CHAPTER 2
LITERATURE SURVEY

2.1 NEED FOR SCHEMA MATCHING

Schema matching is finding semantic correspondences between elements of two schemas. e.g., database schemas, ontologies, and XML message formats. It is used in different database applications, such as integration of web data sources, data warehouse loading and XML message mapping. Now a day, schema matching is manual, time consuming and tedious. Automatic Schema matching technique is required to reduce the number of manual effort. This chapter explains the existing approaches, prototypes for schema matching and features and applicability using a previously proposed taxonomy. Further, this chapter discusses taxonomy of the schema matching technique, genetic algorithm based approaches, clustering techniques and their detailed classifications, and various literatures related to the schema matching, GA based approaches and clustering techniques.

2.2 TAXONOMY OF SCHEMA MATCHING APPROACHES

Match approaches can be differentiated from each other by using in three areas, i.e., the input information, the way the input information is processed, and the characteristics of the output match result. Figure 2.1, explains different matching approaches (Xiao Long Sun & Ellen Rose 2003):

- Schema vs. instance: Match approaches can consider schema-level information, i.e., metadata, such as element names, data
types, and structural properties, or also instance data, i.e., data contents.

- **Element vs. Structure**: Attributes, or combinations of elements that appear together in a structure can be compared and matched by the match operation.

- **Language vs. Constraint**: Linguistic approach (e.g., comparing names and textual descriptions of schema elements) or a constraint-based approach (e.g., data types, uniqueness, keys, etc. are used for considering constraints that are defined on elements) can be used by the matcher.

- **No-reuse vs. reuse**: Instead of relying only on the input schemas and instance data, the matchers also reuse information, such as dictionaries, global schemas, previous matching decisions, and user input from auxiliary sources.

- **Hybrid vs. Composite**: A matcher in turn may be a combination of several individual approaches, for better applicability and accuracy. This can be done either by combining the match results produced by a single approach within a composite or in a fixed way within a hybrid matcher.

- **Match cardinality**: One or more elements of the first schema may be related with one or more elements of the second one in a match result, resulting in different cardinalities (e.g., 1:1, n:1, 1:n, n:m). This match relationships may in turn be represented as a single or multiple correspondences.
2.2.1 Schema-based Matching

Schema-based approaches based on schema information. It depends on schema language, different properties of schema elements and relationships between them. Linguistic and constraint-based matchers are element-level approaches, which compare schema elements to determine their correspondences. Then structure-level methods are discussed which exploits element relationships to consider multiple elements at the same time.

- Linguistic Approaches

Text-based properties of schema elements are used in Linguistic Approaches, such as name and description. Name similarity can be estimated syntactically by relating the name strings or semantically by comparing their values.

Syntactical name matching calculates the similarity between two names based on linking the name strings. It is based on to determine equality of names, i.e., exact string matching. It is sufficient to identify the matching
elements in XML schemas. (Alergawy et al 2008). In other cases, so-called approximate string matching algorithms are powerful to identify similarity between a name and its different abbreviations, e.g., Customer - Cust. This algorithm is also used in text retrieval and automatic spelling correction, record linkage, and sequence alignment in molecular biology. Some of the algorithm like the Edit Distance, N-Gram, and Sound Ex algorithms are already been adopted for schema matching.

Semantic name matching estimates the similarity between names based on the following characteristics like; synonymy, hyponymy and hyponymy. This approach requires the use dictionaries, thesauri, ontologies in which the semantic relationships are captured. WordNet3 can be used for this purpose. Still, element names do not give sufficient context information to narrow down their meaning. Although creating such dictionaries may acquire substantial manual effort, it is worth the investment for supporting many match tasks in the same domain or for dealing with schemas with relatively flat structure, where dictionaries provide the most valuable matching hints.

- Constraint-based Approaches

Schemas constraints are used to declare data types, allowable values, value ranges, uniqueness, optionality and cardinality for single elements. By using these constraints in both input schemas are used to determine element similarity. For example, similarity can be based on the equivalence of data types and key characteristics like unique and primary.

Constraint-based matching alone often leads to imperfect n:m correspondences, as there are typically several elements in a schema with compatible constraints. For example, only considering data types, all elements of the same data type or similar data types would be regarded as similar. Nevertheless, constraints still represent helpful means to restrict the search
space for match candidates and can be combined with other approaches to improve accuracy.

### 2.2.2 Structure-level Approaches

Structure-level approaches based on relationships between elements and match combinations of instances that appear together in a structure. Different types of relationships may be available based on the modeling capabilities of the schema language, such as containment relationships, isa/part-of relationships, or referential constraints. Elements of schema elements and their relationships are denoted in a graph so that different kinds of structurally related elements can be identified for matching. Two different ways are needed for exploiting schema structure, namely, to determine element neighborhood for similarity calculation and to restrict the match scope in iterative matching, which are discussed in the following.

The common use of schema structure is to consider the neighborhood of elements to estimate their similarity. Many schema structures are based on hierarchical tree structure. To estimate the similarity between two elements, it is possible to compare different kinds of their neighbor elements. In an undirected graph representation of schemas, any adjacent elements in the graph can be considered as neighbor elements. The similarities between the neighbors elements can be first computed using an element-level matcher. DIKE and SIMILARITYFLOODING are used to compute neighborhood similarities using fix-point computation. When the fix point is reached, the similarity values stabilize in the schemas and are taken as structural similarity of schema elements.
2.2.3 Instance-based Matching

Instance-based matching based on correspondence of schema elements. It can be used for semi structured data and limited data. Schema extraction techniques can construct a schema when no schema is available. Instance-based approach depends on the availability of instance data. Accuracy of this matching depends much on the quality of the instance data (Massmann & Rahm 2008) possibly incurring additional effort for data cleaning. The quantity of data to be examined in instance-based approaches is greater than that in schema-based approaches. Therefore, execution time becomes a more critical issue, especially for interactive use.

Typically, elements at the finest level of granularity in a schema, such as attributes in relational schemas, XML schemas, are reflected, as these directly accommodate instance values in instance of database. The term attribute is used to indicate such atomic schema elements. While most of the methods discussed for schema-based matching can also be working to characterize instance data, several have been developed for instance-based matching. Instance-based matching at element and structure level discussed below. At the element level, instances of single atomic elements are considered, while the structure level examines instances of multiple attributes at the same time.

- Element-level Approaches

Information retrieval techniques can be used to get linguistic characteristics based on the relative frequencies of word and combination of words in text based attributes. By looking for happenings of special words and abbreviations, it may also be able to identify instances of geographical names and addresses. The similarity between the extracted themes and
keywords can be obtained by using approximate string matching techniques like name matching.

Determining and comparing constraint-based characteristics in numerical attributes, such as data length, data type, numerical value ranges and averages, standard deviation, key and uniqueness constraints, frequencies of characters, etc. is also possible. Also, the characteristics represent the real metadata that may deviate from the specification in the schema. An attribute that is declared of type string for instance may be used to store number values in reality. The schema-based approaches can be improved to get best match predictions.

- **Structure-level Approaches**

  Structure-level matching uses multiple attributes at the same time and requires characterizing the content of those columns. Hence, the main problem is the explosion of the combinations of number of possible attributes for which the instances would have to be estimated. It is used to identify promising combinations of attributes, such as the columns of single tables in a relational schema in case of a schema structure. Attribute combinations may be selected according to the similarity between their properties, such as equality or compatibility of data types, in case if no schema structure is available and input schemas are simple collections of attributes.

2.2.4 **Reuse-oriented Matching**

All forms of using auxiliary information is there in reuse-oriented matching i.e., in addition to schema and instance information given in the input schemas/databases, which enhances the match process. Difference between schema-based and mapping-based approaches is possible, that supports previously determined correspondences and the reuse of common
schema components, respectively. Both kinds of reuse may in turn consider single elements or structures, like schema and instance-based matching, i.e., combinations of elements.

