ISSN No. 0976-5697

Volume 3, No. 2, March-April 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Simulated Annealing Based Service Selection Algorithm for Composite Web Service

Dr.L. Arockiam
Associate Professor, Department of Computer Science
St.Joseph's College
Trichy, TN, India
larockiam@yahoo.com

N.Sasikaladevi*
Research Scholar, Department of Computer Science
St. Joseph's College
Trichy, TN, India
sasikalade@yahoo.com

Abstract: In the web service world, services are the basic amass that aims to support the building of business application in a more flexible and interoperable manner for enterprise collaboration. To satisfy 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 simulated annealing. In this paper, the simulated annealing 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.

Keywords: Web Service, Reliability, Workflow Patterns.

I. 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 simulated annealing based service selection algorithm is developed. The developed algorithm is compared with the heuristic algorithm based on time complexity.

II. RELATED WORK

Yi Xia et.all [1] derived a 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.

III. SERVICE SELECTION PROBLEM FOR COMPOSITE WEB SERVICE

The present web services architecture focus on solving the issues of service discovery, service selection and service composition. Service discovery is the process of finding or locating service implementations that meet a specified condition. In the same way, service selection is a process that deals with choosing a service implementation from the located services. From this, it is obviously seen that service discovery is a prerequisite requirement for selection process, but selection is the main problem that needs to be addressed for retrieving Web services successfully. For any service selection approaches the basic requirements include: Client service requirement, Service offerings by the service provider and aggregating the evaluation results [8].

The customer service requirement may be simple or complex. Simple requirement may not seem for composite services to satisfy the user query. Complex customer requirements may have both functional and non-functional aspects, which needs to be satisfied. For this sort of complex requirement, the services need to be composite. The composite service is a service formed by a composition of other available services. Google research application is acknowledged as a web service and integrated with other services, such as Gmail, AdWords, Picasa, Orkut, You Tube and Google Maps service, to provide an integrated environment for service consumers. The other known pattern for service composition is a tour booking service that can be formed as a web service and integrated with other services such as hotel booking, sightseeing, flight booking or car-rental in order to provide a collaborated environment for user[9]. However, the huge number of (tour booking) services which provide similarly functional characteristics may exist. Service consumers not only expect the service to meet functional aspects but they also require services to meet non-functional properties that are reliability and execution cost, etc. Thus the selection of services based on nonfunctional qualities achieves more advantages nowadays.

The services offered by service provider are anxious about functional and non-functional qualities of services. The functional properties make use of domain ontology. To provide consumer the requested service with non-functional properties makes use of QoS ontology.

The Web Service Selection process is broadly classified as Functional based approach, Nonfunctional based approach and User based approach. Functional based service selection approach represents the Static and Dynamic semantics. Selecting an appropriate service is concerned with retrieving functional descriptions from service repositories and then ensuring that the described and required interfaces match with each other. Static semantics represents the properties of messages and operation semantics. The properties of messages include parameter passed (Data type, language, unit and business role) and message types (Serviceability, provider type, purpose, consumer type). Dynamic semantics represents the properties of behavior and operation logic. With dynamic semantics in the service selection process of Web service, the resultant contains more than one service provider offering similar services [10].

With the swiftly growing number of available services, customers are offered with a choice of functionally similar services. This choice allows customer to select services that match other criteria, often referred to as non-functional attributes. Two fundamental questions crop up because of this: How can these extra attributes be described and how can one select the most appropriate service. These questions should tackle both the selection of isolated services as well as the selection of services within the context of other services. The non-functional based service selection represents the QoS and Context in semantic web service selection. The properties of QoS may be (security, reliability, response time, call cost etc.), the properties of Context may include context of customer (location, intention, consumer's name, application, e-mail, termination of hardware and software) and context of service (provider's details, service descriptions etc,). User based approach represents the selection of best service among plentiful discovered services based on customers' feedback, trust and reputation [11].

