International Conference on Electrical, Information and Communication Technology

# Fuzzy Optimised Power Generation from Moving Vehicles

Roohi Banu I, Sindhu T, Vinothini S P, Yamini J P, Bharath Kumar S
Department of Electrical Engineering, M.A.M College of Engineering,, Tamil Nadu, India

**Article Info**
Article history:
Received 2 February 2015
Received in revised form
20 February 2015
Accepted 28 February 2015
Available online 6 March 2015

**Keywords**

**Abstract**
In our Paper regenerative power system for electric motorcycles and cars that performs regenerative energy recovery from the axle of the vehicle based on fuzzy logic control for a boost converter is used to boost (maintain) the voltage level. Autonomous vehicles have potential applications in many fields, such as replacing humans in hazardous environments, conducting military missions, and performing routine tasks for industry. A constant regenerative current control scheme is proposed, thereby providing improved performance and high energy recovery efficiency at minimum cost. Drivers typically respond quickly to sudden changes in their environment. While other control techniques may be used to control a vehicle, fuzzy logic has certain advantages in this area; one of them is its ability to incorporate human knowledge and experience, via language, into relationships among the given quantities.

## 1. Introduction

Semantic web services frameworks provide comprehensive tools to describe services and their interactions. Semantic approaches add significantly to the resource requirements. As such, cloud computing is candidate to bridge the gap between resource-constrained environment and resource-intensive Web service discovery. It opens up new opportunities for mobile devices to efficiently perform service discovery, while substantially reducing their resource consumption. Cloud computing not only bootstraps the performance of service discovery in mobile environments, but also removes development constraints by expanding the horizon with more options to apply sophisticated techniques that might potentially result in better service discovery.

Researchers over the past few years have focused on optimizing specific aspects of current Web service discovery approaches in isolation or overcoming individual limitations (such as intermittent connectivity) to fit the inherent constraints and dynamic context of mobile domains.

However, the lack of a comprehensive understanding of both user context, and the various constraints of mobile environments, renders most of these approaches incapable of efficient and reliable discovery in mobile scenarios.

Context-aware computing refers to a general class of mobile systems that can sense their physical environment, and adapt their behavior accordingly. Context-aware systems are a component of a ubiquitous computing or pervasive computing environment. Three important aspects of context are: where you are; who you are with; and what resources are nearby. Although location is a primary capability, location-aware does not necessarily capture things of interest that are mobile or changing. Context-aware in contrast is used more generally to include nearby people, devices, lighting, noise level, network availability, and even the social situation, e.g., whether you are with your family or a friend from Context can refer to real-world characteristics, such as temperature, time or location. This information can be updated by the user (manually) or from communication with other devices and applications or sensors on the mobile device. An example of a context-aware service could be a real-time traffic update or even a live video feed of a planned route for a motor vehicle user.

The paper should start with an abstract, followed by the main body of the paper as follows. Section 2 describes the proposed framework and relevant research efforts that may be potentially applied for each component. Section 3 describes the framework functionality is validated and evaluation. Finally, section 4 draws the concluding remarks of the paper.

## 2. Daas: Mobile Cloud Based Service Discovery Framework

Based on the limitations of existing DAAS service only check the client environment and give the top services in mobile now plan to study the tradeoff between performing the context matching on the mobile device and offloading it to the cloud, where service matching occurs.

