Journal of Web Engineering, Vol. 13, No.5&6 (2014) 430-449

AN OPTIMAL CONSTRAINT BASED WEB SERVICE COMPOSITION USING INTELLIGENT BACKTRACKING

M. SURESH KUMAR P. VARALAKSHMI

Department of Information Technology, Anna University, Chennai
suresh.priya.kumar@gmail.com varanip@gmail.com

Received August 9, 2013
Revised August 1, 2014

Composition of web services involves a complex task of analyzing the various services available and deducing the most optimal solution from the list of service sequences. The web services are viewed in the form of layers interlinked with each other based on some conditions to form a service composition graph dynamically. Layering of the web services is done based on a sequential arrangement of the services as designed by the web service provider. From the numerous service sequences available, the most optimal service is computed dynamically from the start to end of a web service composition. The optimal solution set, consisting of a number of services, is deduced as the path that has the least total weight from start to end of the service composition. The anomalies that might arise in the search for optimal solution are solved using the Intelligent Backtracking technique thereby eliminating any absurd problems. The idea of Intelligent Backtracking is to make the optimization more efficacious. Dependency-directed backtracking is used so that the past transaction records are saved, making it easier to track the flow of web service selection. A Log file concept is introduced to keep a record of the service transactions at each level in order to satisfy user constraints in the best possible way. In case the user constraints are not feasible enough to complete the composition of services, then based on the data in the log files, negotiation can be done with the user for the reselection of certain anomalous web services. Negotiation is a process of dynamic mediation with the user in case his requirements constraints cannot be satisfied with the list of services provided by the service provider. This concept, if put to use will be revolutionary as it not only helps achieve optimization but also enriches the QoS constraints for user satisfaction.

Key Words: Web service, Web Service Composition, Service Composition graph, Intelligent Backtracking, Optimization, Log File, dynamic mediation, service provider, value ordering.
Communicated by: B. White & R. Mizoguchi

1 Introduction

Web services are rapidly emerging as the most practical approach for bridging distributed and heterogeneous applications. As the applications become more and more complicated, a single web service becomes inadequate to satisfy the need. Thus, we integrate the pre-existing services, producing a value-added service to fulfill the requirements, bringing web service composition into play. A web service composition provides an open, standard-based approach for connecting web services together to create higher-level business processes and thereby providing a user friendly interface to complex sets of services. Some concepts have been designed to reduce the complexity required to compose web services, hence reducing time, cost and increasing overall efficiency in business. One such standard in depicting the web services is the graphical representation of services. There are several contributions addressing the composition of Web Services. A growing interest can be noticed in this field, because the composition of Web Services can enhance the execution of workflows. The concept in the paper comprises of standards like WSDL (Web Service Description Language), BPEL4WS (Business Process Execution Language for Web Services), WSLA (Web Service Level Agreements). The Web Services Description Language (WSDL) is an XML-based interface description language that is used for describing the functionality offered by a web service. A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. It thus serves a purpose that corresponds roughly to that of a method signature in a programming language.

A web Service Composition Graph (SCG) has been used to represent the web service composition. It consists of layers of services as listed by the service providers. The service composition graph is used as a standard for easier computation of an optimal solution in achieving user satisfaction. Linear arrangement of the layered web services is done in order to denote a sequence of web service selection avoiding complexity in optimization. Value ordering technique and intelligent backtracking are used to improve the efficiency of the service selection process.

Web service composition is done by a minimal total weight path algorithm which will identify the path with atmost user satisfaction and least total weight in the set of web services. This algorithm is efficient when coupled with the web services composition graph as the concept of linear arrangement in a service relationship will help to achieve a path like structure for the entire web service composition. Optimal solution is deduced at each layer based on the weight and set of optimal solutions obtained for the entire composition of web services. The proposed system is clearly explained by means of a composite travel web service as an example with the optimal resources being the result.

The proposed web service composition system is basically organized into the following sections. Section-II deals with related work of the proposed system. Section-III explains the workflow of the proposed system. Section-IV deals with the service composition graph generation and different composition patterns. Section-V deals with weight assignment for web services and ordering of web services. Section-VI comprises of Web service selection and composition and handling the service unavailability. Section-VII concentrates on customer satisfaction through maintaining the compromising log and negotiation. Section-VIII discusses the implementation of the proposed concepts through various scenarios. Section-IX deals with conclusion and future work that intended to be carried out.

2 Related Work

The method of providing an optimal solution for the consumer using web services is proposed by various researchers. In the QoS aware web composition problem, it is very much essential to provide fault tolerance in case of service unavailability. The proposed system provides fault tolerance through Intelligent Backtracking methodologies and to achieve the most optimal solution [3].

In the work of Hatzi et al. [2], which is based on Artificial Intelligence planning, is a very good approach in case of a Web service composition but it is a practical thing that the planning may not

work out always and may lack its dynamic nature. The proposed work mainly concentrates on dynamic graph generation through multi stage graph and layer ordering and maintaining the compromising log and negotiation in case of inability of user to satisfy the requirement of web service.

The problem of thrashing and redundancy can lead to less probability of getting an optimal solution to the web system. The system will be able to avoid thrashing and redundancy through an Intelligent Backtracking approach [10].

In most of the web services composition proposals [8 9], the concept of Intelligent backtracking and Log files are not in much use but in the proposed work we make use of the Intelligent backtracking to provide better user satisfaction and fault tolerance capability of the system.

In the works related to graph, the concept has not been used as a full -fledged model. Rather just an implementation part is worked upon with a single pipeline model [17]. Our paper entirely focuses on the complete representation of the web services in the form of a graph with the concept of dynamic programming to provide best optimal solution.

In the work by Cao et al [15], the algorithm used is the genetic algorithm [9 11]. The major drawback of this system is the survival of the fittest value which consumes a longer time and this paper does not deal with the fault tolerant manner for web service which is most concentrated in the proposed work.

3 Proposed Work

The proposed work of the web service selection and composition through graph generation and dynamic programming can be explained with ease using the proposed composite web service architecture of the system as shown in Fig 1.