- **Schema-based Reuse**

  Commonly used names may be defined and maintained to reduce schema heterogeneity, in a global vocabulary or namespace. The names can be used to declare elements (e.g., by specifying their names, data types, value ranges, etc.) to refer to the same real-world concepts in different schemas. Namespaces are normally supported by several XML-based schema languages, including XSD, and can also be stored in specific dictionaries. A general approach is reusing not only globally defined names but also the entire schema fragments, which includes certain features like data types, keys, and constraints. This is especially rewarding for frequently used entities, in a global schema library, such as address, customer, employee, purchase order, and invoice, which can be defined and maintained properly. These libraries can be accessed to encourage the reuse of predefined names and schema fragments. Using the same namespaces or schema libraries, similar elements can be quickly identified using exact name matching techniques in matching schemas.

  On the other hand, it is known that different organizations can agree on such standardized names and structures, e.g., sharing the same vocabulary or namespace, to construct their own schemas. Particularly, an existing work applies machine learning techniques to an amount of schemas to cluster similar elements and to derive characteristics concerning the co-occurrence and ordering of the elements of their neighbors. These characteristics are then used to identify similar elements in new schemas as constraints. While other considers, highly overlapping schemas from a limited
domain at the same time. A statistical model is developed assuming the same semantics for attributes with equal names, i.e., from the same vocabulary, it develops a model that is based on the co-occurrence of the attributes in single schemas to predict synonym attributes between multiple schemas.

- **Mapping-based Reuse**

Mapping-based reuse which are discussed already can be implemented by exploiting previously determined similarity relationships or correspondences, such as dictionaries, thesauri, domain-specific synonym tables, and user-provided matches or mismatches encoded in different kinds of auxiliary sources. Schema-based approaches for example, look up in such auxiliary sources and check whether the element names are meanings or data types that are compatible to decide about similarity of elements. User-specified correspondences are employed by instance-based approaches to learn instance characteristics for single schema elements. Both examples represent a simple form of reuse which utilizes confirmed correspondences at the level of single schema elements.

### 2.2.5 Combination Approaches

The match approach applicability depends on the kind of on it exploits. If a single approach is used by the matcher then it is unlikely to achieve good match quality for a huge variety of schemas and domains (Amshakala & Nedunchezhian 2013). Therefore, there is a combination of multiple approaches in the current schema matching systems. There are two different ways to do this: hybrid and composite. A hybrid matcher uses multiple approaches and integrates in a fixed way, where as a composite matcher combines the results of executed matchers that are independent, which may be either hybrid or composite matchers.
• Hybrid Combination

With a hybrid matcher, the single match criteria or information sources are hard-wired, which are mostly checked according to some specific heuristics in a particular order. For example, elements with data type similarity are first identified, before they are checked for the similarity of name. Hence knowing this heuristics well in advance, it is more easy to optimize the execution within the corresponding hybrid matcher. The result of this is that, a hybrid matcher can offer time performance better than the separate execution of multiple matchers by reducing the number of passes to be performed over the input schemas. Additionally, because poor match candidates qualifying only one of several criteria can be singled out early, effectiveness may be improved.

Prototypes that are previously used follow this approach to combine different algorithms in a specific match solution. This allows developing and implementing an optimized solution to exploit some characteristics explicitly, for example, that are supported by the language or the domain of the schemas. On the other side, extending and improving a hybrid matcher is difficult, for example, to support a new schema type, new match criteria, or new information sources. While hybrid approaches are unlikely successful for a huge variety of match tasks and domains, to achieve high flexibility and adaptability they can be combined using a composite matcher which is discussed below.

• Composite Combination

The results of independently executed matchers are combined by a composite matcher, who may in turn be composite or hybrid matchers. Based on the application domain or schema language, it allows us to select from a repertoire of matchers for combination, offering a good flexibility compared to the hybrid approach. Different matchers for example can be employed to
match structured or semi-structured schemas (McCann et al 2008). Moreover, for combining the individual match results there are different possibilities. One could use machine learning for example to combine the independent matchers, as it is done in the LSD system to combine instance-based matchers, both schema- and instance-based approaches are combined. The selection of the matchers and the strategy for combining them can also be set to default for automatic execution.

As similarity computation or match prediction (Fagin et al 2011) is done in the individual matchers, the composite approach gives little space for optimizing its execution performance. For instance, it is difficult to share the computed name similarities between them if name similarity is utilized by several hybrid matchers and if they are combined within a composite matcher. Due to its flexibility and extensibility, the composite approach represents the method for implementing a generic match solution which supports different application domains and schema types. To combine match algorithms, among others it is observed that only few prototypes currently follow this approach.

2.3 REVIEW OF STATE-OF-THE-ART SCHEMA MATCHING SYSTEMS

A comparison of tools and prototype implementations for schema matching approaches is presented in this section which is based on the type of input it supports, schema matching search space covered, application domain and level of automation. Some of the notable prototypes discussions along with the comparison of their schema matching approaches are as follows,

2.3.1 COMA

A schema matching system called COMA (Combination of Matching algorithms) allows combining matchers and evaluates their schema matching performance. COMA systems have a wide array of individual
matchers. Reusing of match results of the previous match operations is permitted. It is a generic match system where it can be potentially used for many applications and supports XMLs and relational schemas as input types.

COMA presents a set of individual matchers who can perform name-based comparison of schema elements. Different string comparison technique is used by each individual matcher like edit distance, N-gram or Affix. It also presents individual matchers for data type comparison, for the use of meanings in string comparison, and also for user feedback. COMA allows combining these individual matchers to form a hybrid schema matcher or a composite schema matcher. Multiple individual matchers are combined by hybrid schema in a single matcher to generate one match result; while a composite schema matcher combines the output of multiple individual matchers. Composite schema matcher also combines a hybrid matcher with individual matchers. At element-level and structure-level COMA system performs schema matching.

COMA++ is the next version of original COMA system; which supports a graphical user interface and for ontologies (Aumueller et al 2005). COMA++ system is made to solve huge real world problems. A divide and conquer approach which is based on fragment based matching is used to solve the huge matching problems and it also gives an emphasis on reusing the already performed matching.

2.3.2 CUPID

'CUPID' by using a hybrid matcher combines linguistic level matching and structure matching. To improve the accuracy of schema matching 'CUPID' makes use of generic as well as domain specific thesauri. Linguistic schema matching is performed in three steps, normalization, categorization and then actual linguistic comparison by making use of the
thesauri. Along with structure-level schema matching data type matching is done. Mapping results of 'CUPID' can be a one:one mapping or one:n mapping between schema elements (Jayant Madhavan et al 2001).

'CUPID' systems pictures a schema as graph, with elements represented as nodes. Nodes can have edges and is related to the connected element. Every relationship in graph model is represented by a distinct node name. The 'CUPID' extends its tree matching algorithm and performs schema matching by using this graph representation of the schema. This system is capable of performing relationship mapping of references by representing referential constraint as a distinct element in the graph and then to get the mappings it performs tree comparison.

2.3.3 LSD

LSD (Learning Source Descriptions) system solves schema integration problem by using a machine learning approach. LSD system performs element level matching and instance level matching using the base learners which are a set of matchers. It creates schematic mapping using a two phase semi-automatic process. Base learning is first trained by using user supplied mapping between the source schema and the mediated global schema. Matching rules during the learning phase is discovered by learners which are done by analyzing the data instance in the source schema. These matching rules and patterns that are discovered are then used in the matching phase to create mappings between new source schema and the internal mediated global schema. LSD generates one: one mapping between elements of the two different schemas. The schema information can be exploited by the learner to perform XML element matching or perform matching based on data values in the schema elements. For the improvement of more matching accuracy LSD system makes use of user supplied domain constraints.
2.3.4 Other Schema Matching Approaches

SemInt prototype performs the integration called schema integration. It utilizes both schema information and data contents that are fetched from catalog of a relational DBMS of an element to perform schema matching. SemInt perform matching by a classifier method that trains neural networks to recognize the different categories of elements and generates a one:one mapping.

The TranScm prototype drives data translation by using schema matching. This schema is translated and represented as an internal graph. Multiple handcrafted matching rules are applied at each node in order. Using the top-down approach, matching is completed with the rules at higher-level nodes that requires the matching of descendants. This top-down approach performs good only when the highest level structures of the two schemas are more similar. It represents an element-level and schema-based matcher.

