Simulated Annealing Versus Genetic Based Service Selection Algorithms

L. Arockiam and N. Sasikaladevi

Department of Computer Science, St.Joseph's College, TN, India
larockiam@yahoo.com, sasikalade@yahoo.com

Abstract

In service oriented architecture, services are the basic assemble that aims to assist the building of business application in a more flexible and interoperable manner for enterprise collaboration. To satisfy the needs of service clients and to adapt 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 reliability. In this paper, the simulated annealing algorithm is developed to maximize the non-functional Characteristic reliability of the composite web service and the genetic algorithm is developed and theses two algorithms are compared based on the time complexity and optimality of results.

Keywords: *Simulated Annealing, Genetic algorithm, Service Reliability, Service Selection, MMKP*

1. Introduction

Reliability is the quality feature of a Web service that represents the scale of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a Web service. In another way, reliability refers to the assured and ordered delivery for messages being sent and received by service clients and service providers. Service reliability can also be defined as the continuity of correct service delivery. Reliability is a key requirement for building dependable systems. Reliability is the probability that the software will be functioning without failure under a given environmental condition during a specified period of time. Menascé [1] characterizes QoS as a combination of several properties, including availability, security, response time, and throughput. Cardoso et al. [2] develop ontology for the specification of QoS metrics, including task response time, task cost, and reliability as QoS dimensions of interest. Sumra and Arulazi [3] Suggest that web services QoS requirement include performance, reliability, integrity, accessibility, availability, interoperability, and security. Ran [4] proposes a new web services discovery model using QoS metrics, classifying these metrics into four distinct categories, related to execution, transaction, configuration management, and cost.

Reliability, security, cost, and performance are criteria that are often identified as being relevant non-functional requirements when selecting web services. To determine the relative importance of these criteria a survey was conducted among several information technology architects in the Midwest [5]. The relative weights of these criteria were incidental through the analytic hierarchy process [6]. The results were astonishingly consistent, and indicated

that security and reliability were among the most important criteria that architects consider when evaluating web services. These results are depicted in Figure 1.

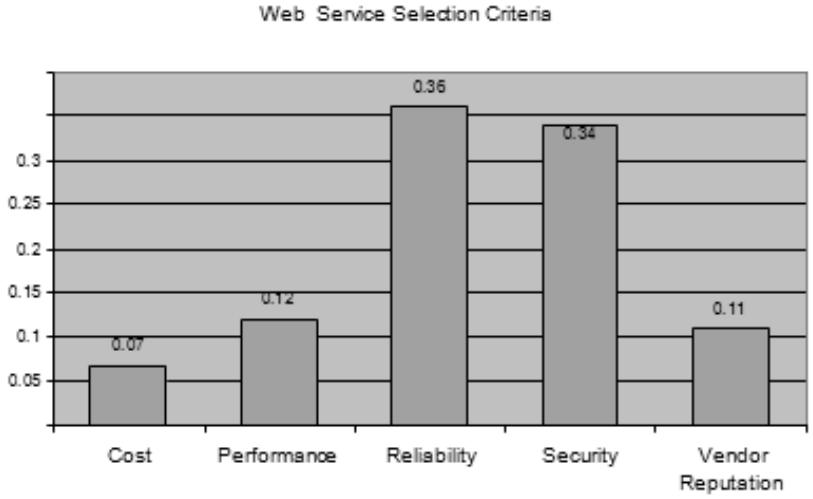

Figure 1. Web Service Selection Criteria

Software reliability is normally characterized as the probability that a piece of software will exhibit failure- free operation during a specified period. Xie [10] defines software reliability as "the probability that the software will be functioning without failures under a given environmental condition during a specified period of time". Other characterizations tend to measure reliability in terms of a percentage of failure cases in a given number of tries to compensate for variations in usage over time [7, 8]. The use of multiple concepts of failure and metrics for assessment of failure gives rise to a host of definitions. In the software context, failure is usually indicative of the exposure of underlying design faults by selection of specific inputs or operating conditions. Breaking down the source and nature of failures permits more precise modeling of software artifact reliability. However, most characterizations of reliability of web services favor to leave the sources of failure unstated [10]. Some researchers view reliability as a non-functional characteristic of web services [11]. Still others characterize it as a component in setting service level agreements with web services vendors. One of the more comprehensive characterizations of web services reliability is provided by Zhang and Zhang [11] where both functional and non-functional components are considered, and reliability is defined as a composite of correctness, fault tolerance, testability, availability, performance, and interoperability. This approach is useful in modeling reliability of an individual web service, with a view towards predicting when it may fail.