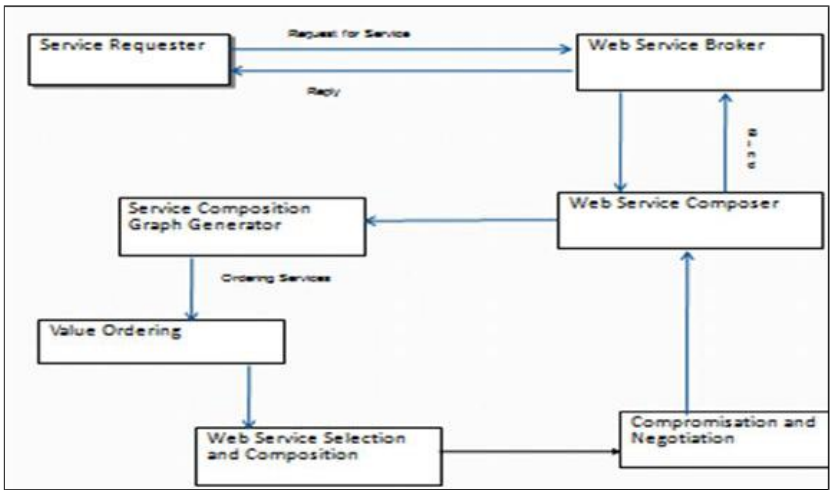

Fig 1. Proposed Composite Web Service Architecture

The service requester i.e. the user requests for a service from the web service broker for better Web service composer. Web service composer helps the user in selection and composition of optimal web service. In the proposed system, we make use of backtracker and maintaining the compromising log along with value ordering block to assure user satisfaction to the maximum extent and provide them the most optimal solution. The web service composer has the following steps: the first step is the Service Composition Graph (SCG) generation by assigning the weights for each edge based on the ranking algorithm. After the SCG generation, the services of each layer are ordered by thei r value

with each layer. During the selection of web service in a layer, the service is checked for availability. If the service is not available, the system backtracks to choose the next optimum solution for the user. The compromising log node maintenance is used to enhance the user satisfaction to a greater extent. If a service in layer is not feasible to be provided for the user due to some constraints the proposed system has the provision of compromising log which maintains a list of unsatisfied constraints for a service in a layer. Finally after composition of services of all layers, the compromising log is checked for any compromisation to be met. If so, then the negotiation with the user comes into play. So, the user is satisfied most of the times with the best optimal solution.

4 SCG Generation And Composition Pattern Of Web Services

Generation of web Service Composition Graph (SCG) and Composition Pattern of web services in each layer of SCG are discussed in the following paragraphs.

4.1. Generation of Service Composition Graph

The proposed work involves the implementation of a linearly arranged web service composition graph. This web service composition consists of various services available in a set of different layers depicted in the form of a specially designed web Service Composition Graph. The main principle behind the layer concept is that the services accessed in composite web service collection is done sequentially with a series of service selections performed one after another till the web service is completed in execution. Therefore, each layer in the composite web service graph consists of set of services represented as candidate services of a particular type of web service (for example, flight booking service) provided by the various service providers.

The layers are depicted in a linear arrangement which is done to provide a frame work for the otherwise complex web service composition. By this, we can specify a sequential order in which web service selection has to be done through each and every layer. An optimal web service selection path will have to select a service from all layers of the web service composition graph.

The web service composition graph is denoted by G. The graph creation is represented as a combination of layers and relationships between the layers.

$$\mathbf{G = (L, R)}$$

In the above representation, G is the web services composition graph, L is the layers of services available and R is the relationship between the available services. The web service composition graph follows 2 basic conditions:

- The services are partitioned into ‘n’ number of composite service layers.
- A relationship R between two web services S1 and S2 is possible only when S1 is in Lith layer and S2 is in Li+1th layer .i.e., S1 ϵ Li and S2 ϵ Li+1.

The web service composition graph creation Pseudo code is specified in the Fig.2.

Fig.3. shows a fully created web service composition graph that consists of n layers of services with a starting point of service composition S, which is the initial state where the use r sets his requirement constraints for the selection of web services. The termination point T is the state at which the user completes his web service selection by satisfying most of his requirement constraints. In between the initial state S and the final state T, the user has to undergo service selection through each

set of service layers. Services are ordered into layers with each layer containing similar type of services.

```
void Graph_Creation(L,R)
{
 S is initial
 state (L0) T is
 final state
 (Ln) IF K>=
 2
 FOR i in k
 {
 CREATE a service layer
 List all services in that layer
 IF Si ϵ Li is related to Sj in Li+1
 THEN Si →Sj
 ELSE
  Si →Sj cannot be in R
 }
 ELSE
 Message as Web service composition graph cannot be
created for k<2
 }
```

Fig.2. Service Composition Graph Creation Pseudo code.

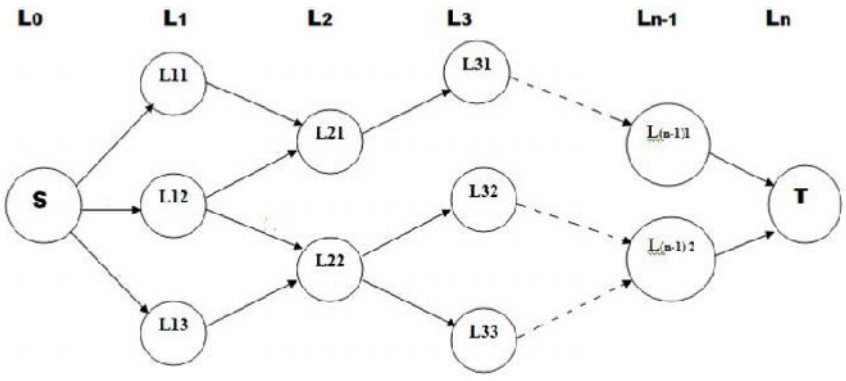

Fig.3. Web Service Composition Graph

These layers are linearly arranged depending on the service list as designed by the service providers. Services in a layer L_i is connected to services in layer L_{i+1} by a weighted relationship. The

weight of a relationship between the services is considered to be the possible relationship between the services across the layers and the weight of the service selection depicts the priority.

