A Combinatorial Procurement Auction for QoS-Aware Web Services Composition

Megha Mohabey∗, Y. Narahari∗, Sudeep Mallick †, P. Suresh † and S.V. Subrahmanya†
∗Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, India,
Email: {megha, hari}@csa.iisc.ernet.in
†Infosys Technologies, Bangalore, India,
Email: {sudeepm, suresh_p01, subrahmanyav}@infosys.com

Abstract—Business processes and application functionality are becoming available as internal web services inside enterprise boundaries as well as becoming available as commercial web services from enterprise solution vendors and web services marketplaces. Typically there are multiple web service providers offering services capable of fulfilling a particular functionality, although with different Quality of Service (QoS). Dynamic creation of business processes requires composing an appropriate set of web services that best suit the current need. This paper presents a novel combinatorial auction approach to QoS aware dynamic web services composition. Such an approach would enable not only stand-alone web services but also composite web services to be a part of a business process. The combinatorial auction leads to an integer programming formulation for the web services composition problem. An important feature of the model is the incorporation of service level agreements. We describe a software tool QWESC for QoS-aware web services composition based on the proposed approach.

I. INTRODUCTION

Web services or more generally “services” are defined as autonomous, loosely coupled units of functional logic which could be automatically exposed, discovered, and invoked by other applications and systems using implementation platform agnostic XML based communication protocol and invocation APIs. Web services have rapidly captured the attention of the industry and provide one of the most favored implementation vehicles of the service-oriented architecture (SOA) approach, which holds out a tremendous potential for the IT industry with applications ranging from the financial sector to retail supply chains. While the concept of encapsulating business logic into “service” units is rapidly proliferating in enterprises for their internal use, B2B scenarios; commercial web services are also increasingly becoming available from either web services marketplaces such as www.strikeiron.com or from service providers such as www.amazon.com, www.google.com. Typically there are multiple web service providers (WSP) offering services capable of fulfilling a particular functionality. Web service providers can provide stand-alone web services or composite web services or both. A composite web service is any service that can achieve the functionality of two or more individual services. In this paper we address the problem faced by a web service requester (WSR) who seeks to find the best mix of web services that would achieve well defined cost and quality criteria. We refer to this composite service, which is desired by a WSR, as the end-to-end composite service.

A. Review of Related Work

The papers by Zeng et al [1] and Zeng et al [2] present a middleware platform which addresses the issue of selecting web services for the purpose of their composition in a way that maximizes user satisfaction expressed as utility functions over Quality of Service (QoS) attributes, while satisfying the constraints set by the user and by the structure of the composite service. They suggest a multidimensional QoS model which captures nonfunctional properties that are inherent to web services in general, e.g., availability and reputation. Their model defines a number of QoS properties and methods for attaching values for these properties in the context of both stand-alone and composite web services. They describe and compare two selection approaches: one based on local (task-level) selection of services and the other based on global allocation of tasks to services using integer programming. They suggest a dynamic composition approach, in which runtime changes in the QoS of the component services are taken into account.

Gao et al [3] suggest a QoS model that includes capacity and load, which may be useful for capacity planning and QoS negotiation at service provider side. They also illustrate the process of defining an objective function when there are composite constructs. They investigate implicit enumeration and found it impractical.

Canfora et al [4] propose a genetic algorithm (GA) based approach for QoS aware service composition. GAs permit to deal with QoS attributes having non-linear aggregation functions. Also, GAs are able to scale up when the number of WSPs for a particular task increases as compared to linear programming. Moreover, to deal with constraints, it is possible to adopt a fitness function with both static or dynamic penalty. However, they find that integer programming is often faster than genetic algorithms.

Gao et al [5] and Gao et al [6] present a 3-layer organization model and an approach for selecting globally optimal web services based on weighted multistage graph considering interface matching between web services. The approach transforms the problem of selecting the optimal execution plan for composite web services into one of selecting the
longest path in a weighted multistage graph. The paper by Gao et al [5] describes a genetic or immune algorithm to select the optimal execution plan while the paper by Gao et al [6] adopts dynamic programming to select optimum web services for composite services dynamically.