Business practice witnesses a rising interest in the ad-hoc model for service composition in the areas of supply chain management, accounting, finances, eScience as well as in multimedia applications. With the mounting number of available services the composition problem becomes a decision problem on the selection of component services from a set of alternative services that provide the same functionality but differ in quality parameters. Given an abstract representation of a composition request and given a list of functionally-simillar web service candidates for each work in the composition request, the goal of service selection algorithms is to find one web service from each list such that the overall QoS is optimized and user's end-to-end QoS requirements are satisfied. This problem can be modeled as Multi-

Choice Multidimensional Knapsack problem (MMKP), which is known to be NP-hard in the strong sense [13]. Therefore it can be predictable that any exact solution to MMKP has an exponential cost. In the dynamic environment of web services, where deviations from the QoS estimates occur and decisions upon replacing some services has to be taken at run-time, the efficiency of the applied service selection algorithm becomes vital.

The rest of the paper is organized as follows. In the next section we included related solutions. Section 3 introduces the system model and gives a problem statement. Genetic based reliable web service selection is presented in section 4. Simulated Annealing based reliable web service selection is presented in section 5. In section 6 and 7 we discuss the results from our experimental evaluation. Finally, section 8 gives conclusions and a viewpoint on possible continuations of our work.

2. Related Work

Sultana M et al [14] proposed a model for a distributed Web service system. Their model is composed of multiple Web service components each of which having multiple alternative versions distributed among multiple servers. For a specified set of service consumer requests an allocation is to be made that maximizes total service client satisfaction focus to the resource constraints of the servers. To solve this multidimensional multi knapsack problem, which is NP hard, they propose a heuristic using a variant of the network flow maximization algorithm. Not only the heuristic is polynomial but also it inherently rules out the number of requests from contributing in time complexity of the algorithm.

Maolin Tang et al [15] proposed hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. They tried to solve the constraints. There may be mutual constraints between some web service implementations. Their model tried to solve dependency constraint and conflict constraint between the functionally similar web service implementations. The optimal web service selection is an emblematic constrained combinatorial optimization problem from the computational point of view. Their hybrid genetic algorithm has been implemented and evaluated. The evaluation results show that the hybrid genetic algorithm outperforms other two existing genetic algorithms when the number of web services and the number of constraints are large.

Zhi-peng GAO et al [16] developed QoE/QoS driven simulated annealing-based genetic algorithm for Web services selection. To develop the efficiency and effect of Web service selection, an algorithm named quality of experience (QoE)/quality of service (QoS) driven simulated annealing-based genetic algorithm (QQDSGA) is given to achieve Web services selection efficiently with excellent QoE. QQDSGA includes two parts: QoE/QoS driven composite Web services evaluation model and simulated annealing-based (SA-based) genetic algorithm for Web services selection. Simulations show that when used in Web service selection, QQDSGA is better than genetic algorithm (GA) and SA in efficiency and is more satisfied in effect than no customer feedback.

Shang-Pin Ma et al [17] proposed a model for optimal service selection for composition based on weighted service flow and Genetic Algorithm. Their work proposes a new approach to solve this service selection optimization issue. In that approach, every abstract component service in the composition will be assigned a weight value to represent its importance through user assignment and execution path analysis, and then a weighted service flow will be generated. A service combination that includes selected concrete component services will be produced based on the weighted service flow and the Genetic Algorithm. The service combination will be of acceptable quality, and ensures the core component services that will most likely affect the whole service flow execution are more robust than other ones.