By exploiting the principle that the similarity of schema elements depends on the similarity of elements in their vicinity the DIKE system integrates multiple ER schemas. The relevance of elements is inversely proportional to their distance from the elements being compared, so the nearby elements influence a match that is more than one that is farther away. Linguistic matching is predicted on manual inputs.

Another important notable prototype is ARTEMIS, which is a schema integration system. It calculates element-level and structure-level schema match affinity using a hybrid model and then uses clustering technique which clusters the attributes based on the match affinity. ARTEMIS makes use of generic and domain specific thesauri. It allows user to assign
adjustable weights to the match operations, and then these operations are used to calculate the matches.

Protoplasm (Bernstein et al 2004) target provides a flexible and a customizable infrastructure for combining different match algorithms. Currently CUPID (Madhavan et al 2001) implementation and Similarity Flooding (SF) (Melnik et al 2002) algorithms are being used as the base matchers. BizTalk Mapper (Bernstein et al 2006) proposed and demonstrated a graphical interface for this which is based on the HCI research presented in (George Robertson 2005) and is heavily dependent on microsoft technologies.

The CLIO (Hernandez et al 2002) which has been developed at IBM is a complete schema mapping and management system. It provides matching for XML and SQL schemas (Object Relational databases converted into relational with the help of a wrapper function) and has a comprehensive GUI. It combines approximate string matcher for element names and Naïve Bayes learning algorithm for exploiting instance data, which is a hybrid approach.

GLUE (Doan et al 2003) finds ontology/ taxonomy mapping using machine learning techniques which is an extended version of LSD(Doan et al 2001). The system is fed in with a set of data instances along with the source and target taxonomies. GLUE classifies and associates these classes of instances from source to target taxonomies and vice versa.

MOMIS (Beneventano et al 2001) is a heterogeneous database mediator. ARTEMIS which is one of its components, is the schema integration tool which employs schema matching to integrate multiple source schemas into a virtual global schema for mediation purposes. The tool operates on hybrid relational-OO model which first calculates the elements
similarity based on name and data type, thus all the possible target elements are acquired.

A schema integration tool called Wise-Integrator (He et al 2004) uses schema matching to find correspondences among web search forms so that they can be unified under an integrated interface. Each input form is compared incrementally against a local interface which has been selected. Without a match candidate in the local interface, the attributes are added to it. Wise-Integrator allows several algorithms to compute attribute similarity.

The objective of DCM framework (Dual Correlation Mining) (He et al 2004) is similar to Wise-Integrator, which on the problem of obtaining an integrated interface for a set of web search forms holistically. It is observed by different authors, that the aggregate vocabulary of schemas in a (restricted) domain, such as a book, tends to converge at a small number of unique concepts, like author, subject, title, and ISBN, although for the same concept different interfaces may use different names.

Another implementation of holistic schema matching, for a given set of web query interface schemas is PSM (Parallel Schema Matching) (Su et al 2006). The objectives are similar to DCM algorithm, but PSM improves on DCM on two things, first for two elements to identify synonyms, the DCM negative correlation computation may give high score for rare elements but PSM does not. Secondly, in DCM the time complexity is, in exponential with respect to the number of elements where as for PSM it is polynomial.

ONTOBUILDER (Roitman & Gal 2006) is a generic multipurpose ontology tool, which can be used for authoring, and matching RDF based
ontologies. Its interface also supports the process of matching web search forms for generating an integrated form. OntoBuilder generates dictionary of terms by extracting labels and field names from web forms, and then it recognizes unique relationships among terms, and utilize them in its matching algorithms.

Auto Mapping Core (AMC) is a generic framework that supports a customizable matching process. The matching process can be defined, executed, debugged, and visualized with highly flexible components. Existing matching tools can be plugged into AMC for a uniform evaluation and reuse. Especially, it supports various aggregation operators (e.g. sum, max, min) to combine the matching results of individual matchers.

Yet Another Matcher (YAM) is a generic prototype that supports both matching tasks and post match effort. One of its key features is the ability to dynamically combine similarity measures according to machine learning techniques, with the benefit of automatically tuning the thresholds for each measure. Its extended version, YAM++, also supports ontology matching using machine learning approach.

Harmony is the matching tool inside the OpenII framework. It provides a graphical user interface and supports many well-known matching techniques. Especially, it supports composite matching where different confidence values (proposed by individual matchers) are aggregated at schema-level. Harmony is known to be able to match large schemas having about a thousand of attributes.
Table 2.1 Schema Matching Tools and Prototypes Comparison-General

<table>
<thead>
<tr>
<th>Tool</th>
<th>GUI</th>
<th>Approach</th>
<th>Card</th>
<th>Internal Rep</th>
<th>Research Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bellflower</td>
<td>No</td>
<td>Hybrid</td>
<td>1:1</td>
<td>Directed Graph</td>
<td>Schema Matching</td>
</tr>
<tr>
<td>Clio</td>
<td>Yes</td>
<td>Hybrid</td>
<td>1:1</td>
<td>Rel. Model, Directed Graph</td>
<td>Schema Matching, Mapping Evolution</td>
</tr>
<tr>
<td>COMA++</td>
<td>Yes</td>
<td>Composite</td>
<td>1:1</td>
<td>Directed Graph</td>
<td>Schema Matching</td>
</tr>
<tr>
<td>DCM</td>
<td>No</td>
<td>Hybrid</td>
<td>N:m</td>
<td>-</td>
<td>Schema Integration</td>
</tr>
<tr>
<td>GLUE</td>
<td>No</td>
<td>Composite</td>
<td>N:m</td>
<td>Attribute based</td>
<td>Data Integration</td>
</tr>
<tr>
<td>MOMIS</td>
<td>Yes</td>
<td>Hybrid</td>
<td>N:m</td>
<td>Directed Graph</td>
<td>Schema Integration</td>
</tr>
<tr>
<td>ONTO BUILDER</td>
<td>Yes</td>
<td>Hybrid</td>
<td>1:1,1:N</td>
<td>Graph</td>
<td>Create/match ontologies</td>
</tr>
<tr>
<td>PORSCHE</td>
<td>No</td>
<td>Hybrid</td>
<td>1:1,1:N</td>
<td>Tree</td>
<td>Schema Integration and Mediation</td>
</tr>
<tr>
<td>Protoplasm</td>
<td>Yes</td>
<td>Hybrid</td>
<td>1:1</td>
<td>Graph</td>
<td>Schema Matching</td>
</tr>
<tr>
<td>PSM</td>
<td>No</td>
<td>Hybrid</td>
<td>N:m</td>
<td>-</td>
<td>Schema Integration</td>
</tr>
<tr>
<td>QOM</td>
<td>No</td>
<td>Hybrid</td>
<td>1:1</td>
<td>Tree</td>
<td>Ontology Alignment</td>
</tr>
<tr>
<td>SCIA</td>
<td>Yes</td>
<td>Hybrid</td>
<td>N:m</td>
<td>Tree, Graph</td>
<td>Data Integration</td>
</tr>
<tr>
<td>Wise integrator</td>
<td>yes</td>
<td>Hybrid</td>
<td>1:1</td>
<td>Attribute based</td>
<td>Web Search Form Integration</td>
</tr>
</tbody>
</table>