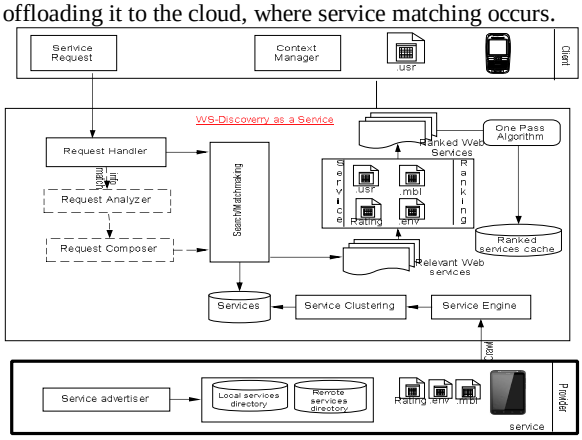


**Fig: 1-** Example of an overview of the service discovery framework Daas.

## 3. Skyline Computation Module:

A service skyline can be regarded as a set of SEPs that are not dominated by others in terms of all user interested QoWS aspects, such as response time, fee, and reputation. The service providers in the skyline represent the best tradeoffs among different user interested quality aspects.

**Corresponding Author,**
**E-mail address:** vino.eee19@gmail.com

There are several key extensions on skyline analysis aiming to make it more flexible and adapt to user's preferences.

## 1.1 Customer layer

### 1.1.1 Service Request

Multimodal mobile user interface that enables end-users to submit Web service request that express their specific objectives. The service request is submitted in a plain text that describes the user's objective. For the Mobile users usually have limited input capabilities, which are unsuitable for a formatted service request or a formal service description language (i.e. in semantic services).

The plain text fits well for the service request within mobile device constraints. Therefore, analysis of the user input is done by the Service Request Handler to extract keywords and meaningful information. These keywords are used in keyword-based matching. In case of semantic matching, they can be used to construct a formatted service request.

### 1.1.2 Context Manager

The Context Manager is responsible for collecting user environment context using mobile device embedded sensors and capabilities. It also handles the user preferences and Such context information is used to rank relevant Web services according to their best fit into such context. For Context Manager also monitors available networks and their status.

## 1.2 Cloud Layer

This layer deployed on the cloud where computational resources and power consumption are of less concern.

### 1.2.1 Request Handler

The Request Handler performs advanced processing operations on the user request. It extracts keywords from the user request using text analytics techniques. These keywords are used to perform keyword-based service matching, or format the service request to match the different possible service description languages (WSDL, OWL-S, WSMO, WSDL-S, etc.), according to the available services in the cloud service repository.

### 1.2.2 Request Analyzer

Mainly gets involved when no atomic services are found satisfying the request objective. In other words, DaaS presents two levels of service discovery.[1]The service request is matched first with atomic services that totally satisfy the user's objective. [2]If no relevant services are found, the Request Analyzer breaks down the request into primitive subtasks ST = {st1, st2. . . stn}, if possible, and satisfies each subtask separately. For Breaking down the request (in collaboration with the Service Composer) may follow approaches Start with the request inputs to get the request outputs with exact or partial matches, and Find services that have exact matches with subtasks, i.e. the matchmaking algorithm yields an EXACT or SUBSUMES match between the subtask inputs/outputs and the relevant service inputs/outputs.

### 1.2.3 Service Composer

DaaS is responsible for the orchestration process and generating composition plans for atomic Web services that satisfy the individual primitive subtasks, and together fulfill the original service request. Resolving users' requests via composite services, Propose a framework for semantic Web service composition in mobile environments. Their framework converts WSDL files into OWL-S specifications and generates a service profile for the request. Then, it performs a semantic reasoning between the advertised service profile and the request service profile.

### 1.2.4 Search/ Matchmaking

The Search/Matchmaking module matches the functionalities described in the request and the capabilities offered by Web services. For non-semantic Web services, i.e. described by WSDL files, the matching between the request and Web services is keyword-based and uses information retrieval techniques. These keywords are used to index Web services and later matched with keywords extracted from the user request. Semantically described Web services are discovered using high level matchmaking approaches.