There are many patterns to be dealt in the composition of web services [18] in which each of the services could be of the following types.

1. **Sequential** : The composition model can have two sequential web services in which two services are executed in a sequential manner.[Sequential] . For example, services H and I can be executed in sequential structure as shown in Fig.4.

2. **Loop** : The next type of composition model can have a loop in which certain web services can be executed iteratively for certain duration of time [Loop] . For example, the web services D, E G and D can be executed iteratively in loop as shown in Fig.4.

3. **Parallel** : When two or more execution paths have common starting and ending services and parallelism relations exist between the services in different paths, we identify these paths as in a parallel structure. For example, the execution path along services D, E, G and the execution path along services D, F, G are converted to a parallel structure because services E and F are in a parallelism relation. Services E and F can be executed concurrently in any order and both need to be completed before performing service G as shown in Fig.4.

4. **Alternative** : Multiple possible execution paths are executed under a certain condition, i.e., only one path is executed at one time. When multiple execution paths branch out from one service, the execution paths can be either in a parallel structure or in an alternative structure. If these paths are not identified as in a parallel structure (i.e., no parallel relations exist between different paths), we identify it as in an alternative structure. For example, the path along services A, B, D and the path along services A, C, D are recognized as an alternative structure as shown in Fig.4.

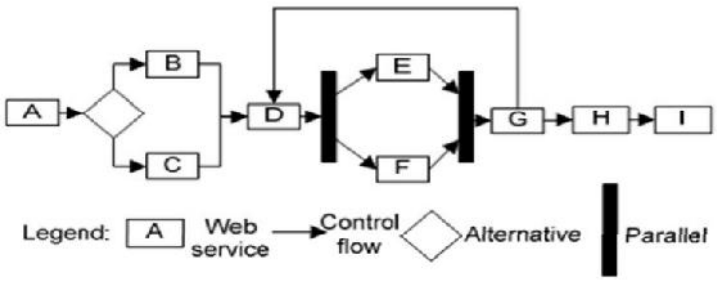

Fig 4. Service Composition Pattern with control flow structures

5 Weight Assignment for Web Service Composition Graph Using a Service Registry:

The real principle behind assigning weighted values based on priority is that it will help in satisfying the user constraints because the prioritized weight value is assigned to the web service edges based on the user constraints set. Prioritized weight value means that the services are ranked by the ranking algorithm and assigned weight based on their proximity to the user constraints with the least weighted service being the best user satisfied web service. The ranking algorithm will rank each of the web service in a layer based upon the user constraint set for that layer by the user. Thus, all the available web services in a layer will be ranked in a particular order for a given constraint value and the weights

for each of the web services will be assigned in increasing order for the ordered services.

The ranking algorithm also has the feature to handle with situations where a concrete constraint is not set by the user or the user does not intend to have any constraints for choosing services in the layer wherein all the services in that layer are assigned equitable weights since no preference or constraint is set.

For Example, in our travel planning service composition set, we have the initial layer to be flight booking services and the description of all the available web services are maintained in the registry along with description of each of the service with certain description fields such as travel fare and luxury factor decided upon by review statistics and customer preference decided upon by customer reviews etc. The user is given with the option of setting up constraints from among the description fields that is defined for a particular layer of web services. If the user sets the constraint for flight booking as fare of $5000, then, the flight services exactly having fare description field as $5000 will be ranked first and given least weights. The flight services of fare with close proximity to $5000 will be ranked successively and the weights will be assigned increasingly. Similar ranking and weight assignment is done for the weighted services in each layer whether or not the user sets any constraint and a complete web service composition graph is generated with service relationship paths and weight. For each stage of service selection, all possible paths are analysed and the minimal total weight path is deduced. The minimal total weight path at the final layer of service selection is considered as the optimal path with a set of web services selected at each layer for the user constraint. Here, the word optimum refers to the web services set that most satisfies the user constraints as user satisfaction is the primary motive of our proposed work.

The detailed algorithm for estimating the total minimum weight path algorithm is given in Fig.7. The main objective of the algorithm is to just estimate the total minimum weight path of the web service composition graph as this will give the best set of services satisfying the user requirements because the services are already weighted according to the user preference by the ranking algorithm specified in Fig 5. The optimal solution is deduced as a path from S to T with least total weight from among the various paths available in the composition graph to complete the web service composition. Here, the total weight factor is considered as the condition to be satisfied for the optimal solution. The services that have a relation with the least weight in each layer being the optimal solution that intends to satisfy the user constraints best in that particular layer.

1. Set up a Service registry consisting of all web services layer-wise with description.
2. Maintain a log file to store information about constraints available for user to be set for each layer.
3. For each value of constraint set by user from log Do
i) Check each Web Service description for corresponding constraint value
ii) Order them with respect to the value in a descending manner
iii) Rank the web services in that order
iv) Assign weights in increasing order starting from 1
4. If No user constraint is set then
assign equal weights for all web services of that layer
5. Else
go to step 3
6. Exit

5.1. Value Ordering Of Web Services

The Web services in each layer need not be in order of the user constraints, so backtracking technique in case of unavailability of web service need not provide the user satisfied web service. In order to enhance this feature of backtracking we make use of value ordering to perform the intelligent backtracking incase of situations of services not being available. In this value ordering technique, we

just order the web services in each layer with decreasing order of optimum service .

Optimum service is identified by the weights assigned to each edge based on the user constraints. The Algorithm for showing the Web service Value ordering is shown in Fig 6.

```
Initialize S1,S2,…Sn
 Set constraint=user_constraint
Set S1,S2…Sn as available
Void Service_order(S,C(s))
{
 FOR i in 0 increment i Till i<n
{
 For j in i+1 increment j Till j<i
{
                if(C(Sj) <C(Si))
              {
                    CREATE Sk
                    EQUATE Sk=Si
                    EQUATE Si=Sj
                    EQUATE Sj=Sk
               }
          }
}
```

Fig 6. Ordering of Web Services in Layer of Services of SCG

```
void Service_Select( G, k, n, p[])
// The input is a k-stage web service composition graph G
// services are indexed in layers
//R is a relationship between services
//c[i][j] weight of path
//p[1:k] minimum total weight path
{
FOR j in n-1 decrement j till j=1
{
Let s be a service such that <s,j> is in relation
such that c[j][s] + weight[s] is minimum and
c[j][s] + weight[s] <=budget
weight[j]= c[j][s] + weight[s]
d[j]=r
}
//Find minimum weight path p[1]=1
p[k]=n
FOR j in n increment j p[j]=d[p[j-1]]
}
```

Fig 7. Minimum Total weight path algorithm

6 Optimal Web Service Selection And Composition

Optimal service selection and no-good service selection are explained in the following paragraphs. A dynamic approach is being employed in the web service selection of the services from the composite setup. The completion of a web service selection will be done only when a service is selected from each layer in the web service composition graph computed.