Nowadays, the encroachment in Web services requires intensification in the areas of service interoperation, discovery, selection, composition, choreography, orchestration and

mining. A possible solution to all these problems can be provided by converting Web services to Semantic Web. Semantic Web services (SWS) can provide a solution to the integration dilemma like composition. In universal, the semantics to be added to a Web service may be called as functional semantics. In Web services, functional semantic is taken into consideration thereby avoiding unsatisfied results which are not of customer interest. Functional property is the functional semantics of a service that describes what a service actually does. Web Service Selection is associated to the process of evaluating and ranking the discovered web services to identify the ones that accomplish a set of functional and non-functional properties requested by the service customer[12,13,14]. Most of the presented techniques rely on syntactic descriptions of service interfaces to find web services with disregard to semantic service parameters. This generates major problems in the service selection mechanism. To solve these problems, Web service descriptions are enhanced with annotations of ontological concepts, semantic matching and by considering non-functional properties.

A. 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) [6]. 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. The QoS service selection problem is mapped to MMKP as follows.

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

Mathematically, the service selection problem is formulated as follows:

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

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

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

where x_{ij} is set to 1 if atomic service j is selected for class S_i and 0 otherwise. $q_{ij} = [q_{ij}^1, \ldots q_{ij}^m]$ is the QoS resource needs of each atomic service j for class S_i; the sum of all resurces used by all service must be less that the overall constraints Q_c. The MMKP problem has been shown to be NP-complete [23]. We may solve MMKP by finding optimal results or use heuristic algorithms to reduce the time complexity

B. Service Reliability Estimation:

During the web service invocation, service consumer finds any of the following three statuses on the invoked web service and these statuses are come to most of the web services. The statuses are Continuous success, continuous failure and transitory failure. In the web service world, encountering continuous successes is the most widespread status. When examining the services' invocation records, we find that some services may encounter unbalanced failures. This status is further divided into two categories: first, a service consumer may encounter transitory invocation failures but successfully access the service in subsequent invocations; second, transitory invocation successes and failures happen alternatively. This status has been discovered in several services. We devise this status with less than continuous five failed invocations as transitory failures. Rarely services may encounter continuous failures during our study. Because the invocation records are collected at the service consumer side, this status presents the failures of either the service or network. These statuses show the runtime characteristics of service invocation successes and failures.

We assume that running web service may fall in any one of the following three statuses based on the invocation records. Invocation records stores the status information about the invoked web service.

a. Continuous Success (S): This status means that the service is running stably and no invocation failure occurs. Commonly services developed by reputable software enterprises are running in this status.
b. Transitory Failure (T): This status indicates that encountering failures during invocation is not predictable and it may be recovered by a simple "retry".
c. Continuous Failure (F): This status means that a service becomes down due to various factors. From the perspective of service consumer, they encounter continuous failures. This status is common for web services.

By using the above three statuses, we devise the new metric for evaluating the web service reliability. We evaluate how lone the particular service is in the above three states and based on this time value, we calculate the reliability rate for the web servie.

Service Invocation results are stored in the Service Invocation Registry. The Structure of the Service Invocation Record is given below,

$$R_{1n} = \{ V_{t1}, V_{t2}, V_{t3}, V_{t4} \ldots, V_{tn} \} \quad (1)$$

Where, R_{1n} is the Service Invocation Record for a specific service form time t1 to tn. V_{ti} is the Service Invocation Result at time ti. The Value is either 0 or 1. "0" refers failure and "1" refers Success of the service invocation. Set of service invocation records are stored in the service registry. The structure of the service invocation registry is shown below,

$$R = \{ R_{i1,i1}, R_{i2,i2}, R_{i3,i3} \ldots, R_{in,in} \} \quad (2)$$

Where, R is the registry of the set of services, $R_{i1,i1}$ is the Service Invocation record from time i1 to j1.

$$S = \{ S_1, S_2, S_3 \} \quad (3)$$

Where, S is the web service status result. S_1 is the continuous success status (S), S_2 is the transitory failure status (T), S_3 is the continuous failure status (F).

The input of the evaluation approach is a sequence of invocation records. The goal of the estimation approach is to calculate the service reliability values under different statuses. We have to identify service running status-based on invocation records and to calculate the reliability value under each different status. The estimation process involves,

a. Segmenting the invocation records
b. Estimating the values of availability metrics under each status.