Dongmei Liu et al [18] developed A Heuristic QoS-Aware Service Selection Approach to Web Service Composition. Their work reflects a heuristic approach to optimizing service selection for composition so that the consumer specific QoS expectation is satisfied. Optimization is enforced for constraints in both local and global level. For each level, a mathematical model is established, and the corresponding heuristic algorithm to solve Web service selection problem by using convex hull frontiers method is proposed. Furthermore, multiple criteria decision making (MCDM) technique is applied to merge multiple dimensional resources in the heuristic algorithms.

These entire models use various algorithms to improve the QoS of Services. But in this paper, we focused on the reliability based service selection. We developed two algorithms to optimize the reliability with respect to cost by using genetic and simulated annealing algorithms. We implemented and compared the efficiency of these algorithms. The result shows that the simulated annealing algorithm out performs the genetic algorithm in reliability based service selection problem.

3. Service Selection Based on Service Reliability

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 constraints, we map the service selection problem to a 0-1 multidimensional multichoice knapsack problem (MMKP) [19]. MMKP is defined as follows.

Suppose there are N object groups, each has l_i $(1 \leq i \leq N)$ 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 reliable service selection problem is to select one service candidate from each service class to construct a composite service that meets users' constraints and maximizes the total utility. The reliable service selection problem is mapped to MMKP as follows.

- Each service class is mapped to an object group in MMKP.
- Each atomic service in a service class is mapped to an object in a group in MMKP.
- The utility a candidate produces is mapped to the profit of the object.
- The users' constraints are considered as the resource available in the knapsack.
- Mathematically, the service selection problem is formulated as follows:

$$Max \sum_{i=1}^{N} \sum_{j \varepsilon S_i} F_{ij} x_{ij}$$

$$Subject\ to \sum_{i=1}^{N} \sum_{j \varepsilon S_i} q_{ij} x_{ij} \leq Q_c \ and \sum_{j \varepsilon S_i} x_{ij} = 1$$

$$x_{ij}\, \varepsilon \{0,1\} \quad i=1,2,\ldots N, j \varepsilon S_i$$

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 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 [43]. We may solve MMKP by finding optimal results or use heuristic algorithms to reduce the time complexity

4. Service Selection Based on Genetic Algorithm

Genetic Algorithms (GAs) are adaptive heuristic search algorithm groundwork on the evolutionary ideas of natural selection and genetic. Genetic algorithm works on the Darwin's principle of natural selection. The theoretical basement of GAs were originally developed by Holland. GAs are 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.

A GA simulates these processes by taking an initial population of individuals and applying genetic algorithm in each reproduction. In optimization terms each individual in the population is encoded into a string or chromosome which represents a feasible solution to a given problem. The fitness of an individual is evaluated with respect to a given objective function. Extremely fit individuals or solutions have opportunities to replicate by exchanging pieces of their genetic information, in a crossover procedure, with other highly fit individuals. This produces new offspring solutions (i.e. children), which share some characteristics taken from both parents. 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.

4.1 Encoding of the Chromosomes

The chromosomes in GAs represent 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'). 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. N*umber of groups* represents the groups that may potentially be included in the knapsack. *Number of items* represents the items that may potentially be included in the knapsack.

4.2 Fitness Function

GAs demands a fitness function which allocates a score to each chromosome in the current population. Thus, it can compute how well the solutions are coded and how well they solve the problem. We calculate 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.

4.3 Group Selection

The selection process is based on fitness. Chromosomes that are estimated with higher values will most possibly 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.

The population was categorized into five groups:

(0*Size/5 … 1*Size/5)

(1*Size/5 … 2*Size/5)

(2*Size/5 … 3*Size/5)

(3*Size/5 … 4*Size/5)

(4*Size/5 … Size)

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.

4.4 Crossover

Crossover is the method 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 behavior 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.

