This chapter presents the review of literature related to the theme of the thesis mentioned in Chapter 1. At first, approaches related to performance optimization of service discovery are discussed in general. Secondly, literature related to two main methods for optimizing the performance of discovery, namely, clustering based approaches and indexing based approaches are reviewed. Thirdly, research works related to the methods of finding semantic relations among concepts of services are discussed. Fourthly, works related to methods of optimizing the performance of QoS based selection are reviewed. Fifthly, difficulties in constructing queries during discovery are analyzed with existing methods. Ultimately the gaps in existing research areas are identified towards defining the problem statement of this research.

2.1 Optimization Approaches to Semantic Discovery (in General)

From research works [23-25] semantic service discovery which uses reasoning on specific ontologies is found to consume high response time of typically 4 to 5 seconds to perform a single service match of around 10 concepts. Several optimization methods are being proposed to reduce the response time of semantic service discovery. As mentioned in Chapter 1, in any semantic web service discovery infrastructure (please refer Fig.1.5), there are two core components, namely, service matcher and ontology reasoner. Performance optimization can be introduced either in semantic matcher or on ontology reasoner or in both. According to our survey [33], we classify various optimization approaches into two categories, namely, optimization by efficient reasoning and optimization by efficient matching.

2.1.1 Optimization by Efficient Reasoning

The primary objective of this method is to optimize the ontology reasoner so that the time involved in loading and classifying ontologies is reduced. Pre-classifying the ontologies for anticipated user queries and numerical encoding of concepts are the two
prominent methods used to optimize the performance of a reasoner. To implement optimization at reasoning component, one should work with the internal architecture of reasoner. As this method of optimization depends on the internal architecture, the optimization method varies from reasoner to reasoner. Hence, optimization at matcher is preferred to optimization at reasoner.

2.1.2 Optimization by Efficient Matching

The primary objective of this method to optimize the performance of service matcher by filtering out the services which are obviously irrelevant to a given query so that the semantic service discovery will be employed only to those services which really need complete semantic matching of service capabilities. With this kind of optimization, the number of services that undergo ontology reasoning is significantly reduced. Optimization by efficient matching is further sub categorized as follows.

- Optimization by clustering
- Optimization by inverted indexing
- Optimization by caching
- Optimization by hybrid approaches
- Optimization by efficient data structures
- Optimization by efficient matching algorithms

2.1.2.1 Optimization by Clustering

This approach initially organizes all the available published services into groups of similar services using a clustering algorithm. Each cluster is associated with a cluster centre which is the representative of all the services in that cluster. Once the clusters are formed, when a query is submitted, during matching, firstly, the cluster which is relevant to the query is found out by computing the distance between the query and each cluster centre. The cluster whose centre is very close to the query is identified as the relevant cluster. Now it is sufficient to perform complete semantic matching of service capabilities only to those services which are present in the relevant cluster. Hence, for a given query, the search space is reduced from an entire pool of services to a specific cluster. With clustering approach, the time taken to find matched services for a given query will be equal to the sum of time involved in finding the relevant cluster and time involved in matching the services present in that relevant cluster with the query. Research works related to clustering based discovery are reviewed with special focus in the later section of this chapter.
2.1.2.2 Optimization by Inverted Indexing
Index is basically a sorted list of key-value pairs in which each key is mapped to its related value. Indexing speeds up the searching of an object by locating it with its key-value rather than searching object by object. Indexing is used for performance optimization especially while retrieving records from databases. Existing indexing based discovery approaches are reviewed with special focus in the later sections of this chapter.

2.1.2.3 Optimization by Caching
Caching technique is employed for performance optimization in several areas of computing. A Semantic Discovery Caching (SDC) which captures the knowledge on the functional usability of all available services and exploits this cache for finding matched services of queries has been proposed in [34].