The algorithm to select the optimal solution through the minimal total weight path algorithm is given in Fig 7.The weight assignment for all the service paths done by the ranking algorithm is the main component of deducing the optimal solution of web services. The minimum total weight of the algorithm is devised so as to identify the web services that fall in the path that constitutes the minimum total weight. The algorithm begins with the start node and searches each web service in a layer and at each layer the web service with the least weight path is selected and proceeds to the next layer. Then at each layer the least weight path is searched and a web service is selected. At each layer, the total weight path is checked for all the paths and the least path is proceeded upon till the final layer of web services is reached and the path that has the least total weight is selected as the optimal solution. The web services present in the least weighted path represent the set of optimal web services best satisfying the user constraints.

6.1 Optimization Of Service Selection

Performance of brute force backtracking can be improved by using a number of techniques such as variable ordering, value ordering, back jumping, and forward checking. The order in which variables are instantiated can have a large effect on the size of the search graph. The idea of variable ordering is to order the variables from most constrained to least constrained. The order in which the value of a given variable is chosen determines the order in which the graph is searched. Since it does not affect the size of the graph, it makes no difference if all solutions are to be found. If only a single solution is required, however, value ordering can decrease the time required to find a solution. In general, one should order the values from least constraining to most constraining in order to minimize the time required to find a first solution. Back jumping is that when an impasse is reached, instead of simply undoing the last decision made, the decision that actually caused the failure should be modified.

When a variable is assigned a value, the idea of forward checking is to check each remaining un-instantiated variable to make sure that there is at least one assignment for each of them that is consistent with the previous assignments. If not, the original variable is assigned its next value, this approach is commonly known as Intelligent Backtracking. Thrashing and redundancy problem can be avoided by intelligent backtracking scheme on which backtracking is done directly to the variable that caused the failure.

In the proposed system we make use of these intelligent backtracking algorithms to overcome the service unavailability problems and to provide an optimal solution for the user. Dependency - directed backtracking approach is made use of in the proposed system which prunes the set of no - goods services and directly selects the optimal web service which reduces the execution time for service composition. The dependency-directed method is considered for backtracking because each and every service of a layer is dependent on some set of services in the next layer.

If the Web services had made use of backtracking algorithm and a service is found unavailable, then it checks each and every service for getting an optimal solution to the user which is highly time consuming. The service unavailability here is expected as services due to some delay of services or some other reasons which could be predicted well in advance. The intimation of service availability or unavailability is given by the web service provider to the broker. The Fig 8 shows the sample Web-service composition in which the circles represent the web services and the edge represent the priority assigned to the service in particular layer based on the comparison with user constraints, WS represents the web services and the line over the circle indicates that the service is not available. For

instance in the Fig 8.a, If the Web Service WS1 is found unavailable and normal backtracking is used to choose an alternate web service WS3 for customer C1 after checking WS2 service priority which is found not to be the best fit of service for the user constraint which generally becomes time consuming. Instead if the service of the layers is ordered before selection, the service selection becomes easier for selection even in case of service unavailability. If the services are not available at the particular instant specified by the user then those services are added to the list of not the good set, so that after certain amount of time another customer C2 comes up with same request, the no good set services are ignored for selection directly and it selects the next service. This reduces the time of getting the optimal solution unlike normal backtracking algorithms. So, if we order the servi ces based on priority as in Fig 8.b, and if the Dependency-directed intelligent backtracking algorithm [5] is used in case of unavailability of service for customer C1, then the service are not considered for any future customer C2 till it becomes available. Thus, the proposed system can provide the best service for customers in a short period of time.

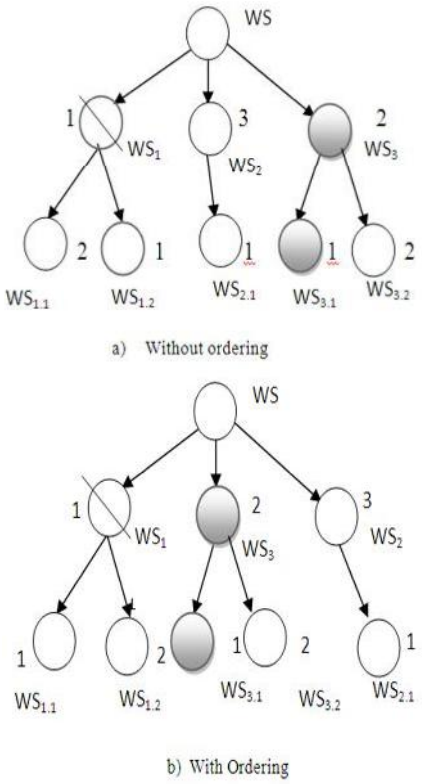

Fig 8. Web Service Composition Using Dependency Directed Back-Tracking

6.2 Mathematical Model

The following steps are followed to derive the mathematical model as follows:

i. Dividing Stages: K=1,……..,n, each type of web service can be considered as a stage, including start node and end node. (Like Flight Booking, Hotel Booking etc..)

ii. Setting states: Each stage has several nodes (used to represent a web service) associated with it and name it as candidate services. Each stage K has L(i), i=1,2,……,m candidate services.

iii. The cost of each candidate service represents the weight of the service (c).
iv. The utility of candidate service produces represents the benefit of the service (b).
v. The Objective is to maximize the overall utility produced by the composite service under the constraint that the total cost of selected services ≤C (i.e the budget of the user).