B. Motivation for the Current Work

The approaches discussed in the previous section consider the problem of composition of stand-alone web services to form an end-to-end composite service. However, WSPs can provide composite services too. An approach that considers both stand-alone and composite services for forming an end-to-end composite service has the following advantages:

- It helps in capturing the fact that a WSP may be willing to provide a composite service at a price lower than the total price of stand-alone services that form the composite service.
- The quality attributes of a composite service offered by a WSP may be not the same as achieved by composing its constituent stand-alone services. For example, the execution delay of composite service AB can be less than the sum of the individual execution delays of stand-alone services A and B, as the AB functionality can be achieved at the same server which will reduce the delay that may be involved in exchanging long XML messages [7] between different servers.
- Different WSPs have different number of stages/steps for achieving the same task and they can expose the services at different points. For example, Fig. 1 shows the services exposed by two different providers. Stage 2 of WSP 2 achieves the functionality of stages 2 and 3 of WSP 1. Thus WSP 2 cannot provide service F_2 alone, but it can provide composite service F_2F_3. Thus having composite web services allows expressing such scenarios.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{services.png}
\caption{Services provided by different WSPs}
\end{figure}

This paper presents a novel combinatorial auction approach to QoS-aware dynamic web services composition. Our approach considers both stand-alone and composite services for composition of an end-to-end composite service. We provide an integer programming (IP) formulation to solve the problem. An important feature of the model is the incorporation of service level agreements. In addition to this, we describe a software tool, QWESC (QoS-Aware WEb Services Composer), for QoS-aware web services composition based on the proposed approach.

The remainder of the paper is organized as follows. Section II describes the problem and provides a sketch of our approach to solve the same. Section III provides the architecture of software tool QWESC. Section IV describes the combinatorial auction approach and an IP formulation to solve it. Section V describes numerical experiments conducted to evaluate the performance of the proposed approach. Finally in Section VI we conclude the paper by giving a few possible directions of future investigation.

II. PROBLEM DEFINITION

A composite service is specified as a collection of generic service tasks described in terms of service ontologies and combined according to a set of control-flow and data-flow dependencies. These dependencies can be represented by means of statecharts [2]. There are multiple WSPs capable of executing one or more tasks. The best mix of web services that achieve well defined cost and quality criteria have to be found for the execution of end-to-end composite service, desired by WSR, thus an execution plan is constructed. Typically, a statechart has several execution paths. In this paper we discuss how to get the optimal execution plan for a single execution path as shown in Fig. 2. Typically, there is a certain cost associated with invoking the web service once, which is the price per hit of the web service.

A. A Specification of Composite Services

We assume that a WSP can provide a bundle of services i.e. if a WSP can provide web service A and web service B, then it can provide composite service AB too, perhaps at a different price. We use the OR bidding language\(^1\) for representing details of web service offerings from the WSPs, since it is most suitable for the model. In this paper, we use the word “bid” to mean such announcements by WSPs. Thus a typical bid structure is as follows:

\(< \text{index}, (S_1 = \{t_1\}, q_1, q_2, \ldots, q_n, c(S_1)), (S_2 = \{t_2\}, q_1, q_2, \ldots, q_n, c(S_2)), \ldots, (S_n = \{t_n\}, q_1, q_2, \ldots, q_n, c(S_n)), (S_{12} = \{t_1, t_2\}, q_1, q_2, \ldots, q_n, c(S_{12})), \ldots >\>

where, index is index of the WSP; \(t_i\) is \(i^{th}\) task in the statechart; \(q_i\) is \(i^{th}\) quality attribute of web service and \(c(S)\) is price per hit for given subset \(S\) of tasks when demand is \(D\) hits. For example if a WSP can provide services A, B, C, AB, and BC, and there are two quality attributes, then the bid will look as follows:

\(< (\{A\}, q_1, q_2, c(A)), (\{B\}, q_1, q_2, c(B)), (\{C\}, q_1, q_2, c(C)), (\{A, B\}, q_1, q_2, c(AB)), (\{B, C\}, q_1, q_2, c(BC)) >\>

\(^1\)Each bidder can submit any arbitrary number of atomic bids. It is assumed that she is willing to obtain any number of these disjoint atomic bids for the sum of their respective prices [8].
B. Incorporation of Service Level Agreement in the Model

A service level is used to define the expected performance behavior of a deployed web service, where the performance metrics are, for example, average response time, supported throughput, service availability, etc [9]. Different WSPs may offer the same web service with different QoS. Also, a WSP can offer the same web service at different service levels to cater different WSRs, for example, at bronze, silver, and gold levels with increasingly better response times. Hence, to receive assurances on the service level, a WSR creates a priori a service level agreement (SLA) associated with this web service with the WSP. In our model, we capture different service levels from the same WSP by representing them as different combinatorial bids. However, while solving the optimization problem, we treat each bid of a particular WSP X as if arising from different providers X_{silver}, X_{bronze}, X_{gold}.

C. Feasible Bids

Consider the statechart in Fig. 2 consisting of three states/tasks A, B, and C. Then the possible combinations for which a particular service provider can bid are A, B, C, AB, BC, and ABC. Notice that a bid for AC may not make sense since A and C are not consecutive, which means that they cannot be offered as a composite service. Any bids not following this requirement could be dropped in a preprocessing step.

