# Formulation Schema Matching Problem for Combinatorial Optimization Problem

Zhi Zhang [1*], Haoyang Che [2], Pengfei Shi [1], Yong Sun [3], Jun Gu [3]
[1] Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University,
Shanghai 200030, China
[2] Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China
[3] Department of Computer Science, Science & Technology University of Hong Kong, Hong Kong, China

**Abstract:** Schema matching is the task of finding semantic correspondences between elements of two schemas, which plays a key role in many database applications. In this paper, we cast the schema matching problem (SMP) into a multi-labeled graph matching problem. First, we propose an internal schema model: multi-labeled graph model, and transform schemas into multi-labeled graphs. Therefore, SMP reduce to a labeled graph matching, which is a classic combinatorial problem. Secondly, we study a generic graph similarity measure based on Contrast Model, and propose a versatile optimization function to compare two multi-labeled graphs. Then, we can design the optimization algorithm to solve the multi-labeled graph matching problem. Based on the matching result obtained by greedy matching, we implement a fast hybrid search algorithm to find the feasible matching results. Finally, we use several schemas to test the hybrid search algorithm. The experimental results confirm that the algorithm model and the hybrid algorithm are effective.

## Introduction

The goal of schema matching is to find semantic correspondences between the elements of two schemas. It plays a key role in many database applications such as schema integration, data warehousing, e-business, XML message mapping, and semantic query processing [19]. However schema matching still remains largely a manual, labor-intensive, and expensive process.

Problem formulation is an extremely important part of problem solving. The choice of a good formulation can result in order of magnitude savings in solving cost. In this paper, we study how to cast the schema matching problem (SMP) into a multi-labeled graph matching problem. For multi-labeled graph matching, which is a kind of graph matching problems. It is well known that graph matching is a classic combinational optimization problem. There are many approaches to deal with graph matching problem. Therefore, based on the framework of graph matching, we can design heuristic approach to attack schema matching.

First, we propose a meta-model: multi-labeled graph model, to represent various kinds of schemas. We extract the elements of schema as the vertices of a graph, and the properties of elements as the labels of vertices, where each vertex and edge can be associated with a set of labels describing its properties. For labeled graph matching, we want to obtain the correspondences between the vertices of two graphs. Therefore, we discuss a generic graph similarity measure based on

* Corresponding author. *E-mail address*: zzh@sjtu.edu.cn (Z. Zhang).

Contrast Model, and propose an optimization function based on multi-labeled graph similarity. Up to now, we transform SMP into a multi-labeled graph matching problem which is a classic combinational problem, and develop the algorithmic model for SMP. Finally, we implement a hybrid search algorithm to find the feasible matching correspondences.
The paper is organized as follows. Section 2 discusses related work on schema matching. Section 3 presents a meta-model of schemas: multi-labeled graph. Section 4 introduces the definition of SMP based on multi-labeled graph. We call SMP as multivalent matching, which is composed of multivalent correspondences. Then, we formulize SMP as a multi-labeled graph matching problem. Section 5 investigates a generic graph similarity measure based on Contrast Model, and proposes an objective function to schema matching. Then, Section 6 studies a hybrid search algorithm in detail. In section 7, we use some experiments to evaluate our approach. Section 8 makes some concluding remarks and discusses our future work.

