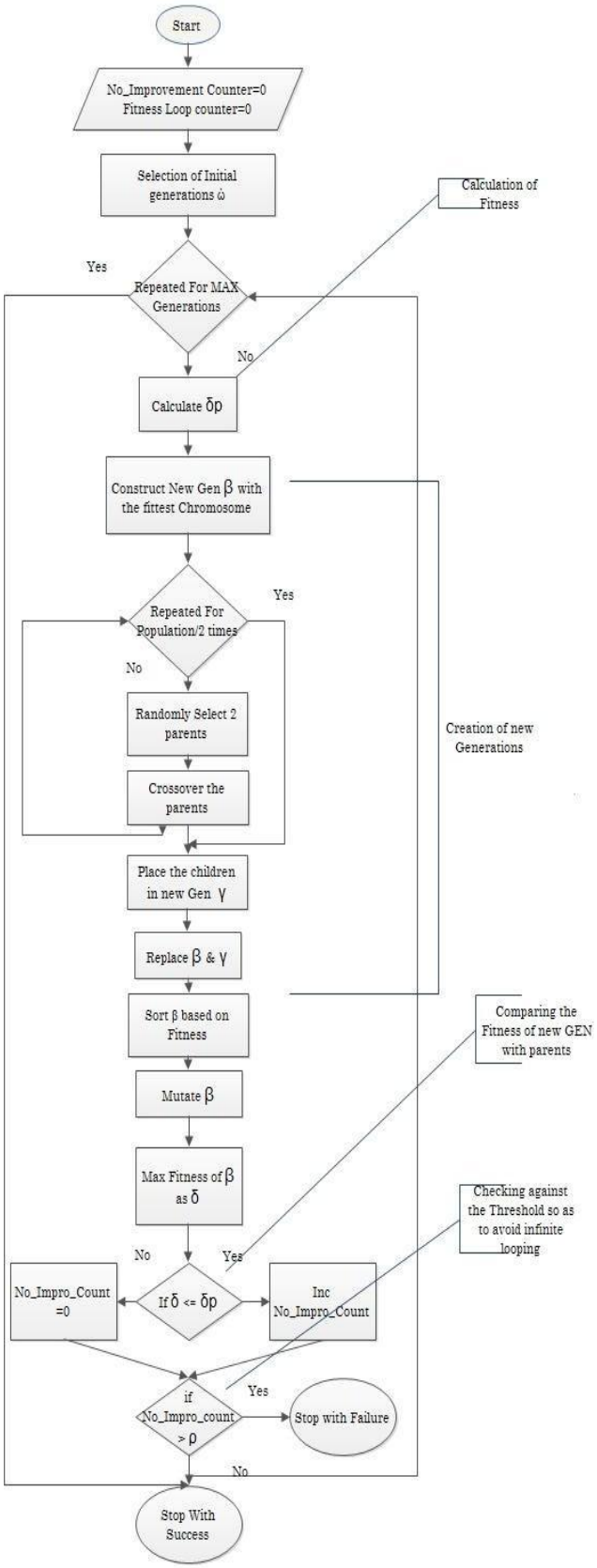

Fig. 1 PBSSA Flow chart

4.4 Computational Complexity

Average complexity of the above algorithm is O(*number of generations * number of chromosomes*).
Cost is defined as a constraint here. The value of the nodes is the reliability rate of services. The above algorithm maximizes the reliability rate of composite web service by choosing the best service from each group.

5. EXPERIMENTAL RESULT

Web services which are relevant to students information processing are collected from web. Among these web services, the fifty most relevant web services for students are identified. Reliability value is calculated for each of these web services. Invocation history for these web services is collected and invocation records are constructed. Totally 10000 invocation records are created for each web service and it is divided into 100 fragments of size 1000. Reliability status is calculated on each fragment and time percentage is calculated using the equation (6). Part of the service invocation registry is shown below (status value in first 10 time period is included).

Each of the 10 web services falls in any of the 3 status from time t1 to t10 where "S" denotes the continuous success, "F" denotes the continuous failure and "T" denotes the transitory failure. By using the status information, we calculate how lone the service is in each of the status and reliability rate. Time percentage in each status for every web services is calculated based on the equation (6). Then we calculate the reliability rate of every web services in each state is estimated. The following graphs show the results.