QoS of the web service composition based on various QoS parameters is derived as in Eq (1).

$$Q(s) = \sum_{i \in SA} (w_i * Score_i(s)) \quad (1)$$

where, Q(s) is the QoS Value of Service, Scorei(s) is an attribute scoring function Score that, given the value of an attribute i of the service s, returns a positive number. SA is the set of selection attributes, such as execution time, execution price, reliability and so on, wi is the weight w corresponding to attribute i, which can be given by customers.

In short, the mathematical function is derived as

$$\text{Max} \sum_{i=1}^{k} \sum_{j \in Si} F_{ij} x_{ij}$$

$$\text{Subject to} \sum_{i=1}^{k} \sum_{j \in Si} c_{ij} x_{ij} <= C \quad \text{where}$$

$$\sum X_{ij}=1,\ i=1.....k,\ j \in S_i$$
$$X_{ij}\{0,1\},\ i=1.....k,\ j \in S_i$$

where, F_{ij} is the Utility value at stage i for candidate j, c_{ij} is the Cost of candidate j at stage i, x_{ij} is the Validity function (Service available or not) and C is the Total Cost affordable by the user. Utility function (F) of Optimal Web Service Selection is calculated as in Eq (2).

$$F(s,1) = w_b * (b(s,1)-avgb)/stdb + w_c * (1-(c(s,1)-avgc))/stdc \quad (2)$$

where

w_b - weight of the benefit, $0< w_b<1$
w_c - weight of the cost, $0<w_c<1$ and $w_c=1-w_b$
b(s,l) - benefit of using service s at level l
c(s,l) -cost of using service s at level l
avgb -average benefit for all services and levels
avgc -average cost for all services and levels
stdb -standard deviation of all benefits
stdc -standard deviation of all costs.

Here, the affordable cost of the web service alone is considered as QoS parameter. In our future work, other QoS parameters will be considered.

6.3 No Good Services

The Dependency-directed backtracking concept can be implemented more effectively for reducing the Web service selection time by pruning the services that are no -good during the Web service selection. Consider, an example of Travel Web Service selection in which if the customer puts a request of user for selecting a flight service in layer L1 with least cost and a hotel service for the layer L2.

Consider the example Fig 8 b, If the flight service WS1 is selected as the least cost flight but unfortunately there is a time lag in departure which makes the hotel services WS1,1 and WS1,2

unavailable due to the exceed of check-in time. In the scenario mentioned above, the use of dependency-directed backtracking will prune the search of WS1 and it directly goes to the next optimal service WS2 selection.

```
//Initialization step before Service Selection of layer Ln with K-services
SELECT a Layer Ln
SET no-good services set as empty
//Repeat the following step after setting up the no-good sets based on service availability
FOR i in 1 Increment till i<K
{
IF (Si Λ no-good services=Φ) OR (Si is available)
{
IF Si is not available
 {
ADD Si to no-good services set
REPEAT the loop with service Si+1
}
ELSE IF Si is available
{
OPTIMUM SERVICE for Li is Si
SELECT Si FROM Ln
PROCEED to service selection of Ln+1
layer
}
}
```

Fig 9: No-good Services ignorance Algorithm

This seems to be very advantageous in scenarios of more than 10 services becoming unavailable due to some constraints or unavoidable situations. In such a situation, normal backtracking takes 10 times the time of normal selection to select the 11th service available but by this non- chronological approach system, it will be able to prune the 10 services as it is available in no-good set. The use of Dependency-sets analysis by means of Dependency-directed backtracking marks the set of services as no-good. Either of the below conditions are checked before the service is selected.

i. Check for the service is available at that instant
ii. Check if the service to be selected is not in the no-good set.

The algorithm for No-Good check is done by No-Good services ignorance algorithm as shown in the Fig 9.

The Fig 10 shows the sample workflow of the proposed system's web service selection and composition strategies.

Initially the SCG generation and ordering of web services is done as mentioned in Section IV. After the generation of SCG, the web services selection and composition is done dynamically based on user constraints by incorporating the techniques of Dependency-directed Back Tracking and compromising log maintenance for Efficient and optimal web service selection and composition.