## Related Work

Numerous solutions have been proposed in specific applications to solve SMP. Madhavan *et al*. [13, 18] implemented a Cupid system to achieve semi-automatic schema matching, which uses a hybrid matching algorithm comprising linguistic and structural schema matching techniques, and computes similarity coefficients with the assistance of a precompiled thesaurus; Machine learning is a promising technique especially for evaluating data instances to predict element similarity, the LSD system [10] uses machine-learning techniques to match a pair of schemas. The accuracy of the predictions depends on a suitable training. The predictions of individual matchers are combined by a so called meta-learner, which weights the predictions from a matcher according to its accuracy shown during the training phase; Berlin and Motro [3] devised Automatch system for database schema matching which also uses machine learning techniques, bases primarily on Bayesian learning. Automatch acquires probabilistic knowledge from examples of schemas that have been "mapped" by domain experts into a knowledge base of database attributes called the attribute dictionary. Then, Automatch uses the attribute dictionary to find an optimal matching; Melnik *et al*. [14, 15] used the graph matching algorithm – Similarity Flooding to achieve schema matching, which can measure the similarity between vertices of two schemas. The similarity between pairs of vertices, described by a nonnegative vector, is computed iteratively until convergence to a fixed point; Bouquet [5] viewed each semantic schema as a context, and proposed an algorithm based on SAT solver to matching two schemas; Furthermore, based on [5], Giunchiglia *et al*. [11] developed S-Match algorithm which is a schema-based schema/ontology matching system implementing semantic matching approach. It takes two graph-like structures (e.g., database schemas or ontologies) as input and returns semantic relations between the nodes of the graphs that correspond semantically to each other as output. They used five semantic relations to represent the matching relationships between two elements: equivalence, more general, less general, mismatch, and overlapping; Miller proposed a semi-automated mapping tool Clio to obtain mappings between a given target schema and a new schema [16]. The algorithm regards schema mapping as

query discovery, which uses query search method to match the schemas; Do and Rahm [8] devised the COMA schema matching system. It follows a composite approach, which provides an extensible library of different matchers and supports various ways for combining match results. For the details of SMP, we can refer to two surveys of schema matching [9, 19].
Graphs are versatile representation tools that have been used in schema matching [13, 14, 15]. In [24], Zhang *et al*. proposed a *meta-meta structure* based on universal algebra, which is named *multi-labeled schema*. In [25], they use a *multi-labeled graph model* as the internal schema model, which is an instance of *multi-labeled schema*. As a result, SMP can be reduced to a graph matching problem. The graph matching problem (i.e., graph homomorphism) is one of the classic combinatorial optimization problems.
To retrieve similar case in a CBR system, Champin and Solnon [6] proposed a generic similarity measure model to compare multi-labeled graphs based on *Contrast Model* [21]. Contrast Model has been proposed by Tversky, wherein similarity is determined by matching features of compared entities. Based on their work, Zhang *et al*. [25] used the labeled graph similarity model to design a greedy matching algorithm.
In this paper, we formulize the schema matching problem as a multi-labeled graph matching problem. Then, we discuss the similarity measure of multi-labeled graph based on Contrast Model, and propose the best matching result based on features of two schemas. At last, we design a hybrid search algorithm to solve this combinational optimization problem.

# Multi-labeled Graph Model

## Multi-labeled Schema

There are many kinds of schemas, such as relational model, object-oriented model, ER model, conceptual graph, DTD, XML schema, etc. In [24], Zhang *et al*. proposed a meta-meta model of schema: *multi-labeled schema*, which views schemas as finite structures over the specific signatures.

**Definition 1.** (*Schema*) *A schema* $\mathcal{S}$ *is a finite structure over a signature* $\sigma$, *consists of individual set* $I^{\mathcal{S}}$, *label collection* $Lab^{\mathcal{S}}$, *function set* $F^{\mathcal{S}}$, *relation set* $R^{\mathcal{S}}$, *written a 4-tuples* $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$, *where*,

1. $\sigma$ *is a finite collection that is composed of individual symbols*, *label symbols*, *function symbols*, *and relation symbols*, *where*, *each function symbol f or relation symbol R*, *respectively comes associated with an arity*, *ar(f) and ar(R)*, *which are non-negative integers*.

2. $I^{\mathcal{S}} = \{s_1, s_2, \cdots, s_n\}$ *is a finite nonempty set that includes individuals*, *which denote the prepared-matching objects*. *Each of them is uniquely identified by an object identifier (OID)*.