2.1.2.4 Optimization by Hybrid Methods
This approach uses hybrid techniques which combine non-logic methods such as syntactic, graph and Information Retrieval (IR) approaches with description logic based matching instead of finding matched services using purely logic based semantic matching. The syntactic and IR methods are helpful to remove services which are irrelevant to a given query quickly without employing semantic reasoning for them. WSMO-MX [35] is a hybrid matchmaker for WSML oriented services. It combines F-Logic reasoning with syntactic similarity where each of them alone would fail. The experimental results with WSMO-MX showed that the method outperforms pure logic based matching in terms of computation time. Another hybrid matcher OWLS-MX [36] performs semantic service matching with five different filters, exact, plug-in, subsumes, subsumed-by and nearest-neighbor of which the last two are hybrid methods. The last two methods employ IR similarity along with a user defined threshold. Experimental results with OWLS-MX show that under certain constraints, the matcher outperforms the purely logic based matching in terms of performance and scalability. SAWSDL-MX [37] is a hybrid matchmaker for SAWSDL oriented services, which combines pure logic based matching with text retrieval strategies. The experimental results of the method show that the hybrid method using cosine similarity yields better performance.

2.1.2.5 Optimization by Efficient Data Structures
This method improves the efficiency of finding matched services by storing services in efficient data structures. A discovery method proposed in [38] uses tree-form data
structure for storing service information in which upper nodes are given more weight than lower nodes. The method stores properties specific to a service called as, Special Properties (SP) in the upper nodes and Common Properties (CP) such as service ID in the lower nodes. With this method, if the upper nodes have been matched in searching, the lower nodes could be omitted which leads to improved efficiency. A data structure, \textit{Double Parameter Inverted File (DuoParaInvertedFile)} has been proposed in [39] to facilitate mutual search operations among service operations, input and outputs.

\subsection*{2.1.2.6 Optimization by Efficient Matching Algorithms}

The primary objective of this method is to improve the performance of matching with the help of efficient algorithms. In contrast to conventional matching algorithm proposed by Paolucci, et al., [22], various other approaches are being proposed to improve both the accuracy of matching and execution time of matching. An improved matchmaking algorithm based on bipartite graph proposed in [40] offers a much better performance than Brute-Force technique. A two-phase semantic service matching method proposed in [41] improves the performance of semantic discovery by first finding the functionally matched services in the first phase and filtering the functionally matched services quantitatively in the second phase.

\subsection*{2.2 Optimization Approaches to Semantic Discovery (With Special Focus)}

Of many types of optimization approaches we inspired by clustering based discovery as clustering provides a special feature of organizing services as groups of similar characteristics in an unsupervised method. Organization of services as clusters lays foundation for any analysis and service mining. Service clusters will act as cloud of services to the entire Service Oriented Computing (SOC) community.

\subsubsection*{2.2.1 Existing Clustering based Approaches}

A method is proposed in [42] clusters similar services based on the heuristics that \textit{`parameters tend express the same concept if they occur together often’}. This method differs from other methods by using unsupervised learning at operation level rather than supervised classification at service level. The authors implemented the approach in Woogle, a web search engine and conducted experiments with over 1500 operations. The approach is found to yield better precision and recall when compared to other methods which consider words in operation names and words in service names. Another method presented in [43] clusters services by finding the composite relations
among the elements of WSDL and their useful information among different combinations. Another WSDL based method presented in [44] uses hierarchical agglomerative clustering to group services based on textual similarity.

In contrast to [42-43] which do not consider service name and service documentation while clustering, the method presented in [45] considers service name, service documentation and parts of WSDL elements while computing various similarities such as element similarity, segment similarity, WSDL similarity and web services similarity. Further, it defines a preprocessing method which considers the programming style and naming rules before clustering. SCAN (Structural Clustering Algorithm for Networks) algorithm based clustering method is used to cluster the services. The method presented in [46] transforms service descriptions by mapping WSDL information into a richer semantic representation language, OWL-S. It uses Jaccard method [47] for finding similarity among services and computes similarity between two services using

\[
Sim_{ws_1-ws_2} = w_1 \times SimDes_{ws_1-ws_2} + w_2 \times SimSerP_{ws_1-ws_2} + w_3 \times SimWSDL_{ws_1-ws_2} + \\
w_4 \times SimPModel_{ws_1-ws_2} + w_5 \times SimGround_{ws_1-ws_2}
\]