4.5 Mutation

Mutation is absolute to prevent GAs from falling 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). For example, consider the following chromosome with mutation point at position 2:

Not mutated chromosome: 1 *0* 0 0 1 1 1

Mutated: 1 *1* 0 0 1 1 1

The 0 at position 2 flips to 1 after mutation. In our program, mutation was performed on random bit position of the chromosomes with probability ranging from 0.2% up to 1.50%.

4.6 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.

4.6 Pseudo Code

Select an initial generation, $\omega = (x1, x2, \ldots, xn)$ and sort its chromosomes according to fitness loop counter initially set to 0; no_improvement_count initially set to 0;
Repeat;
prev_max = maximum fitness of current generation;
copy two top scored chromosomes to the new generation without any change;
Repeat;
Set index = 0;
random_value = Random (0,1);
first_parent = Selected a single chromosome from categories of current generation using random_value;
random_value = Random (0,1);
second_parent = Selected a single chromosome from categories of current generation using random_value;
perform crossover on first_parent and second_parent;
index = index + 2;
copy two children generated to the next generation;
Until index = (population_size/2) – 2;
loop_counter = loop_counter + 1;
Replace the current generation with the new generation;
Sort current generation;
Perform mutation on the current generation;
cur_max = maximum fitness of current generation;
If cur_max ≤ prev_max, then
no_improvement_count = no_improvement_count + 1;
Else
no_improvement_count = 0;
End If
If no_improvement_count > improve_threshold, then
break;
Until loop_counter = maximum number of generations;

4.7 Computational Complexity

Average complexity of the above algorithm is *O(number of generations * number of chromosomes)*. Complexity of merge sort used is *O(number of chromosomes * lg(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. Service Selection Based on Simulated Annealing

Simulated Annealing is implemented to solve the MMKP with static cooling schedule. Here, a sensible strategy is used to generate random solution in the neighbourhood, the applicable parameters concerning cooling schedule are checked in order to obtain finite-time implementation of the algorithm. To formally describe simulated annealing algorithm for MMKP, some definitions are needed. Let X be the solution space; define $\eta(\omega)$ to be the neighbourhood function for $\omega \in X$.

Simulated annealing commences with an initial solution $\omega \in X$.. A neighbouring solution $\omega' \in \eta(\omega)$ is then generated randomly in most cases. Simulated annealing is based on the Metropolis acceptance criterion, which models how a thermodynamic system moves from its current solution $\omega \in X$ to a candidate solution $\omega' \in \eta(\omega)$, in which the energy content is being minimized. *tk* is defined as the temperature parameter at iteration k. Here, finite-time implementations of simulated annealing algorithm are considered, which can no longer guarantee to find an optimal solution, but may result in quicker executions without losing too much on the solution quality.

5.1 Initial Solution

Initial solution is generated by randomly selecting a set of groups and a single random item from each selected group. If half of the existing groups, each with a single item is chosen, without violating constraints, initial solution is generated.

5.2 Exploring Neighbor Solutions

In every iteration process, one group is randomly selected. If it is previously included in the solution set, then this group and its selected item is dropped from the solution set. A new group, with its new item, that is not included in solution set is now chosen by random selection and included in our fairly accurate solution. On the other hand, if the randomly selected group is not previously included in solution set, then a item in this group is randomly chosen and this new group with its item is included in our approximate solution set. In both of the above cases, practicability of the approximate solution is checked.

5.3 Acceptance Probability

The candidate solution, ω' , is accepted as the current solution on the acceptance probability,

$$P\{\text{Accept } \omega' \text{ as next solution}\} = \begin{cases} \exp[f(w') - f(w)/t_k] & \text{if } f(w') - f(w) < 0 \\ 1 & \text{if } f(w') - f(w) \geq 0 \end{cases}$$

based

At each temperature the length of inner loop is obtained by the Markov chain length, M. The program runs until a feasible and acceptable solution is found.