Finally, Reliability value is estimated for each service. The value ranges from 0 to 1. "0" indicates that the service is not available for long period of time. "1" indicates that the service is always available. Reliability value for most of the web services falls between 0 and 1.

Table 1. Test cases results of Service Selection Algorithm

Test Case	Service Groups	Service Candidates	Execution Time(Sec.)	Optimality of Results
1	5	2	0.031	97
2	5	4	0.078	97
3	5	6	0.031	98
4	5	8	0.062	97
5	5	10	0.031	98
6	5	12	0.031	97
7	5	14	0.031	98
8	5	16	0.031	98
9	5	18	0.047	98
10	5	20	0.047	98
11	10	2	0.031	89
12	10	4	0.047	84
13	10	6	0.031	84
14	10	8	0.047	84

15	10	10	0.047	90
16	10	12	0.062	93
17	10	14	0.062	95
18	10	16	0.062	96
19	10	18	0.062	96
20	10	20	0.125	97
21	15	2	0.031	71
22	15	4	0.047	93
23	15	6	0.047	90
24	15	8	0.062	89
25	15	10	0.062	85
26	15	12	0.062	96
27	15	14	0.078	91
28	15	16	0.078	95
29	15	18	0.094	97
30	15	20	0.109	97
31	20	2	0.047	73
32	20	4	0.031	96
33	20	6	0.062	91
34	20	8	0.062	88
35	20	10	0.078	85
36	20	12	0.125	96
37	20	14	0.125	93
38	20	16	0.109	93
39	20	18	0.125	91
40	20	20	0.125	94
41	25	2	0.047	74
42	25	4	0.047	90
43	25	6	0.062	82
44	25	8	0.078	81
45	25	10	0.094	86
46	25	12	0.109	84
47	25	14	0.125	86
48	25	16	0.14	88
49	25	18	0.125	74
50	25	20	0.156	93

The genetic algorithm for MMKP as shown in figure is implemented as win 32 console application in C++. It is developed in Micosoft Visual C++ express edition and debugged. The sample output is shown in Fig.2. Total number of test cases created is 100. The table 4 shows the first 50 test cases. Fifty set of composite web services are developed. Candidate services ranges from 10 to 50 in each composite web services. The execution time is calculated The execution time of genetic algorithm is calculated. Theses algorithm are tested with Intel Core Duo CPU.

Time complexity of t this algorithm is analyzed. The execution time of Genetic algorithm is calculated and shown.

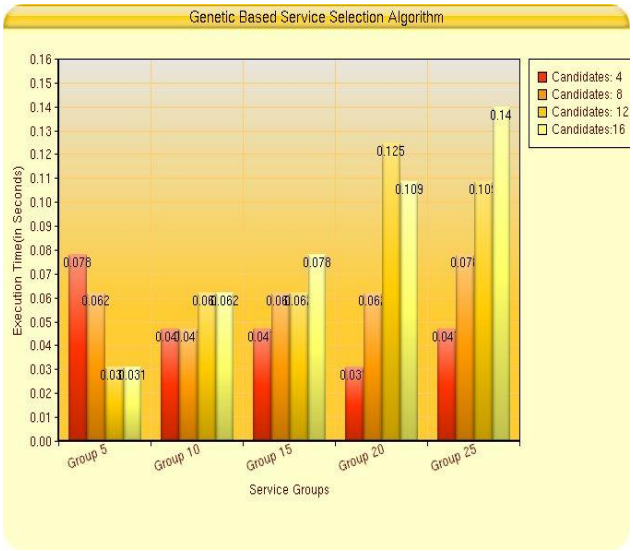

Fig. 2 Service Group ranges from 5 to 25

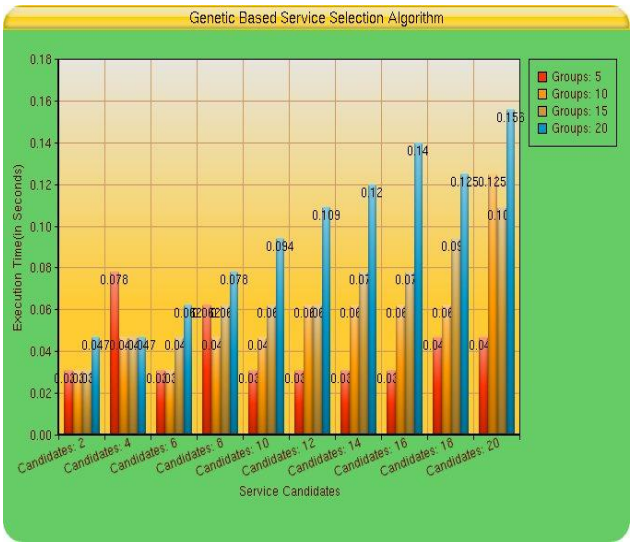

Fig. 3 Execution time chart of genetic (Candidates ranges from 2 to 10)

The Fig.3 shows the execution time for genetic algorithm for service selection problem for varying number of service candidates ranges from 2 to 20 for service groups ranges from 5 to 25. Time complexity gradually increases for large number of service candidate.