In (2.1), \(Sim_{ws_1-ws_2}\), \(SimDes_{ws_1-ws_2}\), \(SimSerP_{ws_1-ws_2}\), \(SimWSDL_{ws_1-ws_2}\), \(SimPModel_{ws_1-ws_2}\) and \(SimGround_{ws_1-ws_2}\) represent overall, description, OWL-S profile, WSDL, OWL-S process and OWL-S grounding similarities respectively. This method presets the values of \(w_1, w_2, w_3, w_4\) and \(w_5\) as 0.1, 0.3, 0.2, 0.1 and 0.1 respectively. Using hierarchical agglomerative clustering, services are clustered and the resulting clusters are stored in an UDDI registry. This method basically extends the semantic representation of services for grouping similar services. With this enhanced semantics and groupings, searching within service repositories becomes more intuitive. Another similar method proposed in [48] extends the semantic representation of services for clustering similar services and enhances the searching in UDDI. This method also uses Jaccard method for finding dissimilarity among services and computes dissimilarity between two services using

\[
DisSim_{ws_1-ws_2} = w_1 \times DisSimDes_{ws_1-ws_2} + w_2 \times DisSimSerP_{ws_1-ws_2} + \\
w_3 \times DisSimWSDL_{ws_1-ws_2}
\]

In (2.2), \(DisSim_{ws_1-ws_2}\), \(DisSimDes_{ws_1-ws_2}\), \(DisSimSerP_{ws_1-ws_2}\) and \(DisSimWSDL_{ws_1-ws_2}\) represent overall, description, OWL-S profile, and WSDL similarities respectively.
Chapter 2: Literature Review

The weights are preset as $w_1=0.5$, $w_2=0.3$ and $w_3=0.2$. The method clusters services using single linkage algorithm. Also, this approach presents an algorithm for finding matched services by scanning each service present in the cluster which is most similar to the query.

In the approach presented in [49], the services are converted into standard vector form with the concept of position weight of WSDL. The approach describes a self-organizing neural network based learning algorithm and clustering algorithm to classify the services into different categories automatically.

The research works [50] [51] use Star-Clustering algorithm to organize the services returned by a Service Search Engine. Here clustering is used to organize the search results of a search engine as clustering view is more useful than the traditional ranked list. These methods do not employ clustering for discovery of services.

The approach [52] presents a clustering based approach to efficiently find appropriate services from large volumes of existing services. But, this method uses WordNet based similarity to find similarity among services. This is applicable to WSDL services.

The approach [53] describes an architecture for clustering and filtering semantic web services which notifies service client about any new service that best fits his functional and non-functional requirements. The method suggests STAR clustering and describes a way to represent the medioid of the clusters. This method represents the most general service’s profile as the medioid of a cluster with factor of generality is computed as the weighted average of inputs, outputs and non-functional parameters. This approach mainly focuses on matching a new incoming service with existing medioids of service taxonomy and classifies the new service under appropriate cluster.

In [54] an extended OWL-S/UDDI matchmaker architecture which uses semantic-based clustering approach and Automatic Knowledge Acquisition (AKA) module is presented to enhance service discovery. While clustering, this method annotates semantics to the concepts of inputs and outputs of services and computes similarity between any two services based on Jaccard similarity and distances between concepts in the ontology models. It computes similarity between two services $ws_1$ and $ws_2$ using

$$Sim_{ws_1-ws_2} = w_1 \times SimDes_{ws_1-ws_2} + w_2 \times SimInput_{ws_1-ws_2} + w_3 \times SimOutput_{ws_1-ws_2} \quad (2.3)$$
Chapter 2: Literature Review

In (2.3), $\text{Sim}_{\text{ws}_1-\text{ws}_2}$ represents the accumulative similarity coefficient between a pair of services, $\text{ws}_1$ and $\text{ws}_2$, $\text{SimDes}_{\text{ws}_1-\text{ws}_2}$ represents the similarity co-efficient for description, $\text{SimInput}_{\text{ws}_1-\text{ws}_2}$ represents the similarity co-efficient for inputs and $\text{SimOutput}_{\text{ws}_1-\text{ws}_2}$ represents the similarity co-efficient for outputs. Further, the values of $w_1$, $w_2$ and $w_3$ denote the weights representing significance to each corresponding co-efficient and the values of $w_1$, $w_2$ and $w_3$ are preset to 0.5, 0.2 and 0.3 respectively.