5.4 Pseudo Code

Select an initial solution,
ω= ($x1$, $x2$, . . . ,xn); an initial temperature $t = t0$; control parameter value α; final temperature e;a repetition schedule, M that defines the number of iterations executed at each temperature;
Approximate solution ← $f(\omega)$;
Repeat;
Set recurrence counter $m = 0$;
Repeat;
Select an integer i from the set of groups {1,2,3, . . . , n} randomly;
If $xi = 0$, pick up group i, i.e. set $xi = 1$, then
Select a random integer j from the set of items {1,2, 3, . . . , li} in this selected group;
Set $xij = 1$, obtain new solution ω',then
While solution ω' is not feasible, do
Drop another group from ω' randomly; denote the new solution as ω';
Let $\Delta = f(\omega') - f(\omega)$;
If $\Delta \geq 0$ or Random (0,1) < $\exp(\Delta/t)$, then $\omega \leftarrow \omega'$;
Else
Drop group i, and select another group, xk randomly from set{1,2, . . ., n}
Select a random integer j from the set of items {1,2, 3, . . . , li} in this selected group, xk ;
Set $xkj = 1$, attain new solution ω', then
While solution ω' is infeasible, do
Drop another group from ω' randomly; denote the new solution as ω';
Let $\Delta = f(\omega') - f(\omega)$;
If $\Delta \geq 0$ or Random (0,1) < $\exp(\Delta/t)$, then $\omega \leftarrow \omega'$;
End If
If approximate solution < $f(\omega)$, then approximate solution ← $f(\omega)$;
$m = m + 1$;
Until $m = M$;
Set $t = \alpha * t$;
Until $t < e$;

A set of parameters needs to be specified that direct the convergence of the algorithm, i.e. initial temperature $t0$, temperature control parameter α, final temperature e, and Markov chain length M, in order to revise the finite-time performance of simulated annealing algorithm. Here $t0$ should be the maximal difference in cost between any two neighbouring solutions, let $t0$ be 500 by jagged estimation of our test instances since exact calculation is quite time consuming. Final temperature e is chosen 1.0e-5. α is ranges from 0.85 to 0.95 Markov chain length M is determined as, M = n * Multiplication Factor where, n = number of groups and Multiplication Factor = a parameter governing speed at each temperature.

5.5 Computational Complexity

Average Time complexity of our implemented algorithm is *O(t*M)*. This complexity increases by a little amount whenever, approximate solution violates feasibility constraints and one or more groups are selected randomly to be dropped from solution set.Simulated Annealing algorithm is used to solve the service selection problem. The service selection problem is formulated as 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 and solved using simulated annealing algorithm depicted above.

Annealing iteration is a preset value. It is a meta- heuristic algorithm. 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.

6. Experimental Result

Functionally similar web services 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 in [20] (our pervious work). 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 shown in [20]. Then we calculate the reliability rate of every web services in each state is estimated.

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. Service Selection Problem-Test Cases

Test Cases	Service Groups	Service Candidates	Simulated Annealing Exe. Time	Result OP	Genetic Algorithm Exe. Time	Result OP
1	5	2	0.092	97	0.031	97
2	5	4	0.187	97	0.078	97
3	5	6	0.234	97	0.031	98
4	5	8	0.218	97	0.062	97
5	5	10	0.296	98	0.031	98
6	5	12	0.312	98	0.031	97
7	5	14	0.312	98	0.031	98