3. $Lab^{\mathcal{S}} = \{Lab_1^{\mathcal{S}}, Lab_2^{\mathcal{S}}, \cdots, Lab_i^{\mathcal{S}}\}$ *is a finite constant collection that includes the label sets for individuals. The labels are the strings for describing the properties of individuals.*

4. $F^{\mathcal{S}} = \{f_1^{\mathcal{S}}, f_2^{\mathcal{S}}, \cdots f_j^{\mathcal{S}}\}$ *is a finite set that includes the labeling functions, which are partial function. The domain of each function is the individual set, accordingly, the codomain is the label collection.*

5. $R^{\mathcal{S}} = \{R_1, R_2, \cdots, R_k\}$ *is a finite nonempty set that includes the relations between individuals. If R is a b-ary relation, then* $R \subseteq (I^{\mathcal{S}})^b$ .

6. *The size of schema* $\mathcal{S}$ *is the size of individuals and is denoted by* $|I^{\mathcal{S}}|$.

## Multi-labeled Graph Model

Based on multi-labeled schema, Zhang *et al*. [25] proposed a multi-labeled graph model, which is an instance of multi-labeled schema, to describe various schemas, where each vertex and edge can be associated with a set of labels describing its properties. Such a multi-labelling could be very useful to describe schemas more accurately.

**Definition 2.** *A schema* $\mathcal{S}$ *can be represented by a labeled graph structure* $\mathcal{S} = (V^{\mathcal{S}}, E^{\mathcal{S}}, Lab^{\mathcal{S}}, r_{V^{\mathcal{S}}}, r_{E^{\mathcal{S}}})$.

1. *V is the finite set of vertices. Vertices are prepared-matching objects, and each of them is uniquely identified by an object identifier (OID).*

2. $E^{\mathcal{S}} \subseteq V^{\mathcal{S}} \times V^{\mathcal{S}}$ *is the finite set of edges. Each of edges denotes the relation between two vertices.*

3. $Lab^{\mathcal{S}} = \{Lab_{V^{\mathcal{S}}}, Lab_{E^{\mathcal{S}}}\}$ *is the finite constant collection of labels. The labels are strings for describing the properties of vertices and edges.* $Lab_{V^{\mathcal{S}}}$ *is the finite collection of vertex labels;* $Lab_{E^{\mathcal{S}}}$ *is the finite collection of edge labels.*

4. $r_{V^{\mathcal{S}}} \subseteq V \times Lab_{V^{\mathcal{S}}}$ *is a relation associating labels to verteices, i.e.,* $r_{V^{\mathcal{S}}}$ *is the set of couples* $(v_i, l)$ *such that vertex* $v_i$ *is labeled by l.* $r_{V^{\mathcal{S}}}$ *is called vertex feature of* $\mathcal{S}$.

5. $r_{E^{\mathcal{S}}} \subseteq E \times Lab_{E^{\mathcal{S}}}$ *is a relation associating labels to edges, i.e.,* $r_{E^{\mathcal{S}}}$ *is the set E of triples* $(v_i, v_j, l)$ *such that* $(v_i, v_j)$ *is labeled by l.* $r_{E^{\mathcal{S}}}$ *is called edge feature of* $\mathcal{S}$.

6. $descr(\mathcal{S}) = r_{V^{\mathcal{S}}} \cup r_{E^{\mathcal{S}}}$ *is the set of all vertex and edge features of a schema* $\mathcal{S}$ *that completely describes the schema* $\mathcal{S}$.

7. $|V| = n$ *is the cardinality of schema* $\mathcal{S}$.

In Table 1, we show the correspondences between multi-labeled schema and multi-labeled graph:

| *Multi-labeled schema* | | *Multi-labeled graph* | *domain* | *codomain* |
|---|---|---|---|---|
| $I^{\mathcal{S}}$ | | $V^{\mathcal{S}}$ | - | - |
| $Lab^{\mathcal{S}}$ | | $Lab^{\mathcal{S}}$ | - | - |
| $R^{\mathcal{S}}$ | | $E^{\mathcal{S}}$ | - | - |
| $F^{\mathcal{S}}$ | $f(V^{\mathcal{S}}) \to Lab_{V^{\mathcal{S}}}$ | $r_{V^{\mathcal{S}}} \subseteq V^{\mathcal{S}} \times Lab_{V^{\mathcal{S}}}$ | $V^{\mathcal{S}}$ | $Lab_{V^{\mathcal{S}}}$ |
| | $f(V^{\mathcal{S}} \times V^{\mathcal{S}}) \to Lab_{E^{\mathcal{S}}}$ | $r_E \subseteq E^{\mathcal{S}} \times Lab_{E^{\mathcal{S}}}$ | $V^{\mathcal{S}} \times V^{\mathcal{S}}$ | $Lab_{V^{\mathcal{S}}}$ |

