2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

Progressive Search Algorithm for Service Discovery in an IoT Ecosystem

Santosh Pattar*, Dwaraka S Kulkarni*, Darshil Vala*, Rajkumar Buyya†, Venugopal K R‡, S S Iyengar§ and L M Patnaik¶
* IoT Lab, University Visvesvaraya College of Engineering, Bengaluru, India.
† University of Melbourne, Melbourne, Australia.
‡ Bangalore University, Bengaluru, India.
§ Florida International University, Miami, USA.
¶ National Institute of Advanced Studies, Bengaluru, India.
e-mail: santoshpattar01@gmail.com

***Abstract*—With the advent of the Internet of Things (IoT) a plethora of applications are being offered across several domains of the society, industry, environment, *etc.* to ease monitoring, configuration, and other such tasks. Smart services are the fundamental components of such an application that are actuated by an IoT resource. An emerging challenge in an IoT ecosystem is the seamless delivery of the smart services to the users based on their preferences and characteristic behaviors. In this paper, we propose a Progressive Search Algorithm (ProSA) that maps the user's requirements to the attributes of the IoT resources and smart services and thereby provide personalized search results. We have categorized the user requirements into essential and optional requirements that are further mapped to the intrinsic and extrinsic properties of the smart services. Using these mapping two search schemes *viz.* Primitive Search Strategy (PSS) and Elaborate Search Strategy (ESS) are proposed that makes use of semantic and proximity measures to fine-tune the search results. We conducted empirical studies based on a Smart Airport ecosystem and compared them with the existing approach in the literature to establish the practicability of the proposed search algorithm.**

***Index Terms*—Context aware, progressive search, semantic matching, service discovery, smart airport**

I. Introduction

The Internet of Things (IoT) is an ecosystem of components *viz.*, sensors, actuators, communication devices and processors that are tightly coupled with each other to orchestrate a coordinated solution for a given task. Through this paradigm of networked objects, a wide variety of applications are being developed across many sectors to provide monitoring, surveillance, management, security, and other such functionalities [1]. In an IoT environment, a physical object is embedded with the above-mentioned components to offer certain services that are utilized to create such applications. With the penetration of the IoT, every connected physical object is capable of interacting with the other objects to share and consume the services to achieve a common goal. Given this nature of the network, there will be billions of objects that operate in a complex environment. Also, a single object or a service will not suffix the requirements to solve the given complex task [2].

To facilitate easy coordination among the objects, there is a need for a mechanism that enables them to look for each other. Search techniques are thus paramount in an IoT application to ease the mash-up of services and achieve the desired results. It is important to note here that the user of an IoT application is also a part of the ecosystem and generally his/her demands are to be met through the use of services. Thus, there is a need for an user-centric, situational aware discovery technique for an IoT environment [3], [4]. In the earlier works, the service discovery issue in IoT was addressed through the use of semantic knowledge to find similarities among the services [5], [6], context-based service ranking model [7], social networks [8], [9] *etc*. The noticeable drawbacks of these approaches are the scalability, query response and processing time, requirement of enormous storage space, and the difficulty in semantic knowledge gathering. Also, these works do not focus on providing enhanced personalized search results to the user.

In this paper, we address the problem of service discovery from the users perspectives by considering his/her needs (*i.e.*, situation) and develop a progressive search mechanism to efficiently and effectively list the relevant services from the large pool of available services provided by the physical objects. User requirements are categorized into essential and optional requirements where essential requirements are to be met mandatorily while the optional requirements are to be fulfilled in the best possible way. Further, these requirements are mapped to the attributes of the IoT devices and the services to construct a similarity model. Based on the similarity model, primitive and elaborate search strategies are proposed to reduce the size of the search space for a given user query. The former strategy makes use of the essential requirements while the latter operates on the optional requirements to address the user demands. The main contributions of this paper are summarized as follows:

- A progressive search scheme is proposed that filters out the services according to the users demand by effectively reducing the size of the search space, thereby improvising the query response time.
- A similarity model for progressive search scheme is designed based on the requirements of the user to select

978-1-7281-2980-8/19/$31.00 ©2019 IEEE
DOI 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00180

and rank the services. In the primitive search strategy, a semantic-based similarity metric is developed through encoding of ontological concepts to effectively reduce the computational time. For the elaborate search strategy, a multi-dimensional extrinsic similarity score is proposed to fine tune and obtain the personalized search results.

The rest of the paper is organized as follows. In the next section, we provide a review of related works for the IoT service discovery. Section III presents the proposed model for the service discovery. The progressive search scheme along with the similarity calculation methods are discussed in Section IV. In Section V, we evaluate and compare the performance of the proposed progressive search scheme along with the related works. Finally, concluding remarks are given in Section VI.

II. Literature Survey

In this section, we present the recent research works on service discovery in an IoT ecosystem. For each work, the potential benefits and shortcomings are discussed from the different aspects.