8	5	16	0.342	98	0.031	98
9	5	18	0.359	98	0.047	98
10	5	20	0.406	98	0.047	98
11	10	2	0.125	98	0.031	89
12	10	4	0.187	98	0.047	84
13	10	6	0.265	98	0.031	84
14	10	8	0.327	98	0.047	84
15	10	10	0.39	98	0.047	90
16	10	12	0.437	98	0.062	93
17	10	14	0.514	98	0.062	95
18	10	16	0.592	98	0.062	96
19	10	18	0.655	98	0.062	96
20	10	20	0.733	98	0.125	97
21	15	2	0.202	97	0.031	71
22	15	4	0.218	98	0.047	93
23	15	6	0.39	98	0.047	90
24	15	8	0.483	98	0.062	89
25	15	10	0.593	98	0.062	85
26	15	12	0.655	98	0.062	96
27	15	14	0.78	98	0.078	91
28	15	16	0.858	98	0.078	95
29	15	18	0.936	98	0.094	97
30	15	20	1.17	98	0.109	97
31	20	2	0.234	97	0.047	73
32	20	4	0.312	98	0.031	96
33	20	6	0.531	98	0.062	91
34	20	8	0.577	98	0.062	88
35	20	10	0.78	98	0.078	85
36	20	12	0.858	98	0.125	96
37	20	14	0.983	98	0.125	93
38	20	16	1.123	98	0.109	93
39	20	18	1.217	98	0.125	91
40	20	20	1.326	98	0.125	94
41	25	2	0.249	97	0.047	74
42	25	4	0.328	98	0.047	90
43	25	6	0.608	98	0.062	82
44	25	8	0.717	98	0.078	81
45	25	10	0.92	98	0.094	86
46	25	12	1.029	98	0.109	84
47	25	14	1.17	98	0.125	86
48	25	16	1.357	98	0.14	88
49	25	18	1.497	98	0.125	74
50	25	20	1.67	98	0.156	93

The genetic algorithm for MMKP as shown in section 4 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 same set of services is used in Annealing algorithm. 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. 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.

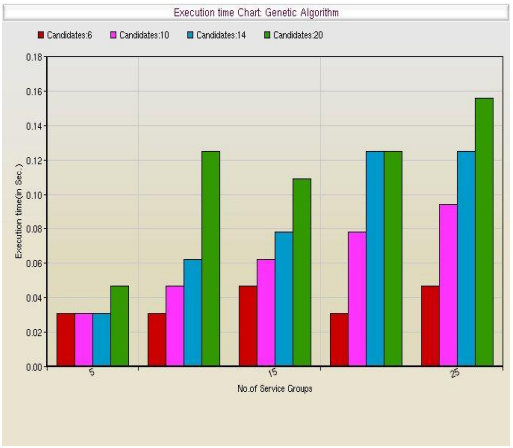

Figure. 2 Execution time chart of genetic (Group ranges from 5 to 25)

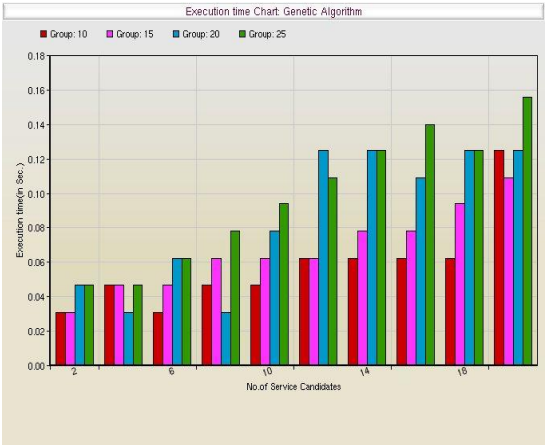

Figure. 3 Execution time chart of genetic (Candidates ranges from 2 to 10)

The simulated annealing algorithm for MMKP as shown in figure is implemented as win 32 console application in C++. It is developed in Microsoft Visual C++ express edition and debugged. The sample output is shown in Figure 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 same set of services is used in Annealing algorithm. The execution time of Annealing algorithm is calculated. Theses algorithm are tested with Intel Core Duo CPU.

Time complexity of this algorithm is analyzed. The execution time of Simulated Annealing algorithm is calculated and shown. The Figure 3 shows the execution time for simulated annealing 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.

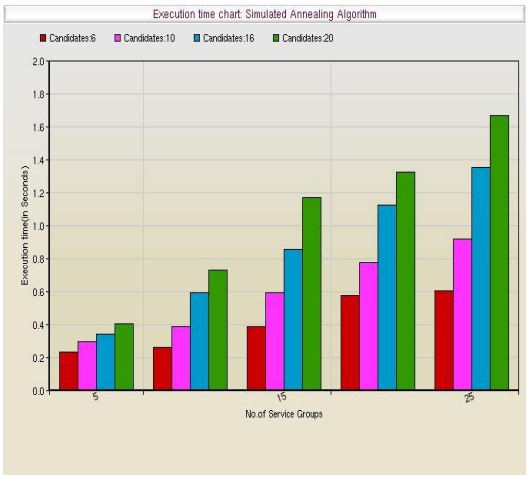

Figure. 5 Execution time chart of SA (Group ranges from 5 to 25)