Table 1. The correspondences between multi-labeled schema and multi-labeled graph

## Encode schemas into labeled graphs

### *Encoding rules*

For encoding relational schemas, XML schemas, SQL views, etc. as multi-labeled graphs, we use the following rules:

1. *A vertex of graph represents the prepared matching object of schema*. *V is the vertex set that comprises all prepared matching objects of a schema*;
2. *The labels of a vertex are composed of properties of a prepared matching object*;
3. *An edge represents the relation between two prepared matching objects of schema*. *E is the edge set that comprises all relations of schema* ($E \subseteq V \times V$);
4. *The labels of one edge comprise properties of two prepared matching objects*, *such as is-a*, *part-of*, *etc*.

### *Motivating Scenario*

The SMP is a critical problem for interoperability in heterogeneous information sources, which plays a key role in many database applications. In this section, we introduce a real-life scenario happens in e-business to illustrate our algorithm framework.

For a multinational company, there are two subsidiary companies locate at different countries (company *A* in *S* area and company *B* in another *T* area), and the companies want to share and interoperate their customers' information by Web Service. The XML description schemas are deployed on their own XML web services. However, the XML schemas used by the company undergoes periodic changes due to the dynamic nature of its business. If do the schema matching by manual operate, it is a tiresome and costly work. Moreover, if the company *A* changes its customer information database structure, and the XML schema is changed synchronously, but they do not notice the company *B* to update its Web Service correspondingly, under this conditions, if the interoperate wants to carry out successfully, two web agents have to automatic matching their schemas again, and need not manual acting. The automatic schema matching can improve the reliability and usability of Web services. Now, two XML schemas are shown in Fig.1, which are based on BizTalk Schema specification, where, Schema $\mathcal{S}$ is used by company *A*, and Schema $\mathcal{T}$ is deployed by company *B*.

```
<Schema name="Schema S "
  xmlns="urn:schemas-microsoft-com:xml-data">
  <ElementType name="AccountOwner">
      <element type="Name"/>
      <element type="Address"/>
      <element type="Birthdate"/>
      <element type="TaxExempt"/>
  </ElementType>
  <ElementType name="Address">
      <element type="Street"/>
      <element type="City"/>
      <element type="State"/>
      <element type="ZIP"/>
  </ElementType>
</Schema>
```

```
<Schema name="Schema T "
  xmlns="urn:schemas-microsoft-com:xml-data">
  <ElementType name="Customer">
      <element type="CFname"/>
      <element type="CLname"/>
      <element type="CAddress"/>
  </ElementType>
  <ElementType name="CustomerAddress">
      <element type="Street"/>
      <element type="City"/>
      <element type="Province"/>
      <element type="PostalCode"/>
  </ElementType>
</Schema>
```

*Fig. 1 Two XML schemas (BizTalk)*

At first, from Fig.1, we can obtain the vertices and edges of two schemas, which are shown in Fig.2, where, $V^{\mathcal{S}} = \{s_1, s_2, \cdots, s_{11}\}$, $V^{\mathcal{T}} = \{t_1, s_2, \cdots, t_{10}\}$.