C. Segmentation Algorithm:

//Given all the historical invocation records L1;n, we first divide them into m small fragments. The parameter m is set manually according to n. Each fragment contains an approximately equal number of records//

Input : $L_{1,n}$, k.
Output : L_v

a. Set number of segments as $\lfloor (n/k) \rfloor$
b. Examine $L_{1,n}$ orderly, put every $\lfloor (n/k) \rfloor$ records into a different sequence $L_{i,i}$. The remaining records are put into the last sequence. Then $L_{1,n}$ is divided into k fragments.
c. Repeat the following procedure until L_v remains unchanged. Examine all sequences in L_v orderly and comparing the records at the boundary of each sequence.
d. Examine all sequences in L_v orderly. Divide sequence $L_{i,i}$ into three sub-sequences. In this step, t is a manually defined threshold value to identify whether the service status is continuous down: when the length of continuous failures is more than t, then the service status can be regarded as continuous down.

D. Setting the Service Status Value:

Status of the individual services is identified by using the following rules.

a. If there is no service invocation failure in $L_{k,i}$, then this sequence is marked as status continuous success (S).
b. If there is no service invocation success in $L_{k,i}$, then this sequence is marked as status continuous failure (F).
c. Others are marked as transitory failure (T).

The Service reliability value is calculated based on the status as shown in the table 1.

Table 1 Service Status and its Reliability Rate

Service Status	*Reliability Rate*
Continuous Success (S)	1
Continuous Failure (F)	0
Transitory Failure (T)	Based on Weighted Average value

The service may have been in the transitory failure status for several times. For every time when the service runs in this status, there are several invocation records being recorded and the reliability value can be calculated as given below,

Where, N_a is the number of invocations and N_f is the number of failures. Weighted average value is calculated based on the given formula.

$$R_T = (N_a - N_f)/N_a \quad (4)$$

In the fragmented sequences L_v, there are many sequences marked with status transitory failure, and each sequence has a different success rate. To produce an overall evaluation for the success rate in status transitory failure, we devise a weighted average technique to calculate the overall success rate, and the weighting value is the time elapsed for each sequence. Finally we calculate the particular service is in each of the three states.

$$R_T^W = \frac{\sum_{i=1..t \wedge S_L = T^*} R_T}{\sum_{i=1..t \wedge S_L = T}} \quad (5)$$

The calculation of the time percentage of each status is based on Equation (6). In this equation, the denominator is the total time duration of the invocation and the numerator is the time duration of one of the three statuses where si denotes a status, such as "S", "T" or "F".

$$T = \frac{\sum_{i=1..t \wedge S_{L_{i,i'}} = s_i} |t_{ki'}, t_{ki}|}{\sum_{i=1..t} |t_{ki'}, t_{ki}|} \quad (6)$$

IV. A META-HEURSITIC ALGORITHM- SIMULATED ANNEALING FOR RELIABLE SERVICE SELECTION

Simulated Annealing is a universal optimization algorithm that belongs to the pasture of Stochastic Optimization and Meta heuristics. Simulated Annealing is an adaptation of the Metropolis-Hastings Monte Carlo algorithm and is used in function optimization. Similar to the Genetic Algorithm, it provides a basis for a huge range of extensions and specialization's of the general technique not limited to Parallel Simulated Annealing, Fast Simulated Annealing, and Adaptive Simulated Annealing. Simulated Annealing is aggravated by the process of annealing in metallurgy. In this usual process a material is heated and gradually frozen under controlled circumstances to increase the size of the crystals in the material and reduce their defects. This has the effect of

improving the potency and resilience of the material. The heat increases the energy of the atoms allowing them to move profusely, and the slow cooling schedule allows a new low-energy configuration to be discovered and exploited [6]. Each configuration of a solution in the search space represents a different internal energy of the system. Heating the system, consequences in a relaxation of the acceptance criteria of the samples taken from the search space. As the system is frozen, the reception criteria of samples are tapering to center on improving movements. Once the system has cooled, the configuration will represent a sample at or close to a global optimum. The information dispensation principle of the technique is to locate the minimum cost configuration in the search space.

The algorithms plan of action is to probabilistically re-sample the problem space where the acceptance of new samples into the currently held sample is managed by a probabilistic function that becomes more discriminating of the cost of samples it accepts over the execution time of the algorithm. This probabilistic finish is based on the Metropolis-Hastings algorithm for simulating samples from a thermodynamic system. Simulated Annealing of Metropolis could be adopted to solve problems in Combinational Optimization.

Simulated annealing (SA) [7] is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global optimum of a given function in a large search space. As its name implies, the Simulated Annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system.