6. CONCLUSION

With the increasing reliability of Web services as a solution to enterprise application integration, the QoS parameters offered by Web services are becoming the chief priority for service providers and their service consumers. This paper presented a novel algorithm for web service candidate selection based on genetic model. The developed algorithm is tested and validated. In this paper services are selected based on the non-functional characteristics called reliability rate. In future, the other QoS metrics can also be considered to select the best candidate service for web service composition.

7. ACKNOWLEDGMENTS

This research work is being funded by the Department of Science and Technology (DST), Government of India.

8. REFERENCES

[1] M. Sathya, M. Swarnamugi, P. Dhavachelvan, G. Sureshkumar, "Evaluation of QoS based Web- Service Selection Techniques for Service Composition", *International Journal of Software Engineering (IJSE)* , 2011

[2] Huiyuan Zheng; Jian Yang; Weiliang Zhao; "QoS Analysis and Service Selection for Composite Services" , *2010 IEEE International Conference on Services Computing*, pp.122 – 129, 2010

[3] Ping Wang, Kuo-Ming Chao, Chi-Chun Lo and Ray Farmer, "An evidence-based scheme for web service selection ", *Special Issue: Advances in E-Business Engineering*, 2010

[4] Matteo Baldon, Cristina Baroglio, Alberto Martelli, Viviana Patti. "Reasoning about interaction protocols for customizing web service selection and composition". *The Journal of Logic and Algebraic Programming, Elsevier*, No. 70, pp. 53 – 73, 2007.

[5] Shangguang Wang; Zibin Zheng; Qibo Sun; Hua Zou; Fangchun Yang; "Cloud model for service selection", *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* pp.666 – 671, 2011

[6] Xin-She Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, Wiley Publications, 2010.

[7] E.-G. Talbi, *Metaheuristics: From Design to Implementation,* John Wiley Publications,2009.

[8] Kellerer H, Pferschy and Pisinger, *Knapsack Problems*, springer-Verlag, 2004

[9] 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, No. 1, Article 6, May 2007.

[10] Yi Xia; Ping Chen; Liang Bao; Meng Wang; Jing Yang; "A QoS-Aware Web Service Selection Algorithm Based on Clustering", *2011 IEEE International Conference on Web Services(ICWS),*pp. 428 – 435,2011.

[11] Chao Lv, Wanchun Dou, Jinjun Chen. "QoS-Aware Service Selection Using QDG for B2B Collaboration". *In Proceedings of the fourteenth IEEE International Conference on Parallel and Distributed System*s, pp. 336 – 343, 2008.

[12] Mobedpour, D.; Chen Ding; Chi-Hung Chi; "A QoS Query Language for User-Centric Web Service Selection" , *2010 IEEE International Conference on Services Computing (SCC),* pp.273 – 280, 2010.

[13] Qibo Sun , Shangguang Wang, Hua Zou, Fangchun Yang, "QSSA: A QoS-aware Service Selection Approach", *International Journal of Web and Grid Services*, Volume 7, Number 2 , pp.147 - 169 , 2011

[14] Liu Zhi-Zhong; Wang Zhi-Jian; Zhou Xiao-Feng; Lou Yuan-Sheng; Shang Ling; "A New Algorithm for QoS-Aware Composite Web Services Selection", 2nd *International Workshop on Intelligent Systems and Applications (ISA),* pp.1 - 4 ,2010

[15] Chengwen Zhang; Beijing, "Adaptive Genetic Algorithm for QoS-aware Service Selection", *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA),* 273 – 278,2011.

[16] Swarnamugi .M, Sathya .M. "Specification Criteria for Web Service Selection Approaches". *International Journal on Computer Engineering and Information Technology*, vol(23), Issue No: 01, pp. 29 – 382010.