### 1.2.5 Request Matching Algorithm

The service request component receives a set of relevant Web services S = {s1, s2,., sl} and ranks them based on the following four various types of context domains: user preferences P = {p1, p2, . . . , pi}, device profile D = {d1, d2, . . , dj}, environment context E = {e1, e2, . . . , ek}, and user ratings Rsl , where Rsl represents the normalized average rating for a service sl.

```
Void function ReqMatch(SR)
{
int s,sr,st,p,d,e;
//search for atomic service first
//trying to fulfill the whole request with single request
s=match(sr);
if{//subtasks st;
st=split(sr);
for(int st=0;st<=ST; st++)
{
do{s=s +match( st );
}}
else if
{
s=composer(s);
}
return s;
}}
```

### 1.2.6 Ranking Algorithm

The Service Ranking module employs the information in this file and that collected at the customer side to rank the relevant services. Accordingly, the discovery process may realize that the requester is able to execute limited functionalities of particular services and so rank them accordingly.

```
Void function rank(s,p,e,d,r){
int ranks;
for{
        rank s=calculate Eq.(2)
    rank S=Rank S+ Rank s
}
//sort the results
Sort(rank s);
  return rank s;
```

}

During the request communication session, the discovery mechanism collects the features of the customer's device Dc = {dc1, dc2. . . dcj} corresponding to the required device profile D. Simultaneously, the device senses various ambient conditions to better assess the environment status Ec = {ec1, ec2, . . . , eck}. For each service sl, there are features ps corresponding to user preferences P as expressed by the matrix in Eq. (1). These features can be extracted from the service description files.

$$Sp=\begin{pmatrix} Ps1,1 & Ps1,2 & \dots & Ps1,i \\ \vdots & \ddots & & \vdots \\ Psl,1 & Psl,2 & \dots & Psl,i \end{pmatrix} \quad (1)$$

The rank of each service S in then is represented by the formula

Rank Sl =

$$\sum_i^{'} wi * f(pi,psi) + \sum_j^{'} wj * f(dj,dcj) + \sum_k^{'} wk * f(ak,ack) + Rs1 \quad (2)$$

Where w represents the weight of each corresponding feature. This weight indicates the level of importance of such a feature to either the service customer or the service provider. The function f computes the relation between two objects as

$$f(a,b)=\begin{cases} \frac{b}{a+b} \; if\; a,b\; are\; numbers \\ sim(a,b)\; if a,b\; are\; strings \end{cases} \quad (3)$$

Where sim(a, b) is a featureless similarity factor computed between objects a and b using Normalized Google Distance (NGD) . sim(a, b) is calculated by Eq. (4).

Sim(a,b)=1-NGD(a,b) (4)

f (a, b) yields values ∈ [0 . . . 1]. In the case where a and b are numbers, values greater than or equal 0.5 means that b satisfies a. The closer the value to 1, the greater the satisfaction that condition achieves relative to the condition a. In the case where a and b are text (keywords), the function value close to 1 indicates that the terms a and b are semantically related, where values close to 0 indicates that they are not related. Algorithm 2 symbolically explains how ranking works.

### 1.2.7 One Pass Algorithm

During the single pass of the CSEP space, OPA enumerates the candidate CSEPs one by one and only stores the potential skyline CSEPs. It outputs the skyline after all the candidate CSEPs have been examined. OPA requires that all the SSkies are sorted according to the scores of the SEPs. OPA works as follows (shown in Algorithm 1). It starts by evaluating the first CSEP (referred to as CSEP1) that is formed by combining the top SEPs from each S-Sky. It is guaranteed that CSEP1 2 C-Sky because CSEP1 has the minimum score so that no other CSEPs can dominate it. With the minimum score, CSEP1 is expected to have a very good pruning capacity.