It employs single linkage algorithm to cluster services and the clusters are stored in the UDDI registry. It extended UDDI architecture with ontology repository which stores the vocabularies for a wide variety of domains. The ontology definitions are modeled in Ontology Matcher using Jena API. An incoming query is matched with the concepts in the ontology model and extracts the matched ontology model. If the query is not included in the ontology model, then Automatic Knowledge Acquisition extends the query with the help of dictionary processor and WordNet and generates extended ontology model. Then the outputs of the query are matched with the outputs of clusters in order to find the matched cluster of the query. Then the method compares each service present in the matched cluster with the query.

In [55], the functional similarity between two services is computed using

$$F_{\text{sim}}(S_i,S_j) = w_1 \times \text{sim}(\text{OP}_i,\text{OP}_j) + w_2 \times \text{sim}(\text{input}_i,\text{input}_j) + w_3 \times \text{sim}(\text{output}_i,\text{output}_j)$$

In (2.4), $F_{\text{sim}}(S_i,S_j)$ represents functional similarity between $S_i$ and $S_j$, $w_1\times\text{sim}(\text{OP}_i,\text{OP}_j)$ represents the similarity in description of service function, $w_2\times\text{sim}(\text{input}_i,\text{input}_j)$ represents similarity of inputs and $w_3\times\text{sim}(\text{output}_i,\text{output}_j)$ represents similarity of outputs. This method uses Chameleon algorithm to cluster the services. This algorithm builds K-Nearest Neighbour (K-NN) graph from the similarity matrix. Then it partitions the graph to find initial sub clusters. Ultimately the algorithm merges any two clusters only when the inter-connectivity between the clusters is high relative to the internal inter-connectivity and closeness of the items within the clusters. Its main focus lies in filtering functionally chosen services by a factor, called User Experience Degree (UED) and the computation of UED is based on the non-functional properties of services.
The method presented in [56] combines syntactic analysis with a clustering semantic approach for discovering WSDL services using UDDI. For a given query, initially this method eliminates irrelevant services using Vector Space Model to find working dataset. Then Probabilistic Latent Semantic Analysis Approach (PLSA) is applied to the working dataset to find common matching concepts between services in the working set and query. This approach uses K-Means clustering to cluster services. Another K-Means clustering based approach is presented in [57]. In [58] a genetic algorithm based clustering approach is discussed.

The work proposed in [59] clusters and characterizes services semantically by the output parameters. The method treats the parameters of a service as a set of elements with each element is a duality tuple. Each element has entity_name and an attribute set, represented as $<\text{Entityname, attributeset}>$. The method transforms the parameters into multi-dimensional vector and finds the similarity between services based on the number of common attributes between them. Similarity between two services is computed as the ratio of number of common attributes between them to the geometric mean of the respective number of attributes of the concerned services. During querying, the approach finds the most similar cluster of the query. Then it employs semantic matching to each service present in the most similar cluster of the query to find semantically matched services of the query. The method presents only the theoretical treatment and it lacks experimental evaluations. The work did not focus on accuracy or quality of cluster partitions.

In [29] an ant-inspired clustering approach is used to cluster large sets of services based on the semantic similarity between the service descriptions. This approach mainly focuses on the computation of semantic similarity and clustering services using ant based algorithm. It computes the similarity based on the relations that exist among the concepts and the properties of concepts.

### 2.2.2 Existing Index based Approaches

There are few approaches which use index technique during service discovery. Approaches such as [60-62] maintain an index of output parameters of services. These methods identify a set of atomic services or parallel combination of services (candidate services) which can deliver the outputs as required by the query with the help of the index. After finding candidate services, the inputs of each candidate service will be matched with that of query in order to discover matched services for a given query. The
method [63] presents an approach for integrating semantic features of services into UDDI and discovers matched services of a query based on semantic similarity of service properties. To quicken the service discovery, this method uses an ontology concept index and similarity data table. In the similarity data table each ontology concept is mapped to its related concepts which have similarity values higher than a minimum value. The ontology concept index maps each concept with its occurrence locations (service name, inputs and outputs) in published services. When a query comes, the semantically related concepts of each concept of the query are obtained from similarity data table. Then, services which contain the query concepts and their related concepts are retrieved from the ontology index. The method [28] stores the subsumption relations among concepts in a numerically encoded format and uses two R-Trees to index the intervals associated with the input and output parameters of services to enhance the performance of discovery.