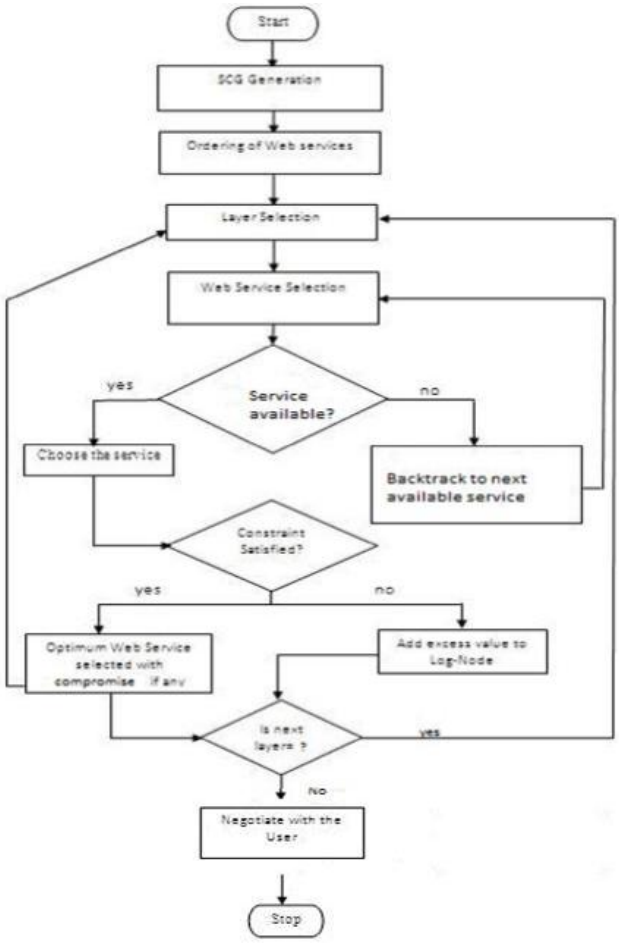

Fig 10.Workflow Of Web Service Selection and Composition

7 Compromisation and Negotiation of Web Services

The main aim of User Satisfaction is achieved in the proposed system by means of compromising log maintenance. The system maintains a list of nodes which stores the value of service in each layer that is compromised for the sake of user. In the T ravel Web Service Composition, consider the scenario in which the user needs to select a cheaper flight service in layer L1 and the constraint made by the user is C1 But the Flight services starts only at C0 such that C0>C1 and in the next selection of hotel service of layer L2 user is affordable for a costlier and luxurious hotel. In most of the web service compositions, the scenario above goes back to the user for negotiation but in the system which is proposed keeps the User satisfaction into consideration, we provide some negotiation by storing the excess amount of C0-C1 in the log node corresponding to the particular layer in the linked-list of nodes which is referred during each service selection in a layer. If there is any balance that is to be met by the user, then the amount is compensated in the next service selection of hotel booking by scanning through the list of log nodes for any compensation that can be made to satisfy the user constraint. The loss incurred by the service in layer L1 can be transferred directly from the layer L2.

The creation of linked-list of log nodes plays a major role in the compromise and negotiation with the customer if the user's constraint is not affordable by the service. The system maintains a log node for each layer to drop down all the pending value of constraint like cost, time etc as shown in the diagram Fig.11. These log nodes are checked at every stage of Web service composition.

The following steps are followed for compromisation and negotiation:

- When the nodes with services are traversed through each and every layer of service composition graph, the condition R(c) <C (WS) is to be checked for any pending amount in a particular layer and is to be noted down in the list of log nodes, where C(WS) is the cost of web service to be selected and R(c) is the user preferred cost for the particular layer of web service.
- This pending value of cost is written down into the log node corresponding to the particular layer in the linked-list in each layer selection.
- This process continues for all web service selection but finally after the selection of service for the last layer, the log node is verified for the pending cost.
- If the value in the log node after the final web service selection is zero or positive, then the job is successful and user is returned with composite web service.
- If the log node value is negative, then compromisation is undergone whether the additional amount can be made by the user.
- If the user is not ready to bear the additional cost, then backtracking using the skip-list concept is done to go for the layer of web service which is based on user's option for the selection of particular web service for compromisation and finally the alternative web service is selected from that layer.
- Once the negotiation process is over, the job is finished and user is returned with composite web service.
- If the selection of web service cannot be made to negotiate with the user, job is cancelled leading to unsuccessful job completion.

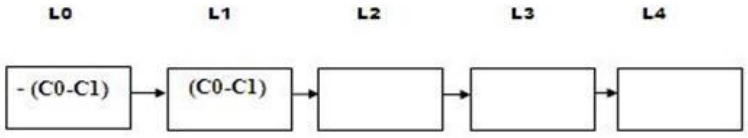

Fig .11. Linked –list of Log-nodes

8 Implementation

The proposed system can be implemented and deployed using the Java Net beans IDE. The proposed system is evaluated for three scenarios as described in the further part of the implementation.

8.1 Scenario-I

Consider a scenario in which all the services required by the user are available in a Travel Web service composition example. For instance, consider a situation in which the user require a flight service with a cheaper rate less than Rs 10000 followed by the hotel booking of the rate of Rs.20000 for the stay of five days. After the hotel booking, the user affords to take up a taxi to the conference place with no constraint of cost but with the aim of reaching on time. On finishing the conference, he needs to tour the place for two days within a range of Rs.8000. After finishing up the tour before returning to the hotel, he decides to watch a movie for less cost of Rs 300.

If all the required constraints match with services available i.e. the user gets all the services in each layer at first attempt.

In such a scenario, the user gives the request to the Service provider who interacts with the Composer by means of a broker. The first step is to generate the Service Composition Graph (SCG)

and order the services of flight service with the cheapest flight service at first and hotel services are ordered with a costlier hotel of Rs 20000 at first. The taxi services are ordered with time as constraints, the best match of time is placed on the top. Similarly the tourism and movies services are ordered with respect to user constraints. After ordering the services, first services are selected for each layer since all the services are available and the Web service composer composites the selected services and it is passed to the Web Service Broker who in turn forwards it to Service Provider and the Requester.

8.2 Scenario-II

Consider a scenario in which the user requires a Flight service with the cheaper rate of less than Rs.10000 which is followed by the hotel rate of Rs 15000, the customer then puts a constraint of taxi to reach on time with an expenditure of Rs. 250 to reach the place on time. After finishing the conference, the user wishes to tour the city for even more than Rs. 10000. After the tour completion, he likes to go for a movie for Rs. 300, assuming there is an expected delay in departure of flights due to some reasons.

The user gives the request to service provider which is forwarded to composer to composite the services by means of SCG generation followed by the ordering of values. Since the Flight will be delayed, the hotel services WS1,1 and WS1,2 associated with the first flight service WS1 wouldn't be available because of the delay. So, the composer marks the set of no-goods as the hotel services associated with first flight service WS1.The next alternative service WS2 which has the comparative high cost of Rs 11000 would reach on time and extra amount Rs.1000 is stored in the log node.

So the composer directly selects the WS2 and follows the selection of first service of hotel WS1,1. After finishing the hotel service selection, the taxi is required to be selected. But unfortunately, the taxis that are available for reaching on time are only above Rs.350. In such a situation, taking in to the consideration of user satisfaction, the service is selected and the extra Rs 100 is stored in log node.

After finishing the conference, the tourism service of just Rs 8000 is available. The tourism service is selected for Rs 8000 and Rs.100 left over in the taxi service can be compensated with this service of tourism. Then, the movie service is selected for Rs.300.

Finally the composer selects the services satisfying the user by verifying the log node. It doesn't have any pending transaction and the money gained in tourism service is compromised in the flight service and taxi service by means of the broker to the service providers. Thus, the efficiency and user satisfaction is ensured to a greater extent in the proposed system than the other algori thms which opts for user negotiation for changing the constraints.

8.3 Scenario-III

Consider the worst scenario as Scenario-II, the tourism service by the time he finishes the conference are exactly the same as user constraint i.e. .Rs 8000 only. Now the log node is verified, the compensation can't be made .Next the service selection of movie by the time he finishes the tour is Rs 300.