Table 2.2 Tool Catalogue of Schema Matching Research Prototypes

<table>
<thead>
<tr>
<th>Matching Tool</th>
<th>Year of Introduction</th>
<th>Supported Format</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUPID</td>
<td>2001</td>
<td>XSD</td>
<td>N/A</td>
</tr>
<tr>
<td>Onto Builder</td>
<td>2004</td>
<td>XML</td>
<td>CLI,GUI</td>
</tr>
<tr>
<td>Coma</td>
<td>2002-2012</td>
<td>XSD,OWL,Relational</td>
<td>GUI,JAVA, API</td>
</tr>
<tr>
<td>Clio</td>
<td>2009</td>
<td>XML,Relational</td>
<td>Commercial Tool of IBM</td>
</tr>
<tr>
<td>YAM</td>
<td>2009</td>
<td>XSD,OWL</td>
<td>GUI,CLI</td>
</tr>
<tr>
<td>Harmony(Open II)</td>
<td>2010</td>
<td>XSD, OWL, Relational</td>
<td>GUI,open-source</td>
</tr>
<tr>
<td>AMC</td>
<td>2011</td>
<td>XSD</td>
<td>Commercial, GUI</td>
</tr>
<tr>
<td>Tool</td>
<td>Input</td>
<td>Output</td>
<td>Match</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bellflower</td>
<td>XSD</td>
<td>Schema Matches</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIO</td>
<td>SQL,XSD</td>
<td>Mappings(Query)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMA++</td>
<td>XSD,XDR, RDF, OWL</td>
<td>Mappings, Merged Schema</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCM</td>
<td>Web Query Interface</td>
<td>Mappings between all input Schemas</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLUE</td>
<td>DTD, SQL, Taxonomy</td>
<td>Mappings, IMap functions</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOMIS</td>
<td>Rel, OO datamodel</td>
<td>Global View</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONTO Builder</td>
<td>RDF</td>
<td>Mediated Ontology</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORSCHE</td>
<td>XSD Instance</td>
<td>Mediated Schema</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protoplasm</td>
<td>XDR, SQL, RDF</td>
<td>Mappings</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSM</td>
<td>Web Query Interface</td>
<td>Mappings between all input Schemas</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QOM</td>
<td>RDF(S)</td>
<td>Mappings</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCIA</td>
<td>Rel , DTD, XSD, OWL</td>
<td>Mappings(Query)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WISE</td>
<td>Web Query Interface</td>
<td>Integrated Schema</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrator</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.4 VARIOUS EXISTING WORKS RELATED TO SCHEMA MATCHING APPROACHES

Several techniques are proposed by various authors for schema matching and a few of them are explained below:

Wang et al (2004), by using domain-specific query probing proposed a unified solution based on an instance-based schema matching technique, used for Web databases. In certain case, a Web database contains two dissimilar schemas, first is the interface schema (IS) and second is the result schema (RS). The users can query using data attributes which is included in the IS of an individual Web database, while the RS contains data attributes in which the query results that users receive is arranged by these data attributes. This approach is based on three observations about Web databases:

1. Queries often cause search failure that returns no results.
2. There is reappearance in the returned result pages for the keywords of proper queries that return results.
3. For Web databases, there is a global schema (GS) of the similar domain [He & Chang 2003], where the global schema consists of the representative attributes of the data objects in a particular domain.

Domain-specific Query Probing is not fully automatic, according to author for global schema generation techniques. Additionally, for improving this many authors have suggested to implement with label-based approaches to construct a more robust matching system.

Hess et al (2006) proposed G-Match algorithm used for geographic ontology matching. G-Match takes two inputs which are different geographic
ontologies, and measures the similarities of their concepts by considering the category and attribute names (string similarity). Here WordNet (WordNet 2006) is used to provide the algorithm with synonyms and assumes that the syntactical and structural similarity implies semantic proximity, which is often not warranted. Natural language dictionaries may be useful, and even multi-language dictionaries are also useful to deal with schemas which uses terms in different languages. To deal with organizational standards, such as abbreviations for schema element names, domain or enterprise – specific dictionaries may sometimes be useful.

Brauner (2007), based on the use of a small set of typical instances, presented a semantic approach for matching export schemas of geographical database Web services. The paper demonstrates with a wide experiment that uses two gazetteers, Geonames and the ADL gazetteer. The system contains a limitation of instance-based schema matching process, therefore it is required to improve the re-occurrence detection method, which executes a validation step to define a threshold to the proportion between the re-occurrence values and a proof of concept is also required.

Brauner (2008), proposed an instance-based approach for matching export schemas of databases available through Web services. This technique describes to construct the mediated schema and also to discover schema mappings, based on matching query results. The approach is illustrated with two case studies from rather different application domains. To validate this approach, an experiment using bookstore databases and gazetteers is discussed. In order to enhance more investigation of the alignment of exportable schemas with the mediated schema and its implications, as the association of the mediated schema with a global instance set required to derived from existent sources.
E Toch (2011) deduced the similarity among Web services as a basic construct for service matching and their composition. On the other side, there is too few indication of how humans observe similarity among services, a more important key for designing usable and real time service comparison and composition algorithms. Also a similarity model is defined, on the basis of experimental investigation and demonstrates that the similarity model, stated by a distance metric, is complete and that it accurately predicts the views of service in human. Finally, it refers to an application of a Web service discovery technique that utilizes this model. It demonstrates that humans take a more appropriate technique when evaluating similarity than forecast by logic-based methodologies. This difference may provide some explanation of the slow adoption rate of automatic service composition paradigms. Given similarity computation demonstrate some necessary properties, like explainability, as every similarity-based judgment can be analyzed and explained to the user, based on the underlying ontology. To overcome the issue of this simple search scheme, variety of methods available some of them, NLP techniques used to improve search results so that they would be semantically more relevant to the submitted query. The main difference between proposed work and recent work is to provide a generalized framework for Schema Matching for Heterogeneous Web databases. Moreover, the instance-based schema-matching method is infrequently used for schema matching in the Web database since it is hard to obtain instances from Web databases.

A two-step schema-matching method is suggested by Jaewoo Kang & Jeffrey Naughton (2003), which holds good, even when opaque column names and data values exist. Initially, by calculating the relationship between pair wise attributes in the tables, a dependency graph is generated and later by employing the mutual information found in them. Then in search of matching node pairs across the dependency graphs, a graph-matching algorithm is run. It is also revealed that like entropy-only matching the enhancement of single
column uninterpreted matching, is feasible by taking inter attribute correlations into consideration. Approximation algorithms are also explained for the matching problems and these algorithms find efficient implementation in the proposed work. In particular, the HC approach provides optimal solutions, making the graph-matching problems visible in the schema matching domain to be agreeable to HC.

Zohra Bellahsene & Fabien Duchateau (2008) has given some general idea on how to tune schema matchers. Initially, the schema matchers allows the users to configure thresholds which are low-level parameters. Later, filtering or selecting the output (the set of mappings) was allowed with better flexibility and the revealed mappings were based on parameter configuration. For a given matching situation the selection of suitable matcher is available at present and they reduce the work load of almost all low-level parameters by automatic tuning. To incorporate user preferences or input data parameters, attempts were made that exploit machine learning techniques for the utilization of schema instances or expert response. By these attempts the matching quality is enhanced. Accuracy and recall which are user preferences allow users to deal with post-match effort. More attention is paid for these qualities in situations where the dynamicity is too large to develop data sources and a high precision is desired. But in contrast, when the complete integration and manual verification of data sources are made, the recall can be encouraged. Even though a default configuration with a matcher is still recommended, a specific configuration of the schema matcher for a given matching problem has been approached. Depending on the earlier researches or features, the finest configuration is created by the tool that calculates several matching state characteristics.

Philip Bohannon et al (2006) has given a contextual schema matching, in which some of the observations of source and/or target tables are
assumed to be a portion of the schema matching process by the schema matching system. A group of appropriate conditions include numerous realistic examples that are simple and disjoint have been introduced. Only under these conditions the matches were found to be significant, and new algorithms have emerged to describe those conditions. The outcomes of an extensive experimental study on the performance and quality of the techniques have been elaborated by them, due to the uncertainty in observations. It was concluded that, in problems like horizontal partitioning of schemas and promotions of data values to attributes, the contextual matching is capable of providing additional automation of schema mapping. The proposed method has showed a way for the future works to be carried out both theoretically and practically. Specifically, where similar matches carry out both horizontal partitioning and attribute promotion, extra view conditions for the target schema becomes essential in situations. With increase in the number of observations automated interference of views become more challenging. Another problem associated with this is the demand for a new form of constraints to hold complex contextual conditions, attribute promotion and related semantic-association rules. The final issue is the communication between the constraints associated with the contextual schema mapping that requires the usage of a systematic method for not violating the target constraints.