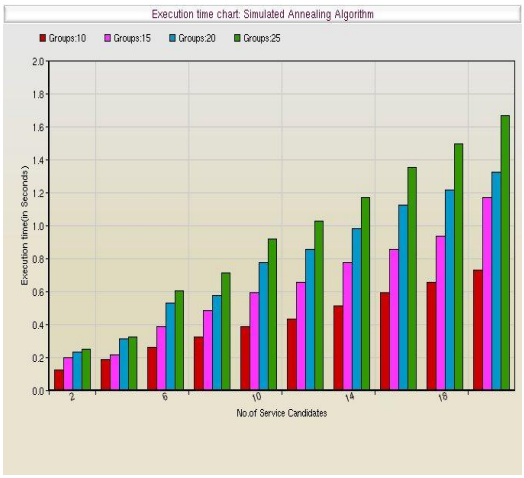

Figure. 6 Execution time chart of SA (Candidates ranges from 2 to 10)

7. Performance Comparison

Time complexity of simulated annealing algorithm for service selection problem is calculated for 100 test cases. Similarly time complexity of genetic based service selection algorithm is validated against the same set of test execution time of genetic algorithm is more than 2 times longer than the simulated annealing algorithm for service selection problem for composite web services

Figure 7. Execution Time of SA versus GA

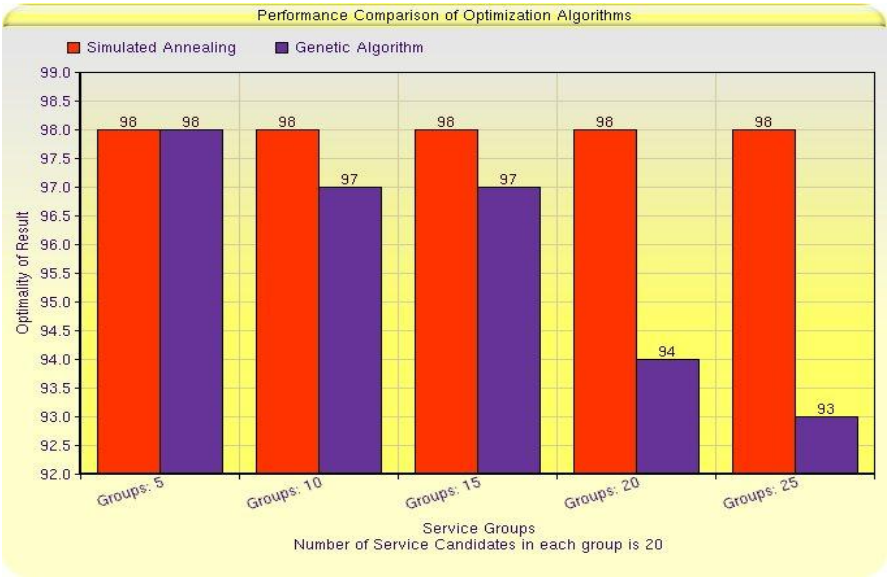

Figure 8. Optimality Value of SA versus GA

The developed algorithms are compared based on the time complexity and the optimality of the result. Figure 7 shows the time comparison of these two algorithms and figure 8 shows the optimality of results. The simulated annealing based reliable service selection algorithm out performs the genetic bases reliable service selection algorithm.

8. 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 novel algorithms 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.

Acknowledgement

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

References

[1] Menascé DA, "QoS Issues in Web Services," IEEE *Internet Computing*, **(2002)**, pp. 72-75.

[2] Cardoso J, Miller J, Sheth A, Arnold J, "Modeling Quality of Service for Workflows and Web Service Processes", Technical Report #02-002, LSDIS Lab, Computer Science, University of Georgia, **(2002)**.