2.3 Existing Methods to Handle Similarity Demands

Semantic services are described using description logic formalism and during matching, correspondingly description logic based reasoning is used to determine semantically matched services of the query. Examples for description logic based semantic matchmaking include [19] [22] [64] [65] [66]. The above approaches meet similarity demands of clients with the help of four standard levels of Degree of Match (DoM), namely, equivalent, plugin, subsume and fail. Besides the above pure DL based discovery approaches there are other hybrid approaches such as [36] which combines syntactic Information Retrieval (IR) based similarity measures with pure DL based reasoning. In hybrid methods syntactic IR based similarity will produce the outcome of the matching when pure DL matching produces fail. In this manner, the similarity demands of clients are met through the standard subsumption relations.

2.4 Existing Methods to Handle Inflexibility in Querying

In semantic web services, the intended semantics of services are expressed through ontologies and the purpose of explicit semantics in services is to provide high retrieval accuracy and automation in discovery. But semantic matching of services imposes a condition to service clients that while constructing queries a client should use the same ontologies as used by the service providers. That is if a service provider uses an ontology say concept.owl to express while describing the semantics of an output...
parameter, say price, then the client is expected to construct or express his required output using the same ontology concept.owl. But clients tend to express their queries using terms in their own ontologies or even without ontologies. Also, in some situations clients may not be very clear in expressing their needs. The work [26] allows clients to specify their queries in their own ontologies and performs mapping from user ontologies to relevant domain ontologies used in services. Though [26] addresses the above issue, the process of creating and maintaining domain ontologies is challenging and costly. Further, it requires special expertise. This situation has to be addressed.

2.4.1 Existing WordNet based Approaches to Service Discovery

As an alternate to [26] which provides flexibility in querying by mapping user’s ontologies to relevant domain ontologies, our idea is to analyze generic ontology based approaches so that the process of creating and maintaining domain ontologies can be prevented. From a study with a collection of service descriptions (publicly available WSDL services and OWL-S Service Retrieval Test Collection Version 3.0), it is found that the parameters of services are named using meaningful English words. So, we analyze the literature related to applicability of WordNet [67] in finding similar services. In service discovery WordNet is extensively used to expand queries or to extend Vector Space Model. To handle the inadequacy of VSM which considers words only at syntactic level, the method [68] expands the query and concept descriptions in WSDL using their synonyms from WordNet. To avoid excessive sparsity of feature vector that arises while expanding a query, a novel WordNet powered feature vector extraction is presented in [69]. Though this method reduces the dimension and sparsity of feature vectors, it achieves only 2% increase in precision when compared to VSM.

The approach presented in [46] transforms WSDL services into a richer semantic representation language, OWL-S and it enhances the semantics of the terms of WSDL and OWL-S using their synonyms from WordNet. Similarly the approach [70] presents a collaborative tagging-based environment for service discovery. In this approach users are allowed to tag or annotate a service with a keyword or a tag for which synonym sets are obtained using WordNet. Another method [71] classifies the properties of services into four types, namely, Common Properties, Service Properties, Service Interface and Quality of Service. It uses WordNet and HowNet powered similarity measure to find similarity of strings used while describing common properties such as service name, service key, service description, service owner, etc.
A suite of methods, namely, WordNet-powered VSM, WSDL structure matching, similarity among WSDL terms based on semantic distance in WordNet (called semantic structure matching), a combination of WordNet-powered VSM and semantic structure matching are discussed for service discovery in [72]. Out of these four methods, combination of WordNet-powered VSM and semantic structure matching is found to be more precise for discovery.