Philip A. Bernstein et al (2004) has proposed a technique where, the system’s integration work frequently needs the construction of mappings between models like database schemas, interface definitions, message formats and user-interface forms. The schema mapping is similar to database design and semantics of schemas are thoroughly viewed for its design. Schema matching deals with the detection of elements of two given schemas that match up one another. There are only few systems available for schema matching to be practically used, despite the existence of several algorithms.
Authors have expressed their preliminary experience in constructing an adaptable schema matcher called Protoplasm. The schema matching algorithms are weak, that they require a customizable schema matcher to incorporate numerous uncomplicated customizable scripts. The challenge is to have a small number of such drafts that wrap the entire space of functional matchers and to provide a small set of easy customization options. The advantage of a schema matcher is that it can use the mappings that are created earlier. The reusable mapping is carried out among the sub-schemas of the schema matcher’s inputs. Finding reusable mappings and relating them to a certain mapping problem is often discouraged.

Jayant Madhavan et al (2002) has proposed a technique where in, two schemas serve as inputs and a mapping that detects matching elements in the two schemas is produced. Thus it forms a schema manipulation operation. Schema matching is the significant step employed in various applications like XML message mapping, data warehouse loading, and schema integration. The algorithms for generic schema matching that remains external to certain data model or application are being proposed in the paper. Initially, an arrangement of previous solutions that depicts the existence of a wide range of techniques is explained. Later, a novel algorithm called Cupid was proposed. This algorithm is capable of creating mappings among schema elements depending on their names, data types, constraints, and schema structure by making use of extensive techniques than previous methods. The integrated use of linguistic and structural matching, context-dependent matching of shared types and a bias toward leaf structure are a few creations where the schema content exist to a large extent.

A fundamental form of schema matching problem was revealed by Jaewoo Kang and Jeffrey Naughton (2008) where the schema elements are
mapped from one data storage area to the related elements of another information storage, provided that the column names and data values are opaque in nature. A two-step procedure is explained that does not involve the interpretation of data elements or schema elements. The un-interpreted matching technique was initiated that makes use of inter-attribute dependency affairs. It is also proved that the utilization of inter-attribute relations could enhance a single column un-interpreted matching like entropy-only matching and is found that the improvement was in the range between 9% to 31% based on the cardinality constraints of the mapping problem and the data sets used for testing. This has paved a huge way for future works. This work has utilized two simple distance metrics, namely, Euclidean and normal. But, with the usage of numerous complicated distance metrics, improved results can be made possible. The selection of metrics is only focused on searching an accurate subset of features from target schemas, which seems to have more attention. Creation of correct but computationally efficient approximation algorithm for the instances of the graph matching problem that resulted from their technique needs more concentration to the schema mapping problem. Using various un-interpreted methods is so significant in making assessments on additional dependency models. In the proposed work, much concentration is given to match flat tables. The future work includes the broadening of this technique to nested structures such as XML or object-oriented schemas.

Yu Tang et al (2004) discussed the similarity measure termed Earth Mover’s Distance (EMD), and hence received a lot of attention in the fields of multimedia and probabilistic databases. EMD provides better distinguish ability on multidimensional histograms between the objects approximated by the histograms (e.g., images), that is compared to classic measures like Euclidean distance. EMD has a high computational cost, inspite of its usefulness and therefore a number of effective filtering methods have been proposed, during similarity search, to reduce the pairs of histograms for which
the exact EMD has to be computed. However, still EMD calculations remains a bottleneck of the whole similarity search process in the refinement step. Hence, the authors focused on optimizing the refinement phase of EMD-based similarity search by,

(i) adapting an efficient min-cost flow algorithm (SIA) for EMD computation

(ii) proposing a dynamic distance bound, by which an EMD refinement can be used to terminate early

(iii) Proposing a dynamic refinement order reduces the amount of needless computations, for the candidates who paired with a concurrent EMD refinement strategy.

The authors studied the efficient evaluation of similarity queries using Earth Mover’s Distance (EMD). First, the authors showed how SIA has adapted, an algorithm originally proposed for spatial matching problems, to compute the EMD between two histograms efficiently. Then, the authors proposed a progressive refinement strategy, which updates a lower bound for EMD during its computation, in order to abandon early a partial EMD refinement, if the object cannot make it in the query result. Finally, the authors proposed a technique which concurrently handles the refinement of multiple candidates, by dynamically reordering them and computing upper bounds that help to tighten the pruning threshold early. The experiments showed that the methods were very effective in practice, decreasing the overall cost of EMD-based similarity queries by up to two orders of magnitude, compared to the state-of-the-art solution.

As per Philip A. Bernstein et al (2004) proposed a schema matching technique as a critical step in many applications, such as XML
message mapping, data warehouse loading, and schema integration. The authors have investigated algorithms for generic schema matching, outside of any particular data model or application. The authors proposed a new algorithm, CUPID that discovers mappings between schema elements based on their names, data types, constraints and schema structure which uses a broader set of techniques than past approaches. Some of the innovations involved were the integrated use of linguistic and structural matching, context-dependent matching of shared types, and a bias toward leaf structure where much of the schema content resides. Experimental results were presented that compares CUPID with two other schema matching systems.

Philip A. Bernstein et al (2004) discussed Schema matching which identifies elements of two given schemas that correspond to each other. Although there are many algorithms for schema matching, few algorithms have been written about building a system that can be used in practice. The authors described their initial experience building such a system, a customizable schema matcher called Protoplasm.

Erhard Rahm et al (2001) proposed a schema matching with its application in many database domains, such as data integration, E-business, data warehousing loading and semantic query processing. In particular, the author has distinguished clearly between schema and instance-level, element and structure-level, language and constraint-based matchers. Based on the classification, they review some previous matching implementations thereby indicating which part of the solution space they cover.

Jaewoo Kang et al (2008) discussed most precious solutions to the schema matching problem which rely in some fashion upon identifying "similar" column names in the schemas to be matched, or by recognizing common domains within the data stored in the schemas. Whereas each of these approaches is valuable in several cases, they are not infallible, inerrant
and there exist instances of the schema matching problem for which they do not even apply. These types of problem normally arise when the column names in the schemas and the data in the columns are "opaque" or very difficult to interpret.

The schema matching problem refers to the problem of mapping schema elements (for example, columns in a relational database schema) in one information repository to corresponding elements in a second information repository at the most basic level. Even though schema matching has always been a hectic and also an interesting aspect of information integration, the problem is worsened as the number of information sources to be integrated is increased, and therefore the number of integration problems is grown which must be solved. Such problems in schema matching arise both in “classical” scenarios such as company mergers and in “new” scenarios such as the integration of diverse sets of queryable information sources that are over the web.

Schema matching problem are too labor intensive to be scalable if using purely manual solutions; which as a result, has led to a great deal of research in the area of automated techniques that can speed this process by either automatically discovering good mappings, or by proposing similar matches that are later verified by some human expert. The authors have presented an automated technique that is designed to be of help in the particularly difficult cases in which the column names and data values are “opaque,” and/or other cases where the column names are opaque and the data values are taken from the same domain in multiple columns. This approach works within each schema by computing the “mutual information” between different pairs of columns, and then by using this characterization of pairs of columns in one schema to propose the matching of columns in the other schema. Consider a classical schema mapping problem that arises in a
corporate merger to clarify aims and provide some context. It is required to integrate the databases of the two companies to complete the merger. How can we determine which attributes in one company’s table should be mapped to which attributes in the other company’s table? Initially, a logical approach is used to compare attribute names across the table. Few of the attribute names for matching will be clear candidates, due to common names or common parts of names, which is an example of schema-based matching. Anyway for more number of columns, schema-based matching may not be effective, because different institutions uses different terms or encoding schemes for semantically identical attributes, or uses identical names for semantically different attributes.

The next logical approach when schema-based matching fails, is to look at the data values that is stored in the schemas. This approach is called instance based matching referring to the classification. Instance-based matching also will work in various cases. For example, if the approach is deciding whether to match Dept in one schema to either DeptName or DeptID in the other, by looking at the column instances it may easily find the mapping, because DeptName and DeptID are taken from different domains, e.g., names and alpha-numeric codes. But, however, instance-based matching is also not successful always.