Hussein *et al.*, [10] presented a service discovery model that is based on the intellectual analysis of social conditions and users circumstantial demands in an IoT environment. Through the use of objective and subjective context-based properties, user's situation is identified and a list of relevant services is provided. Although when compared to a location-based approach, the proposed scheme returns enhanced set of results and it introduces a computational overhead to calculate the similarity index on the resource-constrained IoT devices. Ma *et al.*, [11] devised a search strategy based on an incremental approach for service discovery in an IoT ecosystem. Three different search schemes in diverse spaces namely feature, spatiotemporal and security are proposed. Based on the assessment of case studies, it is noticeable that the proposed search technique performs better in terms of query response time and accuracy. However, the approach does not deal with the scalability issue of data collection, integration and distribution between the users and IoT devices.

Perera *et al*., [7] developed a sensor selection and ranking model based on the context properties of the IoT devices. By utilizing the user requirements, candidate search results are ranked through a weighted index and heuristic filtering. The advantage of the proposed model is that it helps in minimizing the energy consumption of the sensor network where the search space need not be reconstructed due to the use of distributed processing approach. The shortcomings includes, it demands for a large amount of storage space and also the query execution time is expeditious with an increase in the number of sensors. Cheng *et al.*, [12] proposed an approach to coordinate the interrelated services in an IoT environment based on the situation of the service utilization. To identify the services that are false positive to a given situation, an event-driven action detection algorithm is devised that makes use of mismatch rules. Proposed scheme addresses the heterogeneity and interoperability challenges through the reusability of services. However, the system is not scalable to a large number of services as the processing time to detect and coordinate among the services is intolerable for real-time applications.

Quevedo *et al.*, [13] demonstrated a service discovery mechanism using information-centric network through a forwarding pipeline local network framework. Two types of environments within the network *viz.* restricted and unrestricted are considered to evaluate the service registration and de-registration schemes. Owing to the use of forwarding pipeline architecture, the time taken in both the networks for service composition is the same. But, the authors fail to incorporate appropriate service delivery scheme to guarantee the effective search results. Corbellini *et al.*, [14] implemented a technique to detect the services demanded by the users in an IoT ecosystem based on the association between them. The determined services are assembled into various clusters to facilitate service discovery within a short duration of time in comparison with the content-based approach. As only one cluster is nominated to each service, this methodology does not deal with the overlapping of services in the similarity calculation. Fang *et al.*, [15] conceptualized a system model for "big search" in an IoT ecosystem to develop strategies for service discovery and composition. Search parameters such as temporal, spatial, sentimental *etc.* are taken into account to improvise the search results.

From the above review, it is evident that several IoT service discovery schemes are efficient in terms of query resolution but they do not consider the user's preferences to obtain the personalized search results and are susceptible to the scalability issue of an IoT network. Here, semantic similarity and Quality of Service (QoS) parameters can be utilized to effectively list the user preferred services. In the following paragraphs, we discuss some of the research works that consider these techniques.

Zhao *et al.*, [5] designed a semantic model based on multiple dimensions to find the degree of similarity between the available services in an IoT environment. Through this score, similar services are clustered into groups. Then, an algorithm to determine and substitute the non-existent services with user-intended services are subjected to the clusters. This approach considers both the structure and semantic description of the services that reduces the computation time for similarity score measurement. The disadvantage of the proposed scheme is the use of multi-level parametric based score that consumes more energy when compared to light-weight schemes based on single-level methods. Han and Crespi [6] presented an architecture that facilitates service provision based on the semantic annotations of smart things. A setup procedure to incorporate IoT application onto the web-based on ontological similarity is introduced. With the use of a weighted priority scheduling algorithm, the proposed system effectively handles concurrent queries. However, due to the use of three levels of authorization, there is an overhead in computation and this increases with an increase in the number of services.

Ko *et al.*, [16] formulated a service discovery mechanism for

an IoT environment based on ubiquitous computing paradigm to provide recommendations to the users based on the social, behavioral and temporal aspects. It filters out the services based on the above aspects and on their availability in the environment. The obtained services are then integrated and finally allocated to the users for productive use of resources that adhere to QoS requirements. The merit of this approach is its scalability in terms of the number of services whereas the demerit is the manual effort needed to deal with the ontological operations that are defined in the model. Sasirekha *et al.*, [17] proposed a multi-layered architecture to discover the appropriate services available in the environment based on the contextual knowledge of the user requests. The developed solution suggests the relevant services to the user with the help of Web Ontology Language (OWL) file. The disadvantage is that it is essential to register the node earlier to gain the semantic information of the sensed data from the node. Li *et al.*, [18] put forth a mechanism for service discovery based on the semantics considering QoS requirements and context conditions. The characteristics of this scheme are localized optimization and scalable for a large number of IoT resources. A social-recommendation based model is used to determine the trustworthy services that are further organized. These techniques help in the enhancement of completeness and security of the discovery mechanism. But, there is an overhead of gathering feedbacks from the users to discover the trustworthy services.