D. Allocation

The problem faced by the WSR is to find the set of web services that maximize quality subject to budget constraint i.e. it needs to come up with an allocation of tasks to web services. We come up with a combinatorial auction approach for allocation. We provide a 0-1 integer programming formulation, solving which gives allocation of tasks to web services.

In the next section, we describe the architecture of a tool, QWSC, that can be used for web service composition by a WSR for dynamic composition of web services.

III. ARCHITECTURE OF QWSC

Fig. 3 shows the architectural diagram of the tool and its interfacing with QoS registry, discovery engine, and WSPs. We build on earlier work on QoS computation and policing [10] and web service composition [11]. WSPs can create service advertisements and publish them in an enhanced UDDI registry [11]. QoS registry collects and keeps QoS criteria via active monitoring (at WSR’s end) and active users’ feedback [10]. The QoS monitoring at the WSR’s end monitors the execution of web services and sends feedback to the QoS registry. The discovery engine, when given a service template, returns the set of service advertisements which match the template [11].

The different components of tool are described as follows:

- **Abstract Process Designer**: The WSR describes the abstract services that form the end-to-end composite service by means of service templates. An abstract service is a placeholder for a set of services matching the abstract service’s template [11]. These service templates are sent to the discovery engine.

- **Preprocessor**: The discovery engine returns the set of service advertisements which match the template. The preprocessor converts them to bid description as described in Section II-A. The preprocessor also filters out infeasible bids as explained in Section II-C.

- **Optimization Problem Formulator**: This component formulates the optimization problem, as described in Section IV-B and IV-C.

- **IP Solver** This component solves the IP formulation of the web service composition problem.

- **Web Service Binder and Execution Engine** This component deals with binding the abstract process to the optimal set of services (that form an optimal solution) to generate an executable process (end-to-end composite service) [11]. The web services of the respective WSPs are invoked accordingly during the execution.

This paper proposes a novel combinatorial auction approach in formulating the optimization problem involved. This is discussed in next section.

IV. A COMBINATORIAL AUCTION FOR WEB SERVICES COMPOSITION

A. Combinatorial Auction

An auction is a mechanism to allocate a set of goods to a set of bidders on the basis of their bids. In combinatorial auctions (CAs), bidders can bid on combinations of items. In a combinatorial procurement auction, a buyer (in this case, WSR) wishes to buy a set of distinct items A (in this case,
web services for distinct tasks). There is a set of selling agents (in this case, WSPs) who are interested in selling the entire set or some subsets of $A$. The buyer wants to minimize the procurement cost i.e. procure the goods from the bidders with minimum total cost (in this case, buyer wants to maximize quality subject to budget constraints). Due to similarities in our setting and combinatorial procurement auction setting, we come up with a combinatorial auction approach for allocation. For a detailed discussion on combinatorial auctions, we refer the reader to Narahari and Dayama [12], and Chandrashekar et al [13], for a discussion on auction based mechanisms for electronic procurement.

B. Objective Function

For the selection of an optimal execution plan for a given execution path, we apply a Multiple Criteria Decision Making (MCDM) technique, namely, Simple Additive Weighting (SAW) technique as in [2]. The two phases of applying SAW are:

1) Scaling Phase: In this section, we provide a method of scaling in the presence of both stand-alone and composite web services capable of fulfilling a given task. We consider the quality attributes reputation, success rate\(^2\), availability, and execution duration, however other quality attributes can also be incorporated in similar fashion. Some of the criteria could be negative, i.e., the higher the value, the lower the quality. This includes criteria such as execution time. Other criteria are positive, i.e., the higher the value, the higher the quality. For negative criteria, values are scaled according to (1). For positive criteria, values are scaled according to (2).

$$Q_{ij}(S) = \begin{cases} \frac{q_{ij}^{max} \times t(S) - q_{ij}(S)}{q_{ij}^{max} - q_{ij}^{min}} & \text{if } q_{ij}^{max} - q_{ij}^{min} \neq 0 \\ 1 & \text{if } q_{ij}^{max} - q_{ij}^{min} = 0 \end{cases} \quad (1)$$

$$Q_{ij}(S) = \begin{cases} \frac{q_{ij}(S) - q_{ij}^{min} \times t(S)}{q_{ij}^{max} - q_{ij}^{min}} & \text{if } q_{ij}^{max} - q_{ij}^{min} \neq 0 \\ 1 & \text{if } q_{ij}^{max} - q_{ij}^{min} = 0 \end{cases} \quad (2)$$

Note that since there are bids for composite services, while calculating maximum and minimum we have adopted the following procedure:

1) If the service is a stand-alone service, take the value as in the bid.
2) If the service is composite, take $r_{ij}(S)/t(S)$. For example, if the execution duration of a composite service $AB$ is 34, then it is taken as $34/2 = 17$ for calculating maximum and minimum.