In the simulated annealing (SA) method, each point *s* of the search space is analogous to a state of some physical system, and the function *E*(*s*) to be minimized is analogous to the internal energy of the system in that state. The goal is to bring the system, from an arbitrary *initial state*, to a state with the minimum possible energy. At each step, the SA heuristic considers some neighbour *s′* of the current state *s*, and probabilistically decides between moving the system to state *s′* or staying put in state *s*. The probabilities are chosen so that the system ultimately tends to move to states of lower energy.

Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. The probability of making the transition from the current state *s* to a candidate new state *s*' is specified by an *acceptance probability function* *P*(*e*,*e*',*T*), that depends on the energies *e* = *E*(*s*) and *e*' = *E*(*s*') of the two states, and on a global time-varying parameter *T* called the *temperature*.

One essential requirement for the probability function *P* is that it must be nonzero when *e*' > *e*, meaning that the system may move to the new state even when it is *worse* (has a higher energy) than the current one. It is this feature that prevents the method from becoming stuck in a *local minimum*—a state that is worse than the global minimum, yet better than any of its neighbors. On the other hand, when *T* goes to zero, the probability *P*(*e*,*e*',*T*) must tend to zero if *e*' > *e*, and to a positive value if *e*' < *e*. That way, for sufficiently small values of *T*, the system will increasingly favor moves that go "downhill" (to lower energy values), and avoid those that go "uphill". In particular, when *T* becomes 0, the procedure will reduce to the greedy algorithm—which makes the move only if it goes downhill. SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm employs a random search which not only accepts changes that decrease objective function f, but also some changes that increase it.

The following analogy was made between the simulated annealing of metropolis and combinational optimization.

Table 2 Comparison of Simulated Annealing of Metropolis with Combinational Optimization

Step	*Simulate Annealing of Metropolis*	*Combinational optimization*
1	Annealing looks for system state at a given temperature and energy	Optimization looks for feasible solution of the combinatorial problems
2	Cooling of the metal is to move from one system state to another	Search procedure (algorithm scheme) tries one solution after another in order to find the optimal solution.
3	Energy function is used to determine the system state and energy	Objective function is used to determine a solution and the objective function value.
4	Energy results in evaluation of energy function and the lowest energy state corresponds to stable state.	Cost results in evaluation of objective function and the lowest objective function value corresponds optimal solution.
5	Temperature controls the system state and the energy	A control parameter is used to control the solution generation and the objective function value.

A. Simulated Annealing Algorithm:

Algorithms provides a pseudo code listing of the Simulated Annealing algorithm for minimizing a cost function. Simulated Annealing algorithm for MMKP problem to optimize the cost is given below.

Optimization Problem:

$\min f(x)$ such that $x \varepsilon S$ where S is a feasible solution

Parameters:

X^i	Solution State
$F(x)$	Objective Function
k	Iteration number
$\delta=f(x^1)-f(x^0)$	Energy change between the states
T	Control parameter
$g(T)$	Control parameter function
$e^{-(\delta/T)}$	Choice probability function

Procedure:

It provides the condition under which a non-improvement solution is not discarded.

Step 1: Selected initial solution assign $x^b = x^0$
Set iteration k=0
Selected initial control parameter $T_k = T_0$
Selected control parameter function $g(T_k)$

Step 2: Choose a solution x^1 in $N(x^0)$
compute $\delta=f(x^1)-f(x^0)$

Step 3: If $\delta=0$ or $\delta>0$ and $e^{-(\delta/Tk)}$, accept the solution x^1
Assign $x^0 = x^1$

Step 4: If stop criteria satisfied, then STOP

Step 5: Update the control parameter, $T_{k+1} = g(T_k) \leq T_k$ set k=k+1

B. Service Selection Algorithm Using Simulated Annealing:

Simulated Annealing is implemented to solve the MMKP with static cooling schedule. Here, a reasonable strategy is used to generate random solution in the neighborhood, the relevant parameters concerning cooling schedule are tested 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 neighborhood function for $\omega \in X$. Simulated annealing starts with an initial solution $\omega \in X$.. A neighboring 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, fiite-time implementations of simulated annealing algorithm are considered, which can no longer guarantee to find an optimal solution, but may result in faster executions without losing too much on the solution quality.

a. Initial Solution:

Initial solution is generated by randomly choosing 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.

b. Exploring Neighbor Solutions:

At each iteration, one group is randomly selected. If it is already 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 approximate solution. On the other hand, if the randomly selected group is not already 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, feasibility of the approximate solution is checked.