As discussed above, semantic and QoS based similarity scores provide an effective solution in the retrieval of user-centric results and thus enhance query resolution and attain effective personalized search results. However, these approaches incur computational burden and are not suitable for the dynamic and large-scale size of an IoT application. In this article, we aim to overcome the above problems by proposing a novel Progressive Search Algorithm (ProSA) to efficiently reduce the search space and thereby decrease the computational complexity and give effective search results to meet the personalized experience in an IoT ecosystem.

III. System Model

This section outlines the proposed progressive search scheme. We describe an application scenario to best explain the usability of the proposed service discovery scheme based on a smart airport application domain of the IoT. In subsequent subsection, we discuss the ontological model and service similarity model that support the progressive search scheme.

A. Application Scenario

Airports are one of the thriving business enterprises in todays urbanized cities where a large number of passengers embark and disembark the flights. Several stakeholders like travelers, visitors, security guards, receptionists, flight attendants, *etc.*, with various kinds of requirements are present here. To increase non-aviation revenue airports have to ensure enhanced customer experience. To this end, there is a need for a service discovery mechanism that provides personalized user results. Several design objectives are to be met to develop such a search system. A lightweight computational model ensures seamless delivery of services, while the scalability of the system is not to be compromised. Also, based on the personal preferences and context the search results are to be fine-tuned.

B. Overall Flow

The execution flow of the proposed service discovery scheme is logically divided into three phases as shown in Fig. 1. Initially, from the user query, essential and optional requirements are extracted. Essential requirements are the services that the user has demanded and they are to be met compulsorily while optional requirements are inherent preferences of the user that are to be resolved in an optimal way. Next, the essential requirements are used to extract a matching set of services through the PSS. It makes use of the ontology-based similarity model to select only the requested services [19]. This step drastically reduces the size of search space. Further, the filtered set of services are then subjected to ESS where the optional requirements of the user are matched with the service through a proximity-based multi-dimensional extrinsic similarity score. Finally, based on the extrinsic scores the services are ranked and returned to the user. To support PSS, an ontological model and an encoding scheme are designed. Through them, the computational time for similarity calculation is drastically reduced. They are discussed in the following subsections.

C. Ontology Model

We have designed an ontological model (with namespace as smart airport services: as) to conceptualize the user requirements and properties of the IoT ecosystem as depicted in

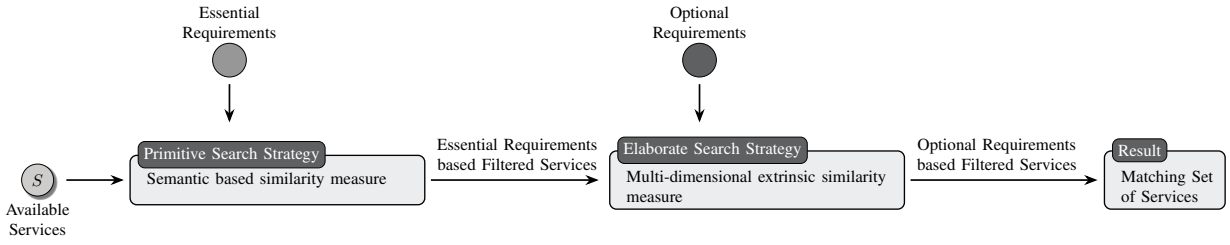


Fig. 1. Overall Execution Flow of the Progressive Search Algorithm

Fig. 2. The model consists of three main concepts of service discovery *viz.* User, Device, Service and other supporting concepts namely Query, Context Properties, Intrinsic and Extrinsic properties, Essential and Optional Requirements. Meaningful relationships have been established between the concepts (*e.g.* User requests for Service, Device offers Service *etc.*) to aid in semantic matching and query resolution. Context Properties class is included to cover the different attributes of the main concepts. Two types of context properties namely intrinsic and extrinsic are utilized to provide the personalized results for the users. Intrinsic properties depict the environmental facets of the users such as time, region, temperature, *etc*. Whereas the personal and communal aspects like individual preferences, trustworthy services, *etc.*, are rendered as the extrinsic properties. To focus on the constraints of the query, essential and optional requirements classes are modeled. Device class comprises of all the devices available in an IoT ecosystem while Service class comprises of the numerous services offered by the different devices.