When instance-based mapping fails, it is not often able to distinguish different columns over the same data domain and, similarly, it is not able to find matching columns of logically similar domains over different encodings. For example, EmployeeID and CustomerID columns in a table are not to be distinguished if both the columns are of numeric data types and the ranges of the IDs are identical. Similarly, if one company uses numeric values for the EmployeeID while the other company uses a formatted text for logically the same column, the permission to make digital or hard copies of all
or part of this work for personal or institutional use is granted for free
provided that, copies are not made for distribution for profit or commercial
advantage and that on the first page the copies bear this notice and the full
citation. To copy or republish, to post on servers or to redistribute to lists,
requires prior specific permission and/or a fee.

Hence the researcher has used the merits of the both schema based
and instance level schema matching approaches along with the Genetic
algorithm and hill climbing approaches to improve the automation of schema
matching.

Timo Mantere (2006) introduced a min-max genetic algorithm that
can be applied to the min-max problems. For simultaneous minimization and
maximization of the same object function during the same optimization run, a
min-max genetic algorithm was designed. Here, a method with multiple
sorting is applied for optimizing constrained functions. One end of the genetic
algorithm population minimizes the constraint violations, while the other end
maximizes the values of feasible solutions. According to the results this
method reaches the feasible area in a reasonably fast and consistent manner
and hence produces relatively good results. This paper is studied for the use of
min-max genetic algorithm (MMGA) being beneficial when optimizing
constrained functions. The MMGA means genetic algorithm where the elitism
is applied to both the ends of the population fitness scale. In other words the
survival to the next generation will be for the number of individuals with the
lowest fitness value and number of individuals with the highest fitness value.
 Practically, this means that it is simultaneously minimizing and maximizing
the same object function. In an elitist steady-state genetic algorithm (GA) the
optimization direction depends on the elitism. If for some individuals the
survival to the next generation is with the highest fitness values and it was
maximizing the target function and vice versa if the number of individuals
with lowest fitness values are minimizing the problem. MMGA, does simultaneous minimization and maximization of the same object function, preserving both individuals with the highest fitness value and individuals with the lowest fitness values. This multi-objective, or bi-objective, optimization method could also be called two headed GA or reciprocal GA (Mantere 2004).

In the constrained optimization heuristic method such as a genetic algorithm generates a lot of infeasible solutions that violate the constraints. The usual way to handle constraints is to add penalty term to the fitness function, so that the constraint violation is penalized. It is difficult to describe a proper penalty function and thus one is often either over penalize or under penalize. In over penalized situation all feasible solutions are considered to be better than infeasible. This means that the possibly beneficial genetic information that infeasible solutions may possess will disappear quickly from the population and it will get stuck to the local optima. On the other hand, in the case of under penalizing the penalty is not hard enough and all the solutions may violate constraints. Hence in this research work, to obtain the optimal local and global minima, Genetic algorithm is integrated with hill climbing approaches.

Pedro Diaz-Gomez et al (2005) discussed one of the primary approaches to the increasingly important problem of computer security is the Intrusion Detection System. Various architectures and approaches have been proposed including: Statistical, rule-based approaches; Neural Networks; Immune Systems; Genetic Algorithms; and Genetic Programming. This paper focuses on the development of an off-line Intrusion Detection System to analyze a Sun audit trail file. Off-line intrusion detection can be accomplished by searching audit trail logs of user activities for matches to patterns of events required for known attacks. Because such search is NP-complete, heuristic
methods will need to be employed as databases of events and attacks grow. Genetic Algorithms can provide appropriate heuristic search methods. However, balancing the need to detect all possible attacks found in an audit trail with the need to avoid false positives (warnings of attacks that do not exist) is a challenge, given the scalar fitness values required by Genetic Algorithms. This study discussed a fitness function independent of variable parameters to overcome this problem. This fitness function allows the IDS to significantly reduce both its false positive and false negative rate. This paper also describes extending the system to account for the possibility that intrusions are either mutually exclusive or not mutually exclusive.

Mike Sewell et al (2006) introduced a novel method of individual level adaptive mutation rate control called the rank-scaled mutation rate for genetic algorithms. The rank-scaled mutation rate controlled genetic algorithm, varies the mutation parameters based on the rank of each individual within the population. Thereby new mutation rates is formed by considering the distribution of the fitness of the population. The best fit mutates at the lowest rate and the least fit mutates at the highest rate. The complexity of the algorithm is therefore the order of an individual adaptation scheme and is lower than that of a self-adaptation scheme. The proposed algorithm is tested on two common problems, namely, numerical optimization of a function and the traveling salesman problem. The results show that the proposed algorithm outperforms both the fixed and deterministic mutation rate schemes hence it is best suited for problems with several local optimum solutions without a high demand for excessive mutation rates.

Intiaz Korejo et al (2009) discussed Genetic algorithms (GAs) as a class of stochastic optimization methods that is inspired by the principles of natural evolution. Adaptation of strategy parameters and genetic operators has become an important emerging and promising research area in GAs. Many
researchers are applying adaptive techniques to guide the search of GAs toward optimum solutions. The key component of GAs is mutation, which is a variation operator to create diversity for GAs. This paper investigates several adaptive mutation operators, that includes population level adaptive mutation operators and gene level adaptive mutation operators, for GAs and then compares their performance, that is based on a set of uni-modal and multi-modal benchmark problems. The experimental results show that, the gene level adaptive mutation operators are usually more efficient than the population level adaptive mutation operators for GAs.

Melanie Mitchel et al (2007) (discussed Genetic algorithms (GAs) show a major role in many artificial-life systems, but there is often little detailed understanding of why the GA performs as it does, and little theoretical basis on which to characterize the types of fitness landscapes that lead to successful GA performance. Genetic operators such as mutation and crossover are then applied probabilistically to the population to produce a new population (or generation) of individuals. New generations can be produced synchronously, so that the old generation is completely replaced, or asynchronously, so that generations overlap. The GA is considered to be successful if a population of highly fit individuals evolves as a result of iterating this procedure. When the GA is being used in the context of function optimization, success is measured by the discovery of bit strings that represent values yielding an optimum (or near optimum) of the given function. A common interpretation of GA behavior is that the GA is implicitly searching a space of patterns, the space of hyper planes in \{0, 1\} (where I is the length of bit strings in the space). Hyperplanes are represented by schemas, which are defined over the alphabet \{O, 1, *\}, where the * symbol means "don't care." Thus, *0 denotes the pattern, or schema, which requires that the second bit be set to 0 and will accept a 0 or a 1 in the first bit position. A bit string x obeying a schema S'S pattern is said to be an instance of s; for example, 00
and 10 are both instances of *0. In schemas, 1's and D's are referred to as defined bits, and the order of a schema is simply the number of defined bits. The fitness of any bit string in the population provides an estimate of the average fitness of the 2' different schemas of which it is an instance, so an explicit evaluation of a population of M individual strings is also an implicit evaluation of a much larger number of schemas. The GA's operation can be thought of as a search for schemas of high average fitness, carried out by sampling individuals in a population and biasing future samples towards schemas that are estimated to have above-average fitness.

Holland's Schema Theorem demonstrated that, under certain assumptions, schemas whose estimated average fitness remains above the population's average fitness will receive an exponentially increasing number of samples. That is, schemas judged to be highly fit will be emphasized in the population. However the Schema Theorem does not address the process by which new schemas are discovered', in fact, crossover appears in the Schema Theorem as a factor that slows the exploitation of good schemas. The "building-blocks hypothesis" stated that new schemas are discovered via crossover, which combines instances of low-order schemas (partial solutions or "building blocks") of estimated high fitness into higher-order schemas (composite solutions).

For example, if a string's fitness is a function of the number of 1's in the string, then a crossover between instances of two high-fitness schemas (each with many 1's) has a better than average chance of creating instances of even higher-fitness schemas. However, the actual dynamics of the discovery process—and how it interacts with the emphasis process—are not well understood and there is no general characterization of the types of landscapes on which crossover will lead to the discovery of highly fit schemas. Specifically, there is no firm theoretical grounding for what is perhaps the
most prevalent "folk theorem" about GAs—that they will outperform hillclimbers and other common search and optimization techniques on a wide spectrum of difficult problems, because crossover allows the powerful combination of partial solutions.