**for (**N= $\prod_{i=1} m$|Ski|){

// number of candidate CSEPs

CSEP i =**Aggregate**(SEP11,…..SEPm1);

C-Sky.add(CSEP1);

**for (** i∈ $[2, \quad N]$ )

CSEP i =EnumerateNext(SK 1,…..SK m);

IsDominated =false ;

**for (** j ∈ $[1, |C - Sky|]$ )

CSEP j= C-Sky.get(j);

**if** { CSEP i.score < CSEP j. score **then**

**if{** CSEP I < CSEP j then

C-Sky.remove(j);

**}**

**else**

**if** { CSEP j< CSEP I then

IsDominated=True;

break;

**}**

**}**

**if**

**{** IsDominated= = False then

C-Sky.add(CSEP i);

}

**}**

### 1.3 Web service Clustering

This component clusters Web services into functionally similar groups based on their descriptions. Therefore, the Service Matching does not need to match the user request against all the service offerings in the corpus, but rather with a particular set of services that share similar functionality. The clustering and classification of new services are performed offline to eliminate any overhead during service matching.

The service clustering renders centralized repositories performing similar to distributed approaches in P2P architectures, where each domain-specific service are maintained and matched separately. Our clustering approach not only reduces the response latency by reducing the search space, but also improves the recall and precision via data mining and text analytic techniques. In addition, it takes advantage of the continuous crawling of service engines by enabling adaptive re-clustering and self organization in order to cope with the dynamicity of Web services.

### 1.4 Provider Layer

Mobile service scenarios, service providers could be a mobile entity with limited resources. DaaS removes the burden on such resource constrained providers by shifting resource-intensive processes to the cloud, keeping only necessary components on mobile elements. Our framework the only component that providers need to run using local resources is the module responsible of announcing Web service availability.

### 1.4.1 Service Advertiser

This component announces the existence of Web services in a local service directory. Publishing Web services in a local service directory is similar to publishing services in an enterprise UDDI but on an individual scale. The direct benefit of this approach is putting service providers in full control of their offerings. This also eliminates the burden of service registration and de-registration that service providers are required to do with the UDDI approach. It also makes it easier to keep local service repositories up-to-date and self-maintained.

### 1.4.2 Local Services Directory

Service providers maintain a local directory that contains all Web services they offer. DaaS adopts a

distributed service directory approach, where each mobile device manages its own offered Web services and maintains references to services it knows about. Local services are hosted and provided by the local system.

### 1.4.3 Remote Services Directory

Remote service are active and running services host the other mobile devices. Handling remote services requires a coordination protocol to manage link updates, advertisement notifications, invalid service and link removal, duplicate reference avoidance.

## 2. Conclusion

This project implements this entities have been used User, cloud and Service Provider. We can dominate the other services based on Quality of services based on response time and the rank of the services. The one pass algorithm improves the efficient service selection in web service. Cloud has an unlimited storage spaces and store a more and more web services. User request the services through our mobile, the cloud return the services based on mobile device configuration, user ranking for the services. Implementation of mobile app through the Android, and useful for mobile based spatial search in tourist place over the world, mainly to reduce energy constraint for any spatial data searching time. More ever, we can tradeoff between performing the context matching on the mobile device and offloading it to the cloud, where service matching occurs. We also contribute how DaaS scales across distributed data centers in the cloud. Performance evaluation reveals that the overall response time is dominated by the service matching process, while context processing contributes a relatively small component, yet this varies according the number of matching service.

## References

[1] Athman Bouguettaya, Qi Yu, Efficient Service Skyline Computation for Composite Service Selection IEEE Transactions on Knowledge and Data Engineering, 25(4), 2013

[2] J. M. García, D. Ruiz, A. Ruiz-Cortés, A model of user preferences for semantic services discovery and ranking, ESWC 2010, II, 6089, 1–14

[3] J. Pathak, N. Koul, D. Caragea, V.G. Honavar, A framework for semantic web services discovery, Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management, 45–50

[4] K. Elgazzar, P. Martin, H.S. Hassanein, A framework for efficient web services provisioning in mobile environments, The 3rd International Conference on Mobile Computing, Applications, and Services, Springer's LNICST, 2011

[5] R. Singh, S. Mishra, D.S. Kushwaha, An efficient asynchronous mobile web service framework, SIGSOFT Softw. Eng. Notes 34, 2009, 1–7