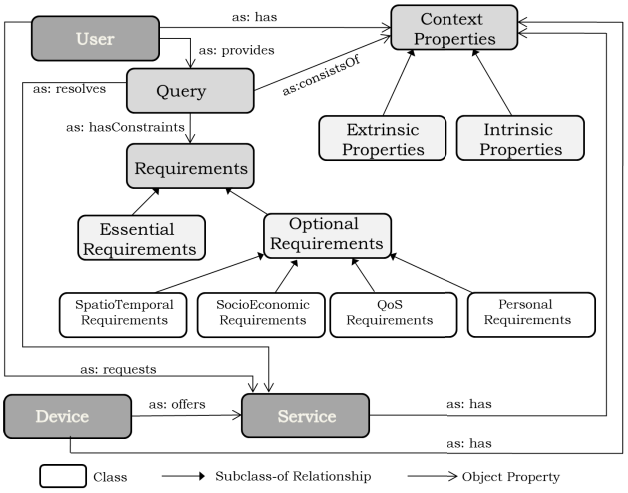


Fig. 2. Smart Airport Services Ontology Model

D. Service Similarity Model

The services demanded by the user are obtained based on the similarity scores computed between the available set of services and the requested ones. PSS makes use of the signature of ontological concepts to calculate the similarity score *i.e.,* semantic similarity. It improvises the searching, indexing and ranking operations for a large and diverse dataset. It's a function of the distance between the terms in a graph corresponding to the hierarchical structure of the underlying ontology.

ESS utilizes the proximity similarity score based on the extrinsic properties. It's calculation deals with the socio-economic, spatio-temporal, QoS and personal context properties. By the application of PSS and ESS in a progressive manner, the computation time necessary for matching the services is adequately decreased. It is evident that the set of services for matching is reduced when compared to the available set.

IV. Progressive Search Algorithm

In this section, we describe the progressive search scheme for service discovery in an IoT ecosystem. Algorithm 1 lists our proposed ProSA. It consists of two search strategies PSS (from lines 3 to 18) and ESS (from lines 19 to 46). In the following subsections, we discuss both the search strategies.

A. Primitive Search based on Essential Requirements

Essential requirements from the user query are utilized by the primitive search strategy to reduce the search space and obtain the services according to the user demands. We develop a signature-based semantic similarity score by leveraging the ontological model discussed in Section III-C. Furthermore, a prime number based encoding scheme is implemented that assigns a unique number (or signature value) to each of the concepts in the ontology [20]. With the help of these unique codes, the similarity between the concepts is computed through numeric operations and thus there is a drastic reduction in the computation time.

Given two services, a requested service (R) and a solution service (S), we define the following types of the match between them.

i) *Perfect Match* ($R \equiv S$): If the signature values of R and S are the same, then it is a perfect match. Here, user requirements are fulfilled completely.

ii) *Partial Match*: In this type of match, the user requirements are met partially. We consider the following sub-types of the partial match:
   a) Plug-in Match ($R \supset S$): If the signature value of R is greater than that of S, then it is a plug-in match *i.e.*, the solution provided is the parent of the requested one.
   b) Subsume Match ($R \subset S$): When the signature value of S is greater than that of R, then it is a subsume match *i.e.*, the solution provided is the child of the requested one.
   c) Satisfactory Match ($R \cap S$): When the signature value of S is a multiple of R's value, then it is a satisfactory match *i.e.*, the solution is a combination of multiple services that supersedes the requested service.

iii) *Irrelevant Match* ($R \neq S$): If the signature value of S is different from that of R, then its an irrelevant match *i.e.*, user requirements are not met here.

Given a user request, we extract the ontological concepts in it to the set R and compare it with a service S from the available service set using the below formula (p is the number of concepts matching between R and S).

$$sim(R,S) = \begin{cases} \sum_{i=1}^{p} \frac{C_{sim}(R,S)}{p}, & p > 0 \\ 0, & p = 0 \end{cases} \quad (1)$$

The semantic similarity between two ontological concepts c_i and c_j is calculated using the equation 2.

Algorithm 1 Progressive Search

Input : User Requirements Set, U_r;
Set of Available Services, S;
Output: Matching Set of Services, R_s;

1: $U_e \leftarrow$ Extract Essential-Requirements(U_r)
2: $U_o \leftarrow$ Extract Optional-Requirements(U_r)
3: $S_e \leftarrow \varnothing$
4: **for each** $u_i \in U_e$ **do**
5: $C \leftarrow$ Map-Requirements(u_i)
6: **for each** $c_j \in C$ **do**
7: **for each** $s_k \in S$ **do**
8: $v \leftarrow$ calculate similarity between c_j and s_k using Eq. 1
9: **if** $v == 1$ **then** # perfect match
10: $S_e \leftarrow s_k \cup S_e$
11: **else if** $v > \theta$ **then** # partial match
12: $S_e \leftarrow s_k \cup S_e$
13: **else** # irrelevant
14: continue
15: **end if**
16: **end for**
17: **end for**
18: **end for**
19: **for each** $u_i \in U_o$ **do**
20: $wu_i \leftarrow$ Extract Weight(u_i)
21: **end for**
22: sort U_o based on W_u in descending order
23: **for each** $u_i \in U_o$ **do**
24: $i \leftarrow$ Extract Ideal Value(u_i)
25: $lb, ub \leftarrow$ Extract Bounds(u_i)
26: $C \leftarrow$ Map-Requirements(u_i)
27: **for each** $c_j \in C$ **do**
28: **for each** $s_k \in S_e$ **do**
29: $d \leftarrow$ Extract Extrinsic Property(s_k)
30: **if** $d == i$ **then** # desired match
31: $R_s \leftarrow s_k \cup R_s$
32: add 1 as weight of s_k to W_s
33: **end if**
34: **if** $d \geqslant lb$ and $d \leqslant ub$ **then**
35: $v \leftarrow$ calculate similarity between c_j and d using Eq. 3
36: **if** $v > \theta$ **then** # favourable match
37: $R_s \leftarrow s_k \cup R_s$
38: add v as weight of s_k to W_s
39: **else** # irrelevant
40: continue
41: **end if**
42: **end if**
43: **end for**
44: **end for**
45: **end for**
46: sort R_s based on W_s in descending order