It is often the case that the search space, in which solutions to a problem exist, is extremely large and complex. For any search algorithm in such a space there exists a fundamental trade-off between exploration and exploitation. An example of a totally exploratory algorithm would be the enumeration of all cases, and selecting the best. This procedure, although eventually yielding the optimum solution, is far too inefficient to be of practical use in addressing most real-world problems.

In order for an algorithm to search a space efficiently it must be able to exploit opportunities for improved performance. This often involves making use of information acquired from previous evaluations of possible solutions. If such information is not used then the search could degenerate to the point at which it is little better than a random sampling of the solution space. An example of a search strategy which makes use of current information is a hill-climbing algorithm. Hill-climbing algorithms concentrate the search effort around the best solution found so far (exploitation), but it is likely that discovered solutions will be suboptimal on non-convex surfaces, because the sequence of trials will stagnate at local optima. Intuitive concepts such as 'peaks' and 'valleys' are often used to describe how such algorithms traverse the fitness landscape. However, such analogies need to be made with care.

The objective of a hill-climbing algorithm is to maximize the (local) fitness, i.e. climb one of the 'peaks'. The normal procedure for this involves making small changes to the solution's representation and accepting as the new best a solution which outperforms the current one. Following this
method and using an integer representation, climbs (with reference to Figure 2.1) such as those from X to X<sub>i</sub>, and from Y to Y<sub>i</sub>, would intuitively be how the search would be expected to proceed.

Fabien Duchateau et al 2014 designed a Benchmark for the Assessment of Schema Matching Tools. This work proposed XBenchMatch, a schema matching benchmark with available datasets and new measures to evaluate this manual post-match effort and the quality of integrated schemas. Finally, the work reported that the results obtained by different approaches, namely COMA++, Similarity Flooding and YAM. Also it showed that such a benchmark is required to understand the advantages and failures of schema matching approaches.

Saira Gillani et al 2013, proposed a Semantic Schema Matching Using DBpedia. Match is an operator that takes as an input two graph-like structures in semantic computing. The input can be database schemas or XML schemas and generates a mapping between the corresponding nodes of the two graphs. In semantic schema matching, it attempts to explore the mappings between the two schemas; based on their semantics by employing any semantic similarity measure. The work used DBpedia dataset while utilizing a hybrid ranking system to dig out the similarity between nodes of the two graphs.

Sumit Jain et al 2014, proposed a Survey of current Research in Schema Matching Technique. The authors focused in recent research trends in problem of Schema matching techniques for Web Database. Jafreen Hossain et al 2014, proposed a review paper on Semantic Schema Matching Approaches. An extensive review of the existing research work in the field of schema matching uncovers the significance of semantics in this subject. The authors listed out several possible future works based on the existing limitations.
In this thesis, Genetic algorithm with Hill-Climbing algorithm to improve the Schema matching automation process has been integrated. The following section discuss about the various existing clustering techniques.

2.5 EXISTING CLUSTERING ALGORITHMS

Clustering is considered as an unsupervised classification process. The clustering problem is to partition a dataset into groups (clusters) so that the data elements within a cluster are more similar to each other than data elements in different clusters by given criteria. A large number of clustering algorithms have been developed for different purposes. Based on the strategy of how data objects are distinguished, clustering techniques can be broadly divided in two classes: hierarchical clustering techniques and partitioning clustering techniques. However there is no clear boundary between these two classes. Some efforts have been done on the combination of different clustering methods for dealing with specific applications. Beyond the two traditional hierarchical and partitioning classes, there are several clustering techniques that are categorized into independent classes, for example, density-based methods, Grid-based methods and Model based clustering methods. A short review of various existing algorithms related to these methods is described below.

2.5.1 Partitioning Clustering Algorithms

Partitioning clustering algorithms, such as K-means, K-medoids PAM, CLARA and CLARANS assign objects into k (predefined cluster number) clusters, and iteratively reallocate objects to improve the quality of clustering results. K-means (Tajunisha & Saravanan 2010) is the most popular and easy-to understand clustering algorithm. However, K-means algorithm is very sensitive to the selection of the initial centroids; in other words, the different centroids may produce significant differences of clustering results.
Another drawback of K-means is that, there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple solution would be to compare the results of multiple runs with different k numbers and choose the best one according to a given criterion, but when the data size is large, it would be very time consuming to have multiple runs of K-means and the comparison of clustering results after each run.

To solve this problem, some efforts have been made based on K-medoids, for example PAM (Partitioning Around Medoids) was proposed by Kaufman and Rouseeuw. PAM inherits the features of K-medoids clustering algorithm. Meanwhile, PAM equips a medoids swap mechanism to produce better clustering results. PAM is more robust than k-means in terms of handling noise and outliers, since the medoids in PAM are less influenced by outliers. With the O(k(n-k)2) computational cost for each iteration of swap (where k is the cluster number, n is the items of the data set), it is clear that PAM only performs well on small-sized datasets, but does not scale well to large datasets.

In practice, PAM is embedded in the statistical analysis systems, such as SAS, R, S+ and etc. to deal with the applications of large sized datasets, i.e., CLARA (Clustering Large Applications). By applying PAM to multiple sampled subsets of a dataset, for each sample, CLARA can produce the better clustering results than PAM in larger data sets. But the efficiency of CLARA depends on the sample size. On the other hand, a local optimum clustering of samples may not the global optimum of the whole data set. Ng and Han abstracts the medoids searching in PAM or CLARA as searching k subgraphs from n point’s graph, and based on this understanding, they propose a PAM-like clustering algorithm called CLARANS (Clustering Large Applications based upon Randomized Search). While PAM searches the whole graph and CLARA searches some random sub-graphs, CLARANS
randomly samples a set and selects k medoids in climbing Sub-graph Mountains. CLARANS selects the neighboring objects of medoids as candidates of new medoids. It tests the sample subsets to verify medoids for multiple times to avoid bad samples. It is known that, multiple times sampling of medoids verification is time consuming. Thus it limits CLARANS from clustering very large datasets in an acceptable time period.

In this research work, Centroid based clustering algorithms are used to improve the Schema matching automation.

### 2.5.2 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms assign objects as a tree structured clusters, i.e., a cluster can have data points or representatives of the lower level clusters. According to their clustering process hierarchical clustering algorithms can be classified into two categories: agglomerative and divisive.

- **Agglomerative**: In a separate cluster it starts with each of the units and ends up with a single cluster that contains all units.

- **Divisive**: It starts with a single cluster consisting of all units and then form new clusters by dividing those clusters that had been determined at previous stages until one ends up with clusters containing individual units.

AGNES (Agglomerative Nesting) uses agglomerative strategy to merge clusters. At the beginning AGNES arranges each object as a cluster, and then merges them as upper level clusters by giving agglomerative criteria step-by-step, until all objects form a cluster. The similarity between two clusters is calculated by the similarity function of the closest pair of data points in the two clusters, i.e., single link.
DIANA (Divisive Analysis) adopts an opposite merging strategy; it initially puts all objects in one cluster, then splits them into several level clusters until each cluster contains only one object. The merging/splitting decisions are critical in AGNES and DIANA. On the other hand, with \( O(n^2) \) computational cost, their application is not scalable to very large datasets.

Zhang et al proposed an effective hierarchical clustering method to deal with the above problems, BIRCH (Balanced and Iterative Reducing and Clustering using Hierarchies). BIRCH summarizes an entire dataset into a CF-tree and then runs a hierarchical clustering algorithm on a multi-level compression technique, CF-tree, to get the clustering result. Its linear scalability is good at clustering with a single scan and its quality can be further improved by a few additional scans. It is an efficient clustering method on arbitrarily shaped clusters. But BIRCH is sensitive to the input order of data objects, and can also only deal with numeric data. This limits its stability of clustering and scalability in real world applications.