Also in the final service selection, the log node value is not compensated. At this situation only the user negotiation comes into picture i.e., negotiation is made with the user whether the user can afford the excess money that is to be paid to make a web service selection and composition. The proposed system web service composer chooses the same composition as selected once the constraints are altered by the user using skip-list based backtracking based on the user's option for the selection of particular service layer for compromisation. Thus, it increases the user satisfaction and efficiency through negotiation of web services. If the user does not agreed for compromise through negotiation, it leads to dissatisfaction of the user and unsuccessful job completion.

8.4 WSDL for Travel Application

With the combination of SOAP and XML Schema to provide web services over the Internet via the usage of WSDL, a client can refer to the WSDL to determine the functions that are available on the

server. Client makes use of SOAP messages to interact with the server by calling the required functions.

```
    <s:element maxOccurs="1" minOccurs="1" name="ident" type="s:string"/>
    <s:element maxOccurs="unbounded" minOccurs="1" name="codeshares" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="tailnumber" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="meal_service" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="gate_orig" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="gate_dest" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="terminal_orig" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="terminal_dest" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="bag_claim" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="seats_cabin_first" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="seats_cabin_business" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="seats_cabin_coach" type="s:int"/>
   </s:sequence>
  </s:complexType>
  <s:element name="AirlineFlightSchedulesRequest" type="FlightXML2:AirlineFlightSchedulesRequest"/>
 ▼<s:complexType name="AirlineFlightSchedulesRequest">
  ▼<s:sequence>
    <s:element maxOccurs="1" minOccurs="1" name="startDate" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="endDate" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="origin" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="destination" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="airline" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="flightno" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="howMany" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="offset" type="s:int"/>
   </s:sequence>
  </s:complexType>
  <s:element name="AirlineFlightSchedulesResults" type="FlightXML2:AirlineFlightSchedulesResults"/>
 ▼<s:complexType name="AirlineFlightSchedulesResults">
  ▼<s:sequence>
    <s:element maxOccurs="1" minOccurs="1" name="AirlineFlightSchedulesResult" type="FlightXML2:ArrayOfAirlineFlightScheduleStruct"/>
   </s:sequence>
  </s:complexType>
  <s:element name="AirlineFlightScheduleStruct" type="FlightXML2:AirlineFlightScheduleStruct"/>
 ▼<s:complexType name="AirlineFlightScheduleStruct">
  ▼<s:sequence>
    <s:element maxOccurs="1" minOccurs="1" name="ident" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="actual_ident" type="s:string"/>
    <s:element maxOccurs="1" minOccurs="1" name="departuretime" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="arrivaltime" type="s:int"/>
    <s:element maxOccurs="1" minOccurs="1" name="origin" type="s:string"/>
```

Fig 12 Screen shot of WSDL for the Sample travel Application

The Fig 12 above shows the snapshot of the WSDL file for the sample travel application which was created to test the proposed algorithm.

8.5 Performance Measures

The graph in Fig 13 depicts the analysis of percentage of job completion using the proposed system and the plot based on the analysis of the values in TABLE I.

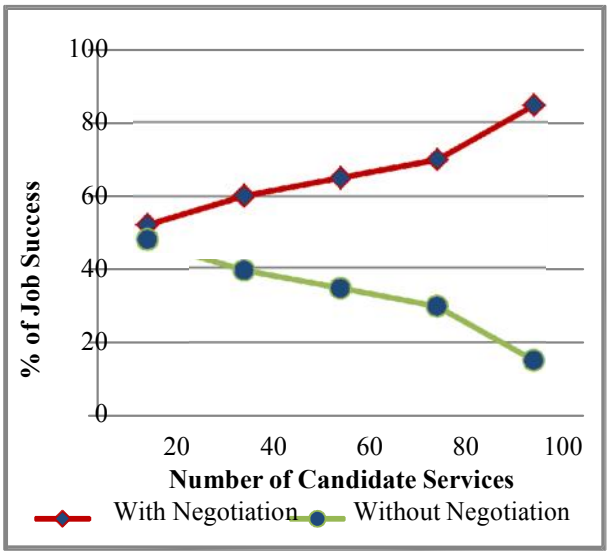

Fig 13. Percentage of Job Completion with and without Negotiation

TABLE I. Percentage of Job success rate

No. of Candidate Services	% of Job Completion	
	With Negotiation	Without Negotiation
20	52	48
40	60	40
60	65	35
80	70	30
100	85	15

The User Satisfaction is analyzed for various back-tracking techniques as in TABLE II and plotted in Fig 14.

The experimental results (Fig 13 and Fig 14) show that the objective of this paper (User Satisfaction and Better Job Success Rate) is achieved by compromising the Execution time. The resultant graph for the proposed system overhead with the execution time is plotted and shown in Fig 14.

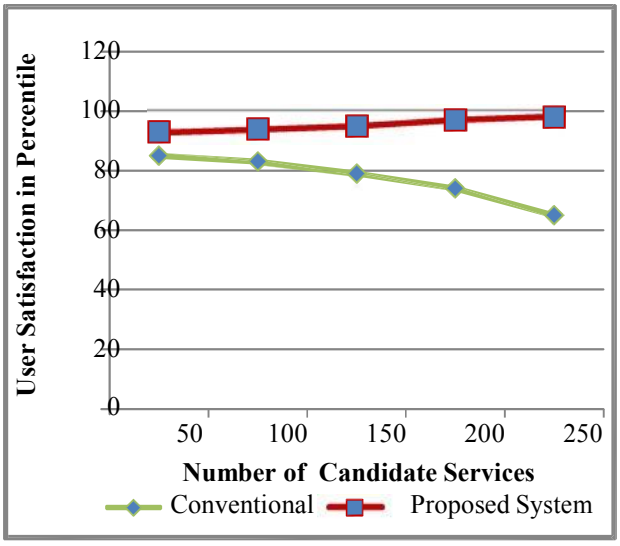

Fig 14 User Satisfaction of Various Backtracking Techniques

TABLE II. Comparison between Various Back Tracking Techniques.