$$C_{sim}(c_i, c_j) = \begin{cases} 1, & \text{if } perfectmatch \\ 0, & \text{if } irrelevant \\ \frac{\zeta}{|height(c_i) - height(c_j)|}, & \text{otherwise} \end{cases} \tag{2}$$

where ζ is the balancing factor that controls the contribution of the number of levels between the given concepts to the similarity score.

In Algorithm 1, PSS is defined from line 3 to 18. It takes essential user requirements set (U_e) and the set of available services (S) as inputs. For each user requirement, the ontological concepts (classes, instances, data and object properties) related to it are retrieved (through the "Map-Requirements" function). These concepts are then compared with the available services from the IoT ecosystem using equation 1. Based on the similarity score, the concepts are categorized into one of the matches as defined above and added to the filtered service list. A threshold parameter (θ) here controls the addition of services to the filtered set and its value is pivotal to obtain the effective search results. As a higher value drastically eliminates potential services that fail to match the perfect semantic definitions of the user requirement while a lower value will degrade the quality of the filter service set by including irrelevant partially matched services. We set the value for the threshold parameter at 0.7 for our experimental studies.

B. Elaborate Search based on Optional Requirements

As the optional requirements qualify the extrinsic properties of the user, thus they are effective parameters to retrieve the personalized search results. ESS makes use of these features to fine tune the search results. A proximity-based multi-dimensional similarity score ($Prox_{sim}$) is devised based on the extrinsic properties of the user and services to improvise the search results. Equation 3 is used to calculate this score between the user requirement R and a service S.

$$\begin{aligned} Prox_{sim} = & \alpha * SpatioTemporal_{sim}(R, S) + \\ & \beta * SocioEconomic_{sim}(R, S) + \\ & \gamma * QoS_{sim}(R, S) + \\ & \delta * Personal_{sim}(R, S) \end{aligned} \tag{3}$$

where α, β, γ and δ are the weighting parameters for respective proximity measures. These are captured from the user interface.

Based on the extrinsic properties, we categorized four types of proximity requirements. Attributes like location, travel timing come under the SpatioTemporal category. The SocioEconomic category consists of attributes such as cost, frequency, class preference. Properties like user satisfaction rating, availability, accuracy, sensitivity are classified into

the QoS category. The personal category includes attributes namely airlines preference, work type, inquiry area preference. Euclidean and Jaccard distances are employed to calculate the proximity similarity measure by the above-mentioned categories based on their types of attributes (as shown in Table I).

TABLE I
TYPE OF SIMILARITY METHODS
EMPLOYED TO CALCULATE PROXIMITY SCORES

Proximity Measure	Extrinsic Properties	Similarity Method
SpatioTemporal	Spatial (location, physical distance, *etc.*)	Euclidean distance
	Temporal (travel time, flight duration *etc.*)	Euclidean distance
SocioEconomic	Social (community, frequency, class preference, *etc.*)	Jaccard distance
	Economical (cost, discounts, royalty points, *etc.*)	Euclidean distance
QoS	User satisfaction rating, availability, accuracy, *etc.*	Euclidean distance
Personal	Preference, work type, *etc.*	Jaccard distance

In Algorithm 1, ESS is defined from line 19 to 46. Here, the optional requirements of the user query (U_o) and essential requirements based filtered service set S_e are taken as inputs. From the user query, for each optional requirement, the ideal value along with permissible upper and lower bounds are retrieved. Further, the optional requirements are mapped to the ontological concepts. Then, extrinsic properties for each service in S_e set are extracted and compared with the user sought values (i). If they are equal, then we define such a match as "Desired Match" and add it the result set (R_s) by assigning the highest weight to the matched service. Otherwise, the bounds (upper bound, ub and lower bound, lb) are taken into consideration and if the extrinsic value of the service falls within this limit, then the proximity similarity between them is calculated using equation 3. Here, we again make use of the threshold parameter (θ) to determine the best possible match. Such a match is known as "Favourable Match" and the proximity similarity score is added as it's weight. Finally, we define "Irrelevant Match" for the services with extrinsic properties that do not fit within the specified bounds and are discarded. The result set is then sorted based on the weights of the services and returned to the user.