CURE uses a set of representative points to describe the boundary of a cluster in its hierarchical algorithm. But with the increase of the complexity of cluster shapes, the number of representative points increases dramatically in order to maintain the precision.

Guha 2008 et al introduced the hierarchical agglomerative clustering algorithm CURE (Clustering Using Representatives). This algorithm has a number of novel features of general importance. It takes special steps to handle outliers and to provide labeling in assignment stage. It also uses two techniques to achieve scalability: data sampling, and data partitioning. CURE creates \( p \) partitions, so that fine granularity clusters are constructed in partitions first. A major feature of CURE is that it represents a cluster by a fixed number, \( c \), of points scattered around it. The distance between two clusters used in the agglomerative process is the minimum of
distances between two scattered representatives. Therefore, CURE takes a middle approach between the graph (all-points) methods and the geometric (one centroid) methods. Single and average link closeness are replaced by representatives’ aggregate closeness. Selecting representatives scattered around a cluster makes it possible to cover non-spherical shapes. As before, agglomeration continues until the requested number k of clusters is achieved. CURE employs one additional trick: originally selected scattered points are shrunk to the geometric centroid of the cluster by a user-specified factor $\alpha$. Shrinkage suppresses the affect of outliers; outliers happen to be located further from the cluster centroid than the other scattered representatives. While the algorithm CURE works with numerical attributes (particularly low dimensional spatial data), the algorithm ROCK developed by the same researchers targets hierarchical agglomerative clustering for categorical attributes.

CHAMELEON employs a multilevel graph partitioning algorithm on the k-Nearest Neighbor graph, which may produce better results than CURE on complex cluster shapes for spatial datasets. But the high complexity of the algorithm prevents its application on higher dimensional datasets.

The hierarchical agglomerative algorithm CHAMELEON uses the connectivity graph $G$ corresponding to the K-nearest neighbor model sparsification of the connectivity matrix: the edges of K most similar points to any given point are preserved, the rest are pruned. CHAMELEON has two stages. In the first stage small tight clusters are built to ignite the second stage. This involves a graph partitioning. In the second stage agglomerative process is performed. It utilizes measures of relative inter-connectivity $RI(C_i,C_j)$ and relative closeness $RC(C_i,C_j)$; both are locally normalized by internal interconnectivity and closeness of clusters $C_i$ and $C_j$. In this sense the modeling is dynamic: it depends on data locally. Normalization involves
certain non-obvious graph operations. CHAMELEON relies more on graph partitioning implemented in the library HMETIS. Agglomerative process depends on the thresholds provided by the user. A decision is made to merge based on the combination $RI(C_i,C_j)$ which is of local measures. The algorithm does not depend on the data model that is assumed. Clusters of different shapes, densities, and sizes in 2D (two-dimensional) space have been proven to be found. It has a complexity of $O(Nm)+N\log(N)+m^2\log(m)$, where $m$ is the number of sub-clusters built during the first initialization phase.

### 2.5.3 Density-Based Clustering Algorithms

For each point of a cluster the neighbor of a given unit distance contains at least a minimum number of points, i.e. the density in the neighborhood should reach some threshold, which has been the primary idea of density-based methods. Anyway, this idea is based on the assumption that the clusters are in the spherical or regular shapes.

To adopt density- reach ability and density connectivity for handling the arbitrarily shaped clusters and noise DBSCAN (Density-Based Spatial Clustering of Applications with Noise) was proposed. But DBSCAN is very sensitive to the parameter Eps (unit distance or radius) and MinPts (threshold density), because the user is expected to estimate Eps and MinPts before doing cluster exploration.

DENCLUE (Density-based Clustering) that performs well on clustering large datasets with higher noise is a distribution-based algorithm. DENCLUE needs a large number of parameters which is significantly faster than existing density-based algorithms. Investigating the arbitrarily shaped clusters is done good by the OPTICS, but it is applicable only to small or medium datasets, because of its non-linear complexity.
2.5.4 Grid-Based Clustering Algorithms

Grid-based clustering algorithm is based on the clustering oriented query answering in multilevel grid structures. The upper level stores the information of its next level, thus the grids create cells among the connected levels.

Example for grid-based methods is STING (Statistical Information Grid Approach), CLIQUE, and the combination of grid-density based technique Wave Cluster. The grid-based methods are efficient on clustering data with the complexity of O (N). However the key concern of grid-based techniques is how to select the size of grids. This based on the user’s experience.

The algorithm STING (Statistical Information Grid-based method) works with spatial data and is designed to enable “region oriented” queries. STING assembles summary statistics in a hierarchical tree of nodes that are grid-cells. Each cell has four (default) children and stores a point count, and attribute-dependent measures: mean, standard deviation, minimum, maximum, and distribution type. Measures are accumulated starting from bottom level cells. They are further aggregated to higher-level cells (e.g., minimum is equal to a minimum among the children-minimums). STING+ is an enhancement of STING that targets dynamically evolving spatial databases using a hierarchical cell organization similar to its predecessor. In addition, STING+ enables active data mining by supporting user defined trigger conditions (e.g., “at least 10 cellular phones are in use per square mile in a region of at least 10 square miles”, or “usage drops by 20% in a described region”). The related measures (“sub-triggers”) are stored and updated over the cell tree. They are suspended until the trigger fires with user-defined action. Four types of conditions are supported: absolute and relative
conditions on regions (a set of adjacent cells), absolute and relative conditions on certain attributes.

CLIQUE starts with the definition of a unit, an elementary rectangular cell in a subspace. Only units whose densities exceed a threshold $\tau$ are retained. A bottom-up approach of finding such units is applied. A cluster is defined as a maximal set of connected dense units and is represented by a DNF expression that specifies a finite set of maximal segments (called regions) whose union is the cluster. Effectively, CLIQUE performs attribute selection (it selects several subspaces) and produces a view of data from different perspectives. The result is a series of cluster systems in different subspaces. Such systems overlap. Thus, CLIQUE resembles more data description than data partitioning. If $q$ is a highest subspace dimension selected, the complexity of dense unit’s generations is $O(\text{const } q + qN)$. Finding clusters is a task quadratic in terms of units. The algorithm ENCLUS (Entropy-based Clustering) follows in the footsteps of CLIQUE, but uses a different criterion for a subspace selection.

The hierarchy of grids allows definition of the Hausdorff Fractal Dimension (HFD). HFD of a set is the negative slope of a log-log plot of the number of cells $\text{Cell}(r)$ (occupied by a set) as a function of a grid size $r$. A fast (box counting) algorithm to compute HFD was introduced. The concept of HFD is fundamental to the FC (Fractal Clustering) algorithm dealing with numeric attributes. It works with several layers of grids. Cardinality of each dimension increases 4 times with each next layer. Although only occupied cells are kept to save memory, memory usage is still a significant problem. FC starts by initializing $k$ clusters. The initialization threshold and a data sample are used at this stage to come up with an appropriate $k$. Then FC scans the full data incrementally attempting to add the incoming point to a cluster so that to minimally increase HFD. If the smallest increase exceeds the threshold
τ, a point is declared an outlier. The FC algorithm has few appealing properties:

- Increasing structure
- Suspend able nature always ready for on-line assignments
- Capability to determine clusters of irregular shapes

It also has a few drawbacks:

- Data order dependency
- Strong dependency on parameter and clusters initialization

2.5.5 Model-based Clustering Algorithms

Model-based clustering methods are based on the guess and they optimize the fit between the data objects and some mathematical model, for example statistical approach, neural network approach and other AI approaches. When facing an unknown data distribution, choosing a suitable one from the model based candidates is still a major challenge. The drawback of this model based clustering algorithm is very high computational cost for very large data set.

2.6 SUMMARY

The taxonomy of the schema matching technique, various prototypes and tools used for the schema matching technique with detailed comparison are given in this chapter. The various existing research works related to the schema matching automation and the merits and demerits are discussed. The existing research works related to Genetic Algorithm and
various Clustering Algorithms with their importance are also highlighted in this chapter.

The forthcoming chapters attempt to an efficient Centroid Based Clustering Algorithm technique and Optimal Hill Climbing Approach with Genetic Algorithm to improve the schema matching automation.