There are some methods which use WordNet based semantic similarity for service discovery. A technique that combines WordNet with WSDL is presented in [73] for service composition. In this method, senses of parameter names from WSDL services are obtained from WordNet and the senses are expanded using synonyms, hyponyms, hypernyms, meronyms and holonyms. In another work [74], the ‘message’ parts of WSDL are annotated with synsets from WordNet and matched services for a given query are discovered using different WordNet similarities. Experimental results show that the Wu-Palmer method of computing similarity is faster than other methods. WordNet similarity proposed by Leacock and Chodorow [75] is used in another work [52] to cluster similar services.

2.5 Existing QoS Based Approaches

There are two major methods to handle QoS based service selection. They are global planning and local selection [30].

2.5.1 Global Planning Methods for QoS based Selection

Global approach performs service selection at composite service level. In this approach, one service from each service class is selected to form a composite service. Then the QoS attributes of composite service are computed and tested for compliance against QoS constraints requested by the user. In global approach the number of possible service combinations involved in finding the optimal combination is given as $l^n$, where $n$ denote the number of service classes and $l$ denote the number of services present in each service class. The global methods such as Linear Programming (LP) or Mixed Integer Linear Programming (MILP) [76] [77] [78], heuristics [79], [80] are time consuming due to exponential growth of service combinations. This restricts the usage of global methods for dynamic service composition applications. Though the other methods such as genetic algorithm [81], Ant Colony Optimization [82] and hybrid methods [83] [84] focus on reducing the computation time, they are applicable only to service composition with limited number of services. A broker based
framework for QoS-Aware Service Composition with end-to-end constraints presented in [85] gives less attention to time complexity issue.

2.5.2 Local Selection Methods for QoS based Selection

Nowadays as alternate to global methods, local selection methods are widely used to reduce the time complexity of service selection. Local selection methods achieve time reduction by dividing the problem of selecting service combination for a given workflow into a set of independent task level sub problems. Local selection method treats selecting services for the workflow as main problem and selecting service for individual task in the workflow as sub problem. In this approach, the possible number of service combinations of \( n \) service classes with \( l \) number of services in each service class is only \( l \times n \). In local selection methods such as [31], [86-91] sub problems are constructed by decomposing the given global (workflow level) constraints into local (task level) constraints and assigning the local constraints to individual tasks. Each sub problem is resolved to select service having maximum utility for each task satisfying the local constraints of the task. The services selected for each sub problem is combined to produce the appropriate service combination for the given main problem.

2.6 Gap Analysis

This thesis gives its special focus to clustering and indexing based approaches for enhancing the performance of discovery. Clustering based approaches assist in organizing services as groups of similar services which helps in eliminating irrelevancy of a query. Also service clusters serve as the basis for any kind of service mining and analysis. Index based approaches are useful for improving performance of discovery by providing look up facility with key. In this section we analyze the limitations of existing clustering and indexing based approaches to discovery.

2.6.1 Limitations of Existing Clustering based Approaches

The methods [42-47] are applicable only to WSDL services. The method presented in [45] uses SCAN algorithm for clustering WSDL services which accepts 3 parameters \( \sigma \) (similarity threshold between two services for connecting the services using an edge), \( \epsilon \) (a threshold applied while assigning cluster membership) and \( \mu \) (minimum number of neighbours in a cluster). Clustering produced by this algorithm is greatly influenced by these parameters and specifying precise value for theses parameters is difficult. Further,
the methods [46], [48] have their main focus on enriching WSDL services in UDDI registry with semantic representations. The method [49] is found to partition WSDL based services into known classification. The clustering algorithm used in [49] accepts two parameters, namely ‘m’, the number of categories and ‘n’, the number of terms that represent services within ‘m’ categories. The values of m and n influence the quality of clustering and the method converts services into vector terms semi automatically. Despite the method uses a self learning and clustering algorithms, the method partitions services into known classification. The approach presented in [52] describes WordNet similarity based clustering for WSDL services.

The methods [50] and [51] are found to use clustering technique to organize the results returned by a search engine and not for discovery of services.

Though the method [53] addresses notifying clients about new services and labeling medioids using factor of generality, the method presents only theoretical aspects and it lacks experimentation part.