V. EXPERIMENTS AND RESULTS

This section describes the implementation details along with the experiments conducted to measure and evaluate the performance of the proposed progressive search algorithm.

A. Implementation Details

We have developed a prototype search system, called *"ProSA Search Tool"* for the proposed model using Java language. Fig. 3 depicts the user interface of the prototype system. Through this interface, the user specifies his/her requirements. Essential requirements are to be given as keywords while for the optional requirements, there are checkboxes, sliders, and textboxes to select, assign weights and bounds. The ontology model proposed in Section III-C is designed through Protégé toolkit (Version 5.3.0) [21] and further managed through Apache Jena Framework (Version 3.9.0) [22] and SPARQL [23]. Also, Apache's Commons Mathematics [24] and Text [25] libraries are used for determining the semantic and proximity similarity scores. To conduct the various experiments, we used a real and synthetic dataset to populate the ontology (as at present no single source provides the real-time IoT dataset for the smart airports). Services and their properties for the airport were collected from [26] while random values are generated for the users and devices. To evaluate and measure the performance of the proposed search system, we used a computer with Intel(R) Core i5-4200M 2.50GHz processor and 8GB RAM.

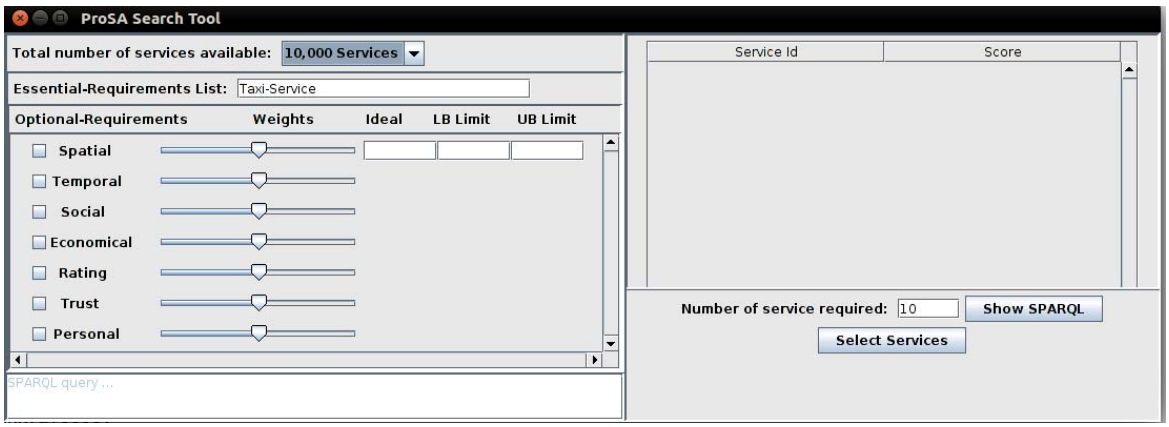
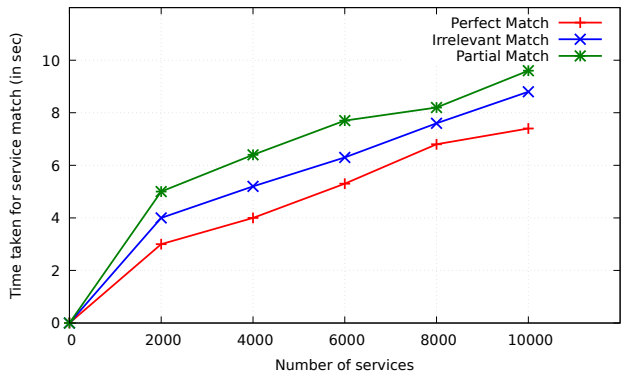
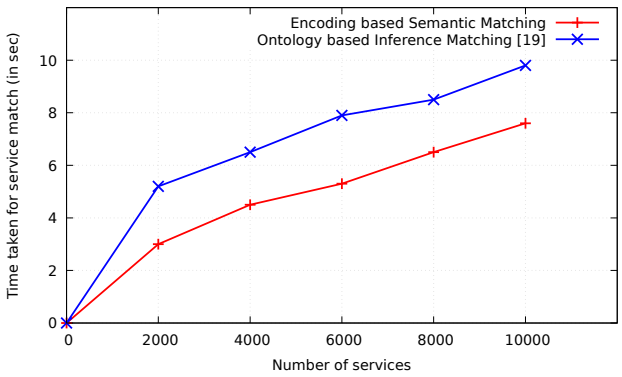