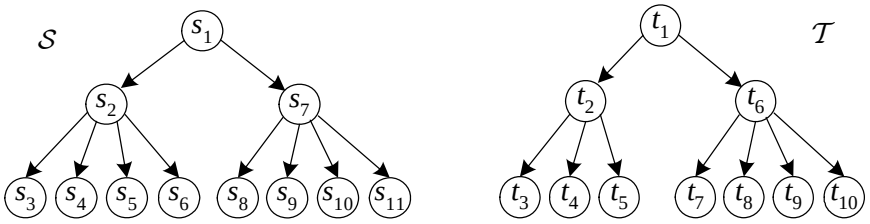


*Fig. 2 Vertices and edges of $\mathcal{S}$ and $\mathcal{T}$*

Then, by Definition 2, Table 2 shows the labels of vertices. $Lab_V$ includes the name set $Lab_{Vname}$, the concept set $Lab_{Vconcept}$ or the type set $Lab_{Vtype}$. In the same way, $Lab_E$ can include the labels for edges. Here, $Lab_E$ = {part-of}.

| OID | *Labels of schema* $\mathcal{S}$ | | | *Labels of schema* $\mathcal{T}$ | | |
|---|---|---|---|---|---|---|
| | $Lab^{\mathcal{S}}_{Vname}$ | $Lab^{\mathcal{S}}_{Vconcept}$ | $Lab^{\mathcal{S}}_{Vtype}$ | $Lab^{\mathcal{T}}_{Vname}$ | $Lab^{\mathcal{T}}_{Vconcept}$ | $Lab^{\mathcal{T}}_{Vtype}$ |
| 1 | Schema $\mathcal{S}$ | Schema | Schema | Schema $\mathcal{T}$ | schema | Schema |
| 2 | AccountOwner | account + owner | ElementType | Customer | customer | ElementType |
| 3 | Name | name | element | CFname | first name | element |
| 4 | Address | address | element | CLname | last name | element |
| 5 | Birthdate | birthdate | element | CAddress | address | element |
| 6 | TaxExempt | tax-exempt | element | CustomerAddress | address | ElementType |
| 7 | Address | address | ElementType | Street | street | element |
| 8 | Street | street | element | City | city | element |
| 9 | City | city | element | Province | province | element |
| 10 | State | state | element | PostalCode | postal code | element |
| 11 | ZIP | ZIP | element | | | |

*Table 2. The labels of vertices*

# Multi-labeled Graph Matching

## Schema Matching

The goal of schema matching is to find the semantic correspondences between the elements of two schemas. We describe SMP informally as follows:

**Problem 1. SMP**
**Instance**: *Given two schemas* $\mathcal{S} = (V^{\mathcal{S}}, E^{\mathcal{S}}, Lab^{\mathcal{S}}, r_{V^{\mathcal{S}}}, r_{E^{\mathcal{S}}})$ *and* $\mathcal{T} = (V^{\mathcal{T}}, E^{\mathcal{T}}, Lab^{\mathcal{T}}, r_{V^{\mathcal{T}}}, r_{E^{\mathcal{T}}})$, $\mathcal{S}$ *is a source schema, and* $\mathcal{T}$ *is a target schema.*
**Question**: *To find the semantic correspondences between vertices in* $V^{\mathcal{S}}$ *and* $V^{\mathcal{T}}$.

In [24, 26], Zhang *et al*. investigated the formal framework for SMP, they proposed the concept of individual matching: if one or more labels of vertex *s* in $\mathcal{S}$ are semantically related to corresponding labels of vertex *t* in $\mathcal{T}$, or the relations of *s* and the relations of *t* are semantically equivalent, then we define that *s* and *t* are matched. Based on the definition of individual matching, they presented an important notion: multivalent matching.

## Multivalent Matching

*Multivalent matching*: a vertex of one schema may be associated with a set of vertices of another schema, which can characterize the *many-to-many* matching between two schemas [25].

**Definition 3.** *If* $\mathcal{S}$ *is the source schema,* $\mathcal{T}$ *is the target schema, the matching result of two schemas is a set* $\boldsymbol{m} \subseteq V^{\mathcal{S}} \times V^{\mathcal{T}}$ *that contains every matched couple* $(s, t) \in V^{\mathcal{S}} \times V^{\mathcal{T}}$.

The matching couples are called *multivalent correspondences*, which are binary relationships that aiming to establish many-to-many correspondences between the vertices of two schemas.