[3] Sumra R, Arulazi D, "Quality of Service for Web Services – Demystification, Limitations, and Best Practices", Developer.com, **(2003)** March 4, http://www.developer.com/services/article.php/2027911.

[4] Ran S, "A Model for Web Services Discovery with QoS", *ACM S1Gecom Exchanges*, vol. 4 , **(2003)** Spring, pp. 1-10.

[5] Zo H, "*Supporting 1ntra- and 1nter-Organizational Business Processes with Web Services*", Ph.D. Thesis, The University of Wisconsin-Milwaukee, **(2006)** August.

[6] Saaty TL, "*The Analytic Hierarchy Process*", McGraw- Hill, New York, NY, **(1980)**.

[7] Goel AL, "Software Reliability Models: Assumptions, Limitation, and Applicability", *1EEE Transactions on Software Engineering*, vol. SE-11, no. 12, **(1985)** December, pp. 1411-1423.

[8] Musa JD, "*Software Reliability Engineering*", McGraw- Hill, New York, NY, **(1999)**.

[9] Arsanjani, Hailpern B, Martin J, Tarr P, "Web Services: Promises and Compromises", *ACM Queue*, **(2003)** March, pp. 48-58.

[10] Xie M, "*Software Reliability Modelling*", World Scientific Publishing Co., Singapore, **(1991)**.

[11] Zhang J, Zhang LJ, "Criteria Analysis and Validation of the Reliability of Web Services-Oriented Systems", in *Proceedings of the 1EEE 1nternational Conference on Web Services (1CWS'05)*, Orlando, Florida, **(2005)** July, pp. 621-628.

[12] OASIS: Web services business process execution language, **(2007)** April http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[13] Pisinger D, "Algorithms for Knapsack Problems", PhD thesis, University of Copenhagen, Dept. of Computer Science, **(1995)**.

[14] Sultana M, Akbar MM, Rouf M, "Network Flow Heuristic algorithm for a distributed web service selection problem", IEEE Pacific Rim Conference on Computers and Signal Processing, **(2009)**, pp. 465 – 470.

[15] Tang M, Ai L, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition", *2010 IEEE Congress on Evolutionary Computation (CEC)*, Australia, **(2010)**, pp. 1–8.

[16] Gao Z, Chen J, Qiu X, Meng L, "QoE/QoS driven simulated annealing-based genetic algorithm for Web services selection", Department of Network Management Product, Zhongxing Telecom Equipment Company, Nanjing 210012, China, **(2009)**.

[17] Ma SP, Kuo JY, Fanjiang YY, Tung CP, Huang CY, "Optimal service selection for composition based on weighted service flow and Genetic Algorithm", *2010 International Conference on Machine Learning and Cybernetics (ICMLC)*, **(2010)**.

[18] Liu D, Shao Z, Yu C, Fan G, "A Heuristic QoS-Aware Service Selection Approach to Web Service Composition", *Eighth IEEE/ACIS International Conference on Computer and Information Science*, Shanghai, **(2009)**, pp. 1184 – 1189.

[19] Yu T, Zhang Y, Lin KJ, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints", *ACM Transactions on the Web*, vol. 1, no. 1, Article 6, **(2007)** May.

[20] Arockiam L, Sasikaladevi N, "Estimation of the reliability of the composed web services-Service consumer Perspective", International journal on advance research in computer science, vol.3, no.1.

Authors

Dr. L. Arockiam working as an Associate Professor in the Department of Computer Science, St.Joseph's College, Tiruchirappalli, Tamil Nadu, India. Having 23 years of experience in teaching and 14 years of experience in research. Published 85 research articles in the International / National Conferences and reputed Journals. Presented two research articles in the Software Measurement European Forum in Rome. Authored a book on "Success through Soft Skills" and "Research in a nutshell".

N. Sasikaladevi, doing research in St.Joseph's College, She presented papers in various national and international conferences. Published papers in various journals. Authored a book titled "Programming in C#.NET". Her research interests are: Web Services, QoS issues on Web Services and Workflow patterns.