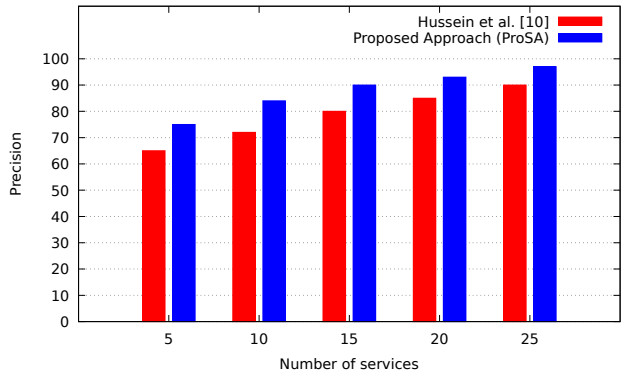
Fig. 3. ProSA Search Tool



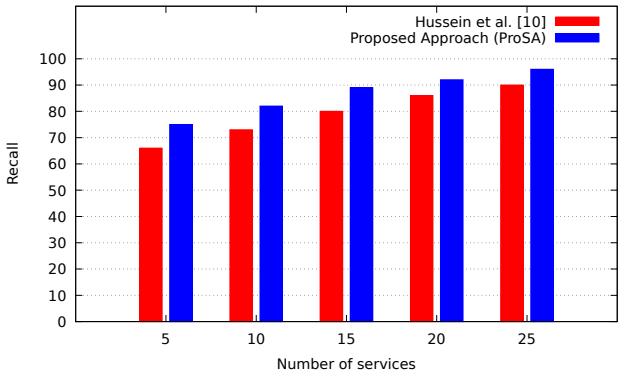
(a) Query Processing Time for Different Signature-based Semantic Matches



(b) Comparison of Query Processing Time



(c) Precision Ratio Comparison with Varying Number of Available Services



(d) Recall Ratio Comparison with Varying Number of Available Services

Fig. 4. Experimental results of ProSA. Processing time, precision and recall ratios are measured with varying size of available services to determine the efficiency and effectiveness of the proposed work.

B. Experimental Setup

To appreciate the practicability of the proposed service discovery scheme, we conducted several experiments. We have considered three queries for testing *i.e.*, Query 1 - Find a coffee vending machine that offers a particular type of beverage, Query 2 - Find the less-crowded waiting lounge near me, and a generic query, Query 3 - List the services that will ease my transit at the airport today. These queries are run on the ProSA search tool and we report the mean of 10 executions. The effectiveness of the search system is measured by two evaluation parameters namely recall and precision. They are calculated as follows:

$$Recall = \frac{N_{ase}}{T_{as}}$$

$$Precision = \frac{N_{ase}}{T_{se}}$$

where N_{ase} is the number of appropriate services extracted, T_{as} is the total number of appropriate services available in the repository and T_{se} is the total number of services extracted from the repository.

These two measures are the performance evaluators of the service discovery approach. Precision is the division of the number of appropriate services extracted over the total number of appropriate services in the repository whereas recall is the division of the number of appropriate services extracted over the total number of extracted services.

C. Results and Performance Evaluation

In the first experiment, we measure the query processing time for different types of semantic matches (*i.e.*, perfect, partial and irrelevant) that are proposed in Section IV-A for a varying number of the available services. Fig. 4a presents the result for this experiment. Here, it is observed that comparatively less time is taken for identifying the perfect match when compared to partial and irrelevant match. Partial match consumes more time in comparison with irrelevant as it involves three levels of computation for the plug-in, subsume and satisfactory matches.

In the second experiment, we evaluate the efficiency of the ProSA with respect to the computation time. Our proposed approach utilizes an encoding scheme to facilitate semantic match, we compare this technique with the ontology-based

inference matching [19] as shown in the Fig. 4b. It is evident from the comparison that application of the signature-based semantic scheme in the proposed approach is efficient than the ontology-based inference matching mechanism. This is attributed to the fact that similarity computation in our work involves only algebraic operations owing to the numeric encoding of the ontological concepts, while the inference technique involves the time-consuming concept logical calculations.

In the last experiment, we evaluate the effectiveness of our proposed work. Here, we consider a previous work on dynamic service discovery proposed by Hussein *et al.* [10] for comparison. We populated the ontology with 25 services for each query and noted the number of relevant and irrelevant responses. As seen in Figs. 4c and 4d, the precision and recall ratios of the proposed approach are better when compared to the existing work. The reason is that our work considers the extrinsic and intrinsic properties of the services and user to retrieve relevant search results and thus offers personalized user experience.

VI. Conclusions

With the penetration of the IoT, a large number of physical objects are now getting connected to the Internet. Search and discovery of the services offered by these objects is of prominent importance for an IoT application. Thus, numerous techniques have been proposed in the past to address this challenge. However, they do not consider user experience and personalized search results. In this paper, we present a progressive service discovery mechanism along with its related search strategies and similarity computation model to effectively list the user desired services. A proof of concept prototype, ProSA Search Tool, for the smart airport domain of the IoT is implemented to demonstrate the applicability of our proposed approach. The results from experimental studies demonstrate that the ProSA technique is highly scalable and consumes less time due to the use of signature-based semantic matching model. Also, when compared with the previous work our approach yields effective search results through the proximity similarity model and thus provides an enhanced user experience in an IoT ecosystem.