## Matching State and Matching Space

By Definition 3, the matching result can be represent as a relation set *m*, we introduce an equivalent concept: matching matrix or matching state to denote the solution of schema matching.

**Definition 4.** (*Matching Matrix*) If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, where, $V^{\mathcal{S}} = \{s_1, s_2, \cdots, s_n\}$ and $V^{\mathcal{T}} = \{t_1, t_2, \cdots, t_k\}$ denote vertex sets of $\mathcal{S}$ and $\mathcal{T}$ respectively. A matching state *m* is a $n \times k$ 0-1 matrix:

$$\boldsymbol{m} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,k} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,k} \\ \vdots & \vdots & \cdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,k} \end{bmatrix}, \quad \begin{array}{l} m_{i,j} \in \{0, 1\}, \\ n = |V^{\mathcal{S}}|, \; k = |V^{\mathcal{T}}| \end{array} \qquad 1)$$

where, $m_{i,j} = 1$ denotes $s_j$ and $t_j$ are matched and $m_{i,j} = 0$ denotes $s_j$ and $t_j$ are unmatched. All the matching couples compose the result of schema matching.

Given an assignment to a $|V^{\mathcal{S}}| \times |V^{\mathcal{T}}|$ matrix, we can obtain a possible matching result of two schemas. All of these matching states constitute the matching space.

**Definition 5.** (*Matching Space*) If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, where, $V^{\mathcal{S}} = \{s_1, s_2, \cdots, s_n\}$ and $V^{\mathcal{T}} = \{t_1, t_2, \cdots, t_k\}$ denote vertex sets of $\mathcal{S}$ and $\mathcal{T}$ respectively. All the assignments of a $|V^{\mathcal{S}}| \times |V^{\mathcal{T}}|$ $(n \times k)$ matrix constitute the matching space *M*, where,

$$M := \left\{ m \;\middle|\; m_{i,j} \in \{0, 1\}, \; \begin{array}{l} i \in \{1, 2, \cdots, n\} \\ j \in \{1, 2, \cdots, k\} \end{array} \right\} \qquad 2)$$

The scale of matching space is the number of the matching states: $|M| = 2^{n \times k}$.

## Schema Homomorphism

Zhang *et al*. [24] prove that: *Two schemas $\mathcal{S}$ and $\mathcal{T}$ are matched iff there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$, $\mathcal{S} \to \mathcal{T}$* . In Table 1, we show the correspondences between multi-labeled schema and multi-labeled graph, here, we present the definition of schema homomorphism based on the multi-labeled graph:

**Definition 6.** *A schema homomorphism (SHOM)* $\varphi : \mathcal{S} \to \mathcal{T}$ *from the source schema* $\mathcal{S}$ *to the target schema* $\mathcal{T}$ *is a mapping* $\varphi : V^{\mathcal{S}} \to V^{\mathcal{T}}$, *which is a set of multivalent correspondences such that*:

Condition 1. *There exists a labeling function symbol f of arity n*

$f^{\mathcal{S}}(s_1, \cdots, s_n) = l_n^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(\varphi(s_1), \cdots, \varphi(s_n)) = \varphi(l_n^{\mathcal{S}})$ , *for* $s_1, \cdots, s_n \in V^{\mathcal{S}}$ , $l_n^{\mathcal{S}} \in Lab^{\mathcal{S}}$

Condition 2. *There exists a semantic relation symbol R of arity m*

$R^{\mathcal{S}}(s_1, \cdots, s_m)$ *holds* $\Rightarrow R^{\mathcal{T}}(\varphi(s_1), \cdots, \varphi(s_m))$ *holds* , *for* $s_1, \cdots, s_m \in V^{\mathcal{S}}$

Because we use the multi-labeled graphs to represent the schemas, the schema homomorphism problem is reduced to a graph homomorphism problem, also called the graph matching problem. Now, we formulize SMP as a multi-labeled graph matching problem, which is a NP-hard problem [2, 6].
Based on SHOM and the multi-labeled graph model, for solving SMP: we will find a semantic homomorphism between two multi-labeled graphs, the homomorphic mapping includes the matching correspondences between two graphs $\mathcal{S}$ and $\mathcal{T}$. In the rest sections, we will develop a practical matching algorithm to solve this intractable problem.