This need not be done for the case of quality attribute reputation since we have assumed that it is specified in scale $[0,5]$. Moreover, we have taken the natural logarithm $\ln$ of success rate and availability before applying scaling phase since the success rate of a composite service is product of success rate of stand-alone services that form them. Thus, we can use summation of these $\ln$ values to achieve the same.

2) Weighting Phase: To compute the overall quality score for each execution plan, weighted sum of different quality criterion is taken. WSR can give weights to different QoS criterion according to its preferences.
TABLE II
VARIOUS CASES CONSIDERED

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Number of agents</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Execution duration weight</td>
<td>0.3</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td>Reputation weight</td>
<td>0.1</td>
<td>0.05</td>
<td>0.25</td>
</tr>
<tr>
<td>Success rate weight</td>
<td>0.3</td>
<td>0.10</td>
<td>0.25</td>
</tr>
<tr>
<td>Availability weight</td>
<td>0.3</td>
<td>0.10</td>
<td>0.25</td>
</tr>
<tr>
<td>Maximum budget</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Maximum of units</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus the of selection of an optimal execution plan for a given execution path can be mapped into ILP (Integer Linear Program) as follows:

The objective function is:

$$Z^* = \max \sum_{S \subseteq A} \sum_{j=1}^l \sum_{i \in N} w_j \times Q_{ij}(S) \times y_i(S)$$  \hspace{1cm} (3)

C. Constraints

This section describes the constraints of the IP problem.

a) Allocation Constraints:

$$\sum_{S \supseteq k \in E} y_i(S) = 1, \ \forall k \in A$$  \hspace{1cm} (4)

$$y_i(S) \in \{0, 1\}, \ \forall S \subseteq A, \forall i \in N$$  \hspace{1cm} (5)

Constraint (4) ensures that exactly one WSP is selected for each task. Here, $Q_{ij}(S)$ is the value of the $j^{th}$ QoS criterion in the bid of $i^{th}$ WSP for subset $S$ of tasks after scaling. Scaling is discussed in more detail in Section IV-B.1. We have used Simple Additive Weighting [2] technique to select an optimal web service.

b) Budget Constraint: We assume that the total demand is $D$ units from each state/task in statechart and all WSPs for each task are capable of providing the same for that task.

$$\sum_{S \subseteq A} \sum_{i \in N} D y_i(S) c_i(S) \leq B$$  \hspace{1cm} (6)

Constraint (6) ensures that the total cost does not exceed the budget.

WSR may specify additional global constraints according to the requirements from the end-to-end composite service. For example following constraint ensures that the execution duration of the composite service does not exceed $T$.

$$\sum_{i \in N} \sum_{S \subseteq A} y_i(S) T_i(S) \leq T$$  \hspace{1cm} (7)

Similarly, constraints on reputation, success rate and availability can be specified.

Variables in ILP: The number of variables in the ILP is $O(n2^m)$.

V. EXPERIMENTS

In order to understand the effectiveness of the proposed approach, and compare its performance with non-combinatorial version, we evaluated the solutions given by combinatorial auction and non-combinatorial version in different settings.

A. Setup

For this investigation, we chose to focus on the simple statechart in Fig. 2. There are three service providers $P_1$, $P_2$, and $P_3$. Table III shows the bid submitted by providers $P_1$, $P_2$, and $P_3$. $P_1$ submits bid for subsets $A$, $B$, and $AB$, $P_2$ submits bid for subsets $B$, $C$, and $BC$ and so on. $P_2^{gold}$ represents the bids submitted by WSP $P_2$ with a high level of service guarantees. We consider four quality attributes namely, execution duration, reputation, success rate, and availability.

B. Implementation

We implemented the code in Java and used ILOG CPLEX 9 (a commercial MIP software package) to solve the combinatorial auction allocation problem.

C. Results

We consider three different scenarios. In Table II, case 1, case 2, and case 3 represent the various scenarios considered. The $\times$ in the last six columns of Table III indicate the bids that cannot participate (or are not submitted) in those cases. For example, a non-combinatorial auction cannot process bundled bids. The $\sqrt{}$ in the last six columns of Table III indicate the winning bids in respective cases.