Backtracking Techniques	**User Satisfaction(in Percentage)**				
Number Of Services	**50**	**100**	**150**	**200**	**250**
Conventional	85	83	79	74	65
Non Chronological (Proposed System)	93	94	95	97	98

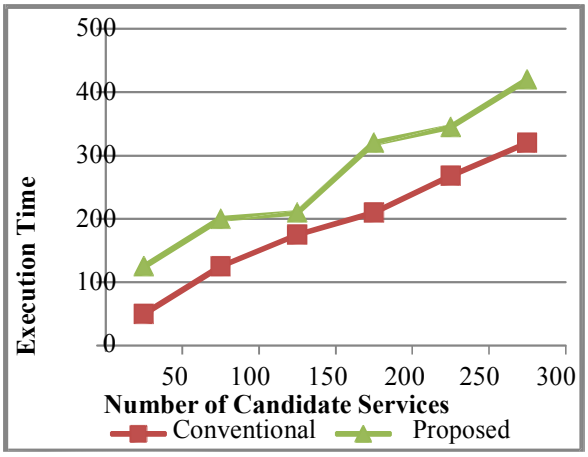

Fig 15 Proposed System Overhead with Execution Time

9 Conclusion

Thus, the proposed system is capable of adapting to the dynamic nature of web services and provides an optimal solution of web service selection and composition. It also enhances the feature of optimal solution through the use of skip-list based Intelligent Backtracking algorithm. The system supports dynamic request and provides the best service for the customer. The proposed system does not encounter the problem of thrashing and redundancy because of dependency-directed base backtracking algorithms. Thus with the various features that have been introduced in our paper through careful research of the present systems make it both viable in real-time and conducive to the development of the industry.

It is a unique approach that fuses the advantages of graph generation and intelligent backtracking techniques. Presently we have considered only five layers of services with cost as the only QoS constraint. Our future work is in the area of provision of more QoS factors to our system and includes more layers. It is important in services of our system that the information is not altered by malicious users. Thus, in any web related application, it is important to ensure that neither that the services misuse the resources of hosts nor the hosts alter the integrity of services. Improved Cost- effectiveness and satisfying the atmost user requirements will be considered in our further research.

References

[1] H.Elfawal Mansour and T.Dillon, "Dependability and RollBack Recovery for Composite Web Services", IEEE Transactions On Services Computing, 2011, Vol 4 Issue 4, pp 328-339.

[2]. Ourania Hatzi, Dmitris Vrakas, Mara Nickolaidou and Nick Bassiliades, "An Integrated Approach to Automated Semantic Web Service Composition through Planning", IEEE Transactions On Services Computing, 2012, Vol 5 Issue 3, pp 319-332.

[3] M.Suresh Kumar and P.Varalakshmi,"Dynamic Web Service Composition based on Network Modeling with Statistical Analysis and Backtracking", 7th International Conference on Computer Science and Education, 2012, pp 1184-1189.

[4] Palanikkumar D, Gowsalya Elangovan ,Rithu B and Anbuselvan P," An Intelligent Water Drops Algorithm Based Service Selection And Composition In Service Oriented Architecture". Journal of Theoretical and Applied Information Technology, 2012, Vol 39 Issue 1, pp 45-51.

[5] Andrew B.Baker,"Intelligent Backtracking on constraint satisfaction problems:Experimental and Theoretical results", Ph.D Dissertation, Graduate School of University of Oregan, 1995.

[6] Andrew Slater," Relevant Backtracking: Improved Intelligent Backtracking Using Relevance", Canberra Research Laboratory, National ICT Australia, Research School of Information Science and Engineering, Australian National University, Acton, 0200

[7] C. Rajeswary," A survey on Efficient Evolutionary algorithms for Web Service Selection". International Journal of Management, IT and Engineering, Vol 2 Issue 9, pp 177-191.

[8] Hui Sun, Jia Zhao ,"Application of Particle Sharing Based Particle Swarm Frog Leaping Hybrid Optimization Algorithm in Wireless Sensor Network Coverage Optimization" Journal of Information & Computational Science,2011, Vol 8 Issue 14, pp 3181–3188.

[9] QuanwangWu, Qingsheng Zhu,"Transactional and QoS-aware dynamic service composition based on ant colony optimization" Future Generation Computer Systems, 2013, Vol 29, Issue 5, pp 1112–1119.

[10] Eduardo Blanco, Yudith Cardinale, Marfa-Esther Vidal," A Non-Chronological Backtracking Unfolding Algorithm for Transactional Web Service Composition" Procedia Computer Science, 2012,Vol 10, pp 888–893.

[11] Sunil R Dhore, Prof. Dr M U Kharat ," QoS Based Web Services Composition using Ant Colony Optimization: Mobile Agent Approach" International Journal of Advanced Research in Computer and Communication Engineering,2012 Vol. 1, Issue 7, pp 519-527.

[12] San-Yih Hwang , Haojun Wang , Jian Tang , Jaideep Srivastava ,"A probabilistic approach to modeling and estimating the QoS of web-services-based workflows" Journal of Information Sciences, an International Journal, 2007, Vol 177 Issue 23, pp 5484-5503.

[13] Tao Yu, Yue Zhang, And Kwei-Jay Lin,"Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints" Journal of ACM Transactions on the Web, 2007, Vol 1 Issue 1, Article No. 6.

[14] Huiyuan Zheng,"QoS Analysis for Web Service Compositions." IEEE International Conference on Services Computing, 2009, pp 235-242.

[15] Cao et al , "QoS Driven Multicast Tree Generation using Genetic Algorithm" Advanced Parallel Processing Technologies, 5th International Workshop, APPT 2003, Xiamen, China, 2003, pp 404 – 413.

[16] Lifeng Ai ,"QoS-Aware Web Service Composition using Genetic algorithms" Ph.D Thesis, Queensland University of Technology, 2011.

[17] Mohammad K. Sepehrifar, Kamran Zamanifar, Mohammad B. Sepehrifar, "An Algorithm to Select the Optimal Composition Of the Services", Journal Of Theoritical and Applied Information Technology, 2005, Vol 8 Issue 2, pp 154-161.

[18] Ran Tang and Ying Zou "An Approach for Mining Web Service Composition Patterns from Execution Logs.