## Similarity of Multi-labeled Graphs Based on Contrast Model

There are many methods to compare the similarity of two graphs, such as graph isomorphism, subgraph isomorphism, graph edit distance, maximum common subgraph, iterative method [2, 4, 20, 25], etc.
In this paper, we propose the multi-labeled graph as the meta-model for schemas, so we will study how to achieve schema matching by using the multi-labeled graph matching method. Champin and Solnon [6] proposed a generic method to measure the similarity of two directed multi-labeled graphs, which is based on the features of vertices and edges, i.e., the Contrast Model of Tversky [21]. Based on Contrast model, we investigate a schema matching approach based on common features between two multi-labeled graphs.

### Similarity Measure: Contrast Model

In [21], Tversky proposed a similarity approach: Contrast Model. In this model, entities are represented as a collection of features (e.g., object *a* can be represented by a feature set *A*), and similarity between objects *a* and *b* can be computed by:

$$sim_{Tversky}(a, b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A) \qquad 3)$$

The similarity of *A* to *B* is expressed as a linear combination of the measure of the common and distinctive features. The term $A \cap B$ represents the features that items *A* and *B* have in common. $(A - B)$ represents the features that *A* has but *B* does not. $(B - A)$ represents the features of *B* that are not in *A*. $\theta$, $\alpha$, and $\beta$ are weights for the common and distinctive components, and the function *f* is often simply assumed to be additive. A feature may be any property, characteristic or aspect of an object [12]. Fig.3 shows the basic principle of Contrast Model.

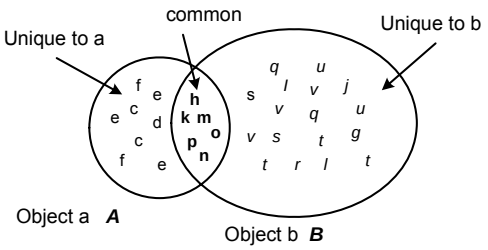


*Fig.3 Representation of two objects that each contains its own unique features and also contains common features. An important aspect of Tversky's model is that similarity depends not only on the proportion of features common to the two objects but also on their unique features (i.e., the differences between two objects). Each letter here represents a feature.*

A number of models are similar to Contrast Model in basing similarity on features and in using some combination of the $A \cap B$, $(A - B)$, and $(B - A)$, such as Sjoberg proposes that similarity is defined as $f(A \cap B) / f(A \cup B)$, Eisler and Ekmanclaim that similarity is proportional to $f(A \cap B) / f(A) + f(B)$, Bush and Mosteller defines similarity as $f(A \cap B) / f(A)$. As such, they differ from Contrast Model by applying a ratio function as opposed to a linear contrast of common and distinctive features [12]. These three models can all be considered specializations of the general equation:

$$sim_{Tversky}(a, b) = \frac{f(A \cap B)}{f(A \cup B) - \alpha f(A - B) - \beta f(B - A)} \quad 4)$$

## Features of Schema

The features of a schema are composed of all the properties of the schema. In section 3, we present the meta-model of schemas, i.e., multi-labeled graph model. A schema can be represented by a graph, where each vertex and edge can be associated with a set of labels describing its properties. Therefore, the feature set of a schema is the vertices and edges of schema and the labels of them (From Definition 2, a schema $\mathcal{S}$ is described by the feature set *descr*($\mathcal{S}$) of all its vertex and edge features). For example, we show the feature sets of schema $\mathcal{S}$ and $\mathcal{T}$ as follows:

1. Features of $\mathcal{S}$: $descr = \{r_{V^{\mathcal{S}}}, r_{E^{\mathcal{S}}}\}$

$r_{V^{\mathcal{S}}} = \{(s_1, \text{Schema } \mathcal{S}), (s_1, \text{schema}), (s_1, \text{Schema});$
$(s_2, \text{AccountOwner}), (s_2, \text{account + owner}), (s_2, \text{ElementType});$
$(s_3, \text{AccountOwner.Name}), (s_3, \text{name}), (s_3, \text{element});\ldots$