The various results are shown in Table IV. We can see that, the optimal value of objective function for combinatorial auction approach is always greater than or equal to that for non-combinatorial one. Thus, the combinatorial auction approach gives better quality at less cost. Note that the reputation of end-to-end composite service in case 2 for non-combinatorial is greater than combinatorial auction approach which is a result of WSR specifying very less weightage to reputation as compared to other quality attributes. Clearly, a combinatorial auction based approach performs better than non-combinatorial auction in the first two cases. Moreover, in the last case, as all the web services that form the optimal solution are stand-alone, combinatorial auction correctly identifies the same. Note that the combinatorial auction will always perform better than non-combinatorial case since its search space subsumes that of the non-combinatorial one.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a combinatorial auction approach to QoS aware dynamic web services composition. Our approach considers both stand-alone and composite services for composition of end-to-end composite service, as compared to other approaches [1], [2], [3], [4], [5], [6] that consider only stand-alone services for composition of end-to-end composite service. We believe that our approach models the real world scenario in a better way. Moreover, we show that the combinatorial auction approach will always perform better than any non-combinatorial approach.

Our current work limits the discussion to a linear execution path. Moreover, the services offered by different WSPs may not be interoperable, if they have incompatible input output interfaces. Service interface is used to transfer control and data messages between web services. Introducing forks and
TABLE III
BIDS SUBMITTED BY WSPs AND THE WINNING BIDS IN THE RESPECTIVE CASES (C - WITH COMBINATORIAL BIDS, NC - WITHOUT COMBINATORIAL BIDS)

<table>
<thead>
<tr>
<th>No</th>
<th>WSP Composite service</th>
<th>Execution duration</th>
<th>Reputation [0,5]</th>
<th>Success rate [0,1]</th>
<th>Availability [0, 1]</th>
<th>Cost per unit</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P₁ A</td>
<td>10</td>
<td>5</td>
<td>0.95</td>
<td>0.98</td>
<td>15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>P₁ B</td>
<td>22</td>
<td>5</td>
<td>0.95</td>
<td>0.98</td>
<td>10</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>P₁ AB</td>
<td>30</td>
<td>5</td>
<td>0.9025</td>
<td>0.9604</td>
<td>20</td>
<td>✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>P₂ B</td>
<td>20</td>
<td>4</td>
<td>0.95</td>
<td>0.98</td>
<td>12</td>
<td>✓ ✓ ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>P₂ C</td>
<td>15</td>
<td>5</td>
<td>0.95</td>
<td>0.98</td>
<td>19</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>P₂ BC</td>
<td>33</td>
<td>4</td>
<td>0.9025</td>
<td>0.9604</td>
<td>30</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>P₃ A</td>
<td>12</td>
<td>4</td>
<td>0.95</td>
<td>0.98</td>
<td>13</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>8</td>
<td>P₃ B</td>
<td>25</td>
<td>4</td>
<td>0.95</td>
<td>0.98</td>
<td>11</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>P₃ C</td>
<td>12</td>
<td>4</td>
<td>0.95</td>
<td>0.98</td>
<td>19</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>10</td>
<td>P₃ AB</td>
<td>35</td>
<td>4</td>
<td>0.9025</td>
<td>0.9604</td>
<td>23</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>11</td>
<td>P₃ BC</td>
<td>34</td>
<td>4</td>
<td>0.9025</td>
<td>0.9604</td>
<td>29</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>12</td>
<td>P₃ ABC</td>
<td>40</td>
<td>4</td>
<td>0.857375</td>
<td>0.941192</td>
<td>40</td>
<td>✓ ✓ ✓</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

TABLE IV
THE VALUE OF VARIOUS ATTRIBUTES WHEN THE BIDS ARE SELECTED AS IN TABLE III TO FORM COMPOSITE SERVICE FOR VARIOUS CASES (C - WITH COMBINATORIAL BIDS, NC - WITHOUT COMBINATORIAL BIDS)

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>39</td>
<td>44</td>
<td>49</td>
</tr>
<tr>
<td>Execution duration</td>
<td>45</td>
<td>57</td>
<td>45</td>
</tr>
<tr>
<td>Reputation</td>
<td>5</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>Success rate</td>
<td>0.857375</td>
<td>0.857375</td>
<td>0.88445</td>
</tr>
<tr>
<td>Availability</td>
<td>0.941192</td>
<td>0.941192</td>
<td>0.950796</td>
</tr>
<tr>
<td>Optimal value</td>
<td>0.90</td>
<td>0.86</td>
<td>1.229379</td>
</tr>
</tbody>
</table>

VII. ACKNOWLEDGMENTS
We wish to thank the support provided by the Infosys Technologies, Bangalore in undertaking this research.

REFERENCES