In [55] to generate $K$-NN graph, the value of $K$ should be specified as one the input parameters to the algorithm which influences the accuracy of clustering. Though [56] combines syntactic analysis with a clustering semantic approach, the method is WSDL based. Also, it clusters services into a known classification. The method [57] also clusters services into known classification. As in [55], the accuracy of [56-57] is affected by the value of $K$.

The method presented in [58] focuses on QoS aspects of services rather than clustering. The method [59] provides mechanism for labeling clusters and algorithm for discovering services using clusters. But it presents only theoretical concepts and does not address evaluation of accuracy of clustering.

Another method [29] is found to be attractive as it uses semantic similarity among services for clustering. But it handles only the core part of clustering i.e. way of computing similarity and clustering services (using Ant’s model). This work does not handle either how to label clusters or how to discover services using clusters.

In nutshell, the existing methods either are WSDL based in nature or it classifies services into known classification. Discovery based on WSDL are syntactic as WSDL elements have no explicit information about the intended semantics of services. This limits the accuracy of discovery. For example, a syntactic based method cannot detect *zip* and *pin* as same. Even some useful services are available to users the method may fail to detect such services. Further, syntactic based methods limits discovery to manual
style. Insufficient retrieval accuracy and manual involvement during discovery restricts the use of WSDL based methods to service composition. In service composition, several services from different domains are combined in specific pattern to achieve a business requirement. Automation in discovery and high retrieval accuracy crucial needs for service composition, WSDL based methods cannot be used for composition based applications.

Some of the existing methods classify services into known classification. In these methods the number of categories/partitions (K) to be produced is given as input to clustering algorithm. In these approaches the accuracy of clustering solution is influenced by the value of K whereas the value of K cannot be guessed always correctly due to the steady growth of services. Either big or small value of K alters the accuracy of clustering solution produced.

2.6.2 Other Open Issues while Clustering Services

When clustering techniques are used for service discovery certain aspects become important due to their practical implications. One such requirement is similarity needs of client applications. It is desirable to have clusters which assure to provide some preferred level of similarity among services as demanded by clients. In K-means algorithm, there is no provision to specify the required level of similarity among the services. In contrast to the approaches [55-57], the method [54] uses hierarchical clustering algorithm with single linkage which does not require the value of K as input. But this method does not focus on how to incorporate the similarity demands of clients while clustering services. Also, this method does not focus on two more crucial factors, namely, how to choose the method of computing inter-cluster distance while clustering services and how to choose value for threshold inter-cluster distance while specifying stopping criterion during clustering. Moreover existing approaches give less focus to tasks such as labeling or representing clusters with some common representation and discovering services using clusters.

Further, if the purpose of clustering services is to eliminate irrelevancy of a given query then the characteristic or feature by which the services are clustered gains importance. For efficient removal of irrelevancy, it is wiser to cluster services based on the similarity among outputs of services rather than based on a combination of input similarity, description similarity and output similarity among services. When services are clustered based on a combination of input similarity, output similarity and
description similarity as in [54], [55] and [29] then within a single cluster itself, it is more likely to have services which might get clustered even just by one component say description similarity or input similarity or output similarity. But clustering services in this manner does not lead to efficient removal of irrelevant services. Because when we find the relevant cluster of a query, there is no guarantee that all services present in the relevant cluster have similar outputs as the query. This implies that the semantic matching will be employed even to services which have different outputs with respect to query. According to our idea, during service discovery, when a query is submitted for finding matched services, a first level relevancy check that whether an advertised service is capable of producing the required outputs will be done. After confirming that a service could produce the outputs required by client, the input parameters and descriptions of services can be considered for further matching at a second level to decide whether the service is a matched service or not.

2.6.3 Limitations of Existing Indexing based Approaches

Though the approaches [60-62] employ index technique to identify candidate services, the main focus of the methods lies in discovering different composition of services which can fulfill the request when no atomic service is available to serve the request. To find matched services of a query these methods have to match the inputs of each candidate service with that of query. This requires semantic reasoning during querying which is time consuming. In [63] similarity table is used in which each ontology concept is mapped to its related ontology concepts which have similarity higher than a minimum. In practice, specifying a minimum similarity as numeric values while storing related concepts in similarity data table is difficult. Also, in this method there is no inner split of matched services; but a split in matched services will help service clients to pick up the most desirable match. The method [28] is found to be interesting as it prevents invoking semantic reasoning during querying completely with the help of two indices one for inputs and the other for outputs. But this method uses the traditional levels of DoM which are insufficient to meet disparate similarity demands of client applications.