($s_{11}$, Address.ZIP), ($s_{11}$, ZIP), ($s_{11}$, element)}

$r_{E^{\mathcal{S}}}$ ={($s_1$, $s_2$, part-of), ($s_2$, $s_3$, part-of), …, ($s_7$, $s_{11}$, part-of)}

2. Features of $\mathcal{T}$: $descr = \{r_{V^{\mathcal{T}}}, r_{E^{\mathcal{T}}}\}$

$r_{V^{\mathcal{T}}}$ ={($t_1$, Schema $\mathcal{T}$), ($t_1$, schema), ($t_1$, Schema);

($t_2$, Customer), ($t_2$, customer), ($t_2$, ElementType); …

($t_{10}$, PostalCode), ($t_{10}$, postal code), ($t_{10}$, element)};

$r_{E^{\mathcal{T}}}$ ={($t_1$, $t_2$, part-of), ($t_2$, $t_3$, part-of), …, ($t_6$, $t_{10}$, part-of)}

## Common features with respect to a matching state

Based on Contrast Model, the similarity of two different schemas $\mathcal{S}$ and $\mathcal{T}$ depends on both the common features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$). Given a matching state $m \in M$, we can calculate the common features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$):

$$
\begin{aligned}
&descr(\mathcal{S}) \cap_{\boldsymbol{m}} descr(\mathcal{T}) \doteq \\
&\quad \left\{ \begin{array}{l} (s_a, l) \in r_{V^{\mathcal{S}}} \,|\, \exists m_{aj} = 1,\ (t_j, l) \in r_{V^{\mathcal{T}}} \\ a \in \{1,\ 2,\ \cdots,\ n\},\ j \in \{1,\ 2,\ \cdots,\ k\} \end{array} \right\} \\
&\cup \left\{ \begin{array}{l} (t_b, l) \in r_{V^{\mathcal{T}}} \,|\, \exists m_{ib} = 1,\ (s_i, l) \in r_{V^{\mathcal{S}}} \\ b \in \{1,\ 2,\ \cdots,\ k\},\quad i \in \{1,\ 2,\ \cdots,\ n\} \end{array} \right\} \\
&\cup \left\{ \begin{array}{l} (s_c, s_{c'}, l) \in r_{E^{\mathcal{S}}} \,|\, \exists m_{cj} = 1,\ \exists m_{c'j'} = 1,\ (t_j, t_{j'}, l) \in r_{E^{\mathcal{T}}} \\ c,\ c' \in \{1,\ 2,\ \cdots,\ n\},\ c \neq c' \end{array} \right\} \\
&\cup \left\{ \begin{array}{l} (t_c, t_{c'}, l) \in r_{E^{\mathcal{T}}} \,|\, \exists m_{ic} = 1,\ \exists m_{i'c'} = 1,\ (s_i, s_{i'}, l) \in r_{E^{\mathcal{S}}} \\ c,\ c' \in \{1,\ 2,\ \cdots,\ k\},\ c \neq c' \end{array} \right\}
\end{aligned}
\qquad 5)
$$

## Splits with respect to a matching state

For SMP, we allow a vertex of $\mathcal{S}$ can be matched with a set of vertex of $\mathcal{T}$, therefore, if given a multivalent mapping *m*, we also have to identify the set of split vertices, i.e., the set of vertices that are mapped to more than one vertex, each split vertex *v* being associated with the set $p_v$ of its mapped vertices:

$$
\begin{aligned}
splits(\boldsymbol{m}) := &\left\{ (s, p_s) \,\middle|\, \begin{array}{l} s \in V^{\mathcal{S}},\ |p_s| \geq 2 \\ p_s = \left\{ t \in V^{\mathcal{T}} \,\middle|\, \begin{array}{l} m_{s,t} = 1 \\ t \