c. Acceptance Probability:

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

P{Accept ωas next solution} =

$$\begin{cases} \exp[(f(w') - f(w)/t_k] & if \quad f(w') - f(w) < 0 \\ 1 & if \quad f(w') - f(w) \geq 0 \end{cases}$$

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

d. 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(ω)*;
Repeat;
Set repetition 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 infeasible, **do**
Drop another group from *ω'* randomly; denote the new solution as *ω'*;
Let Δ = f(*ω'*) – f(*ω*);
If Δ ≥ 0 or Random (0,1) < exp(Δ/*t*), **then** *ω* ← *ω';*
Else
Drop group *i*, and pick up 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, obtain new solution *ω'*, **then**
While solution *ω'* is infeasible, **do**
Drop another group from *ω'* randomly; denote the new solution as *ω'*;
Let Δ = f(*ω'*) – f(*ω*);
If Δ ≥ 0 or Random (0,1) < exp(Δ/*t*), **then** *ω* ← *ω';*
End If
If approximate solution < *f(ω),* **then** approximate solution ← *f(ω);*
m = *m* + 1;
Until *m* = *M*;
Set *t* = *α* * *t;*
Until *t* < *e;*

A set of parameters needs to be specified that govern the convergence of the algorithm, i.e. initial temperature *t*0, temperature control parameter *α*, final temperature *e*, and Markov chain length *M*, in order to study the finite-time performance of simulated annealing algorithm. Here *t*0 should be the maximal difference in cost between any two neighboring solutions, let *t*0 be 500 by rough 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.

e. 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 in pseudo code

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.

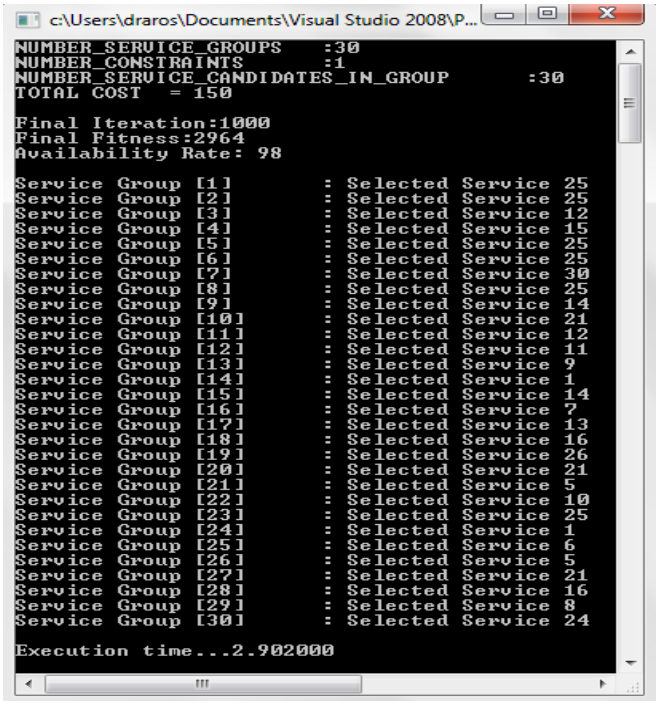

Figure 1 Screen Shot of Service Selection Algorithm

V. EXPERIMENTAL RESULT

Web services which are relevant to students information processing are collected from web. Among these web services, the 1000 most relevant 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 100. Reliability status is calculated on each fragment and time percentage is calculated using the equation (6).

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.

The simulated annealing 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 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.

Table 3 Test cases for service selection problem

Test Cases	*Service Groups*	*Service Candidates*	*Exe. Time (Seconds)*	*Optimality of Results*
1	5	2	0.092	97
2	5	4	0.187	97
3	5	6	0.234	97
4	5	8	0.218	97
5	5	10	0.296	98
6	5	12	0.312	98
7	5	14	0.312	98
8	5	16	0.342	98
9	5	18	0.359	98
10	5	20	0.406	98
11	10	2	0.125	98
12	10	4	0.187	98
13	10	6	0.265	98
14	10	8	0.327	98
15	10	10	0.39	98
16	10	12	0.437	98
17	10	14	0.514	98
18	10	16	0.592	98
19	10	18	0.655	98
20	10	20	0.733	98
21	15	2	0.202	97
22	15	4	0.218	98
23	15	6	0.39	98
24	15	8	0.483	98
25	15	10	0.593	98
26	15	12	0.655	98
27	15	14	0.78	98
28	15	16	0.858	98
29	15	18	0.936	98
30	15	20	1.17	98
31	20	2	0.234	97
32	20	4	0.312	98