2.6.4 Limitations of Existing Methods that Handle Similarity Demands

Existing methods of matching similarity such as [19], [22], [36], [64-66] detect two concepts as similar only when they are in the same branch of the ontology tree. But in reality, two concepts may have some similarity even if they are in different branches of
ontology tree. For example, two concepts in different branches of ontology tree are semantically related by common parent/sibling or grandparent or common children. Such semantic relations should be considered

2.6.5 Open Issues in Service Discovery using WordNet

Comparing the efficiency of WordNet (generic approach) based similarity computation against similarity computation based on service specific ontologies is an important issue and existing literature such as [73], [74], [52] do not address this issue. If the accuracy of computing similarity using WordNet is comparable to that of computing similarity using specific service ontologies, then based on application needs WordNet based computation can replace semantic discovery using specific domain ontologies.

2.6.6 Limitations of Existing Local Selection Methods

While decomposing global constraints, the methods such as [31], [86-91] assign a particular quality level which has the highest ‘benefit’ as local constraint for a task. But the way of computing benefit influences the method to ignore services with high utility defined by user. The methods such as [31], [91] can handle only sequential workflows whereas common business workflows contain execution patterns such as AND, OR and Loop. Though the methods [92] provide a formal way (computation of expected values of QoS attributes) to compute QoS attributes an OR execution pattern, there is no guarantee that the execution of all alternate paths of OR pattern will succeed when each path is given a chance for execution. Some methods such as [93] formulate the local service selection as an LP model for finding optimal services. Theoretically, an LP model takes a user defined utility function, a set of decision variables and a set of linear constraints as inputs. The model optimizes the utility function by optimizing the decision variables subject to the given constraints. Ultimately, the model produces optimal values for decision variables and utility function. But for service selection computation of optimal values alone is not sufficient; the identity of services having optimal values has to be found out. The LP model computes optimal values for decision variables and utility function but it never helps in finding the identity of optimal service (which is a mandatory prerequisite for composition). Though we can find the optimal values using LP models, it is not certain that a service with optimal value should always be available in a service repository. Another method presented in [94] finds the best candidate services for service composition by finding ‘skyline services’ for each class using dominance relationship. Though the work reduces the time complexity by
carrying out the detection of skyline services as an offline job, its performance is
affected by the number of constraints. Further, an efficient implementation of service
selection in satisfying user requirements in many real time applications such as e-
health, e-tourism, etc. is an urgent need.

2.7 Summary

In this chapter research works related to the theme of this thesis are reviewed. Various
optimization approaches to semantic service discovery are broadly classified into two
major categories, namely, optimization by efficient matching and optimization by
efficient reasoning. Optimization by efficient matching is further divided into various
sub categories, namely, optimization by clustering, optimization by indexing,
optimization by caching, optimization by hybrid methods, optimization by efficient
data structures and optimization by efficient matching algorithms. Of various
approaches clustering is found to be more attractive as it provides characterization of
similar groups of services and clustering of services serves as a base for further analysis
and service mining. Indexing of services provides a quick discovery by a mere lookup
and retrieval. Specific review of existing works related to clustering and indexing
techniques based service discovery has been done. Shortcomings of existing
approaches and open issues have been discussed. Problems in query construction while
using service specific ontologies for discovery have been analyzed. As an alternate to
the problems of service specific ontology based services research works related to
WordNet, generic ontology have been studied. Research works related to optimization
of performance of QoS based service selection have been reviewed. Works contributed
to global planning and local selection methods have been analyzed. Ultimately, the
gaps or limitations of existing approaches to optimize the performance of discovery and
selection, other open issues related to discovery and selection, methods to tackle
similarity demands and issues in constructing queries are analyzed towards defining the
problem statement of this thesis.