References

[1] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Large-Scale Indexing, Discovery, and Ranking for the Internet of Things (IoT)," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–52, 2018.

[2] M. Nitti, V. Popescu, and M. Fadda, "Using an IoT Platform for Trustworthy D2D Communications in a Real Indoor Environment," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 234–245, 2019.

[3] G. Shinde and H. Olesen, "A Survey on Service Discovery Mechanism," *in Proceedings of the Intelligent Computing and Information and Communication Conference*, pp. 227–236, 2018.

[4] S. Pattar, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "Searching for the IoT Resources: Fundamentals, Requirements, Comprehensive Review, and Future Directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2101–2132, 2018.

[5] S. Zhao, L. Yu, B. Cheng, and J. Chen, "IoT Service Clustering for Dynamic Service Matchmaking," *Sensors*, vol. 17, no. 8, p. 1727, 2017.

[6] S. N. Han and N. Crespi, "Semantic Service Provisioning for Smart Objects: Integrating IoT Applications into the Web," *Future Generation Computer Systems*, vol. 76, pp. 180–197, 2017.

[7] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 406–420, 2014.

[8] B. Yuan, L. Liu, and N. Antonopoulos, "Efficient Service Discovery in Decentralized Online Social Networks," *Future Generation Computer Systems*, pp. 73–78, 2017.

[9] M. S. Roopa, S. Pattar, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "Social Internet of Things (SIoT): Foundations, Thrust Areas, Systematic Review and Future Directions," *Computer Communications*, vol. 139, pp. 32–57, 2019.

[10] D. Hussein, S. N. Han, G. M. Lee, N. Crespi, and E. Bertin, "Towards a Dynamic Discovery of Smart Services in the Social Internet of Things," *Computers & Electrical Engineering*, vol. 58, pp. 429–443, 2017.

[11] H. Ma and W. Liu, "Progressive Search Paradigm for Internet of Things," *IEEE MultiMedia*, vol. 25, pp. 76–86, 2017.

[12] B. Cheng, M. Wang, S. Zhao, Z. Zhai, D. Zhu, and J. Chen, "Situation-Aware Dynamic Service Coordination in an IoT Environment," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2082–2095, 2017.

[13] J. Quevedo, C. Guimarães, R. Ferreira, D. Corujo, and R. L. Aguiar, "ICN as Network Infrastructure for Multi-Sensory Devices: Local Domain Service Discovery for ICN-based IoT Environments," *Wireless Personal Communications*, vol. 95, no. 1, pp. 7–26, 2017.

[14] A. Corbellini, D. Godoy, C. Mateos, A. Zunino, and I. Lizarralde, "Mining Social Web Service Repositories for Social Relationships to Aid Service Discovery," *in Proceedings of the 14th International Conference on Mining Software Repositories*, pp. 75–79, 2017.

[15] B. Fang, Y. Jia, X. Li, A. Li, and X. Wu, "Big Search in Cyberspace," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1793–1805, 2017.

[16] I.-Y. Ko, H.-G. Ko, A. J. Molina, and J.-H. Kwon, "SoIoT: Toward a User-Centric IoT-based Service Framework," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 2, pp. 1–21, 2016.

[17] S. Sasirekha, S. Swamynathan, and S. Keerthana, "A Generic Context-Aware Service Discovery Architecture for IoT Services," *in Proceedings of the International Conference on Intelligent Information Technologies*, pp. 273–283, 2017.

[18] J. Li, Y. Bai, N. Zaman, and V. C. Leung, "A Decentralized Trustworthy Context and QoS-Aware Service Discovery Framework for the Internet of Things," *IEEE Access*, vol. 5, pp. 19 154–19 166, 2017.

[19] B. Jia, W. Li, and T. Zhou, "A Centralized Service Discovery Algorithm via Multi-Stage Semantic Service Matching in Internet of Things," *in Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, vol. 1, pp. 422–427, 2017.

[20] D. Preuveneers and Y. Berbers, "Prime Numbers Considered Useful: Ontology Encoding for Efficient Subsumption Testing," *Techical Report CW464., Department of Computer Science, Katholieke Universiteit Leuven, Belgium*, 2006.

[21] Protégé Tool. Accessed: Apr 10, 2019. [Online]. Available: https://protege.stanford.edu/

[22] Apache Jena. [Online]. Available: https://jena.apache.org/

[23] SPARQL Query Language. Accessed: Apr 10, 2019. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[24] Apache Commons Math. Accessed: Apr 10, 2019. [Online]. Available: https://commons.apache.org/proper/commons-math/

[25] Apache Commons Text. Accessed: Apr 10, 2019. [Online]. Available: https://commons.apache.org/proper/commons-text/

[26] Airport Quarterly Passenger Survey Dataset. Accessed: Apr 10, 2019. [Online]. Available: https://catalog.data.gov/dataset/airport-quarterly-passenger-survey