33	20	6	0.531	98
34	20	8	0.577	98
35	20	10	0.78	98
36	20	12	0.858	98
37	20	14	0.983	98
38	20	16	1.123	98
39	20	18	1.217	98
40	20	20	1.326	98
41	25	2	0.249	97
42	25	4	0.328	98
43	25	6	0.608	98
44	25	8	0.717	98
45	25	10	0.92	98
46	25	12	1.029	98
47	25	14	1.17	98
48	25	16	1.357	98
49	25	18	1.497	98
50	25	20	1.67	98

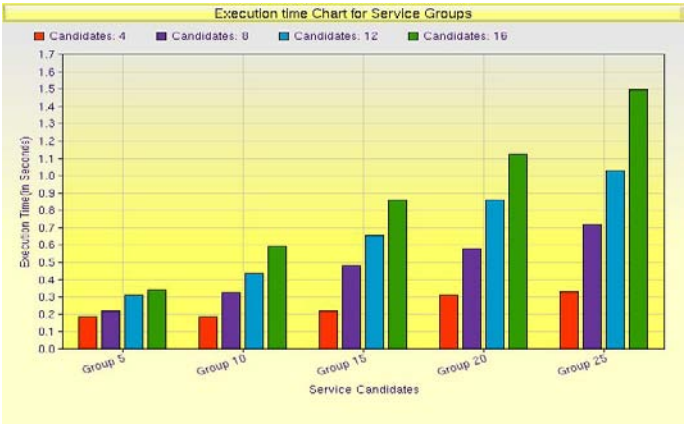

Figure 2 Execution time chart of SA algorithm (Service Group ranges from 5 to 25)

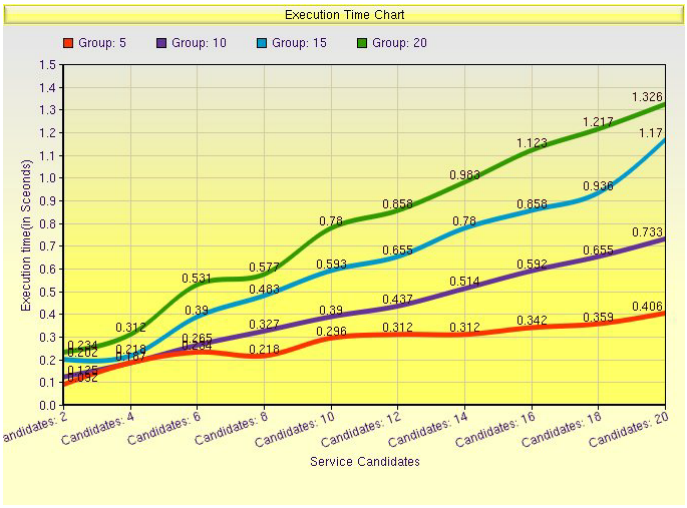

Figure 3 Execution time chart of SA algorithm (Service Candidates ranges from 2 to 10)

Figure. 4 Execution time chart of SA algorithm (Service Group ranges from 5 to 25)

VI. PERFORMANCE OF SIMULATED ANNEALING ALGORITHM

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. Time complexity of t this algorithm is analyzed. The execution time of Simulated Annealing algorithm is calculated and shown. The Fig.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.

VII. 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 simulated annealing 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.

VIII. ACKNOWLEDGMENT

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

IX. 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] 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.

[7] 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.

[8] 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 Systems, pp. 336 – 343, 2008.

[9] 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.

[10] 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

[11] 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

[12] 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.

[13] Zhang Kun; Zhang Hong; Jiang Liming; Xu Jian; "Composite Agent Service Selection Algorithm for Non-functional Attributes Based on Simulated Annealing", 2010 Second World Congress on Software Engineering (WCSE), 101 – 106, 2010.

[14] 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.

Short Bio Data for the Author

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. Chaired many technical sessions and delivered invited talks in National and International Conferences. Authored a book on "Success through Soft Skills" and "Research in a nutshell". His research interests are: Software Measurement, Cognitive Aspects in Programming, Web Service, Mobile Networks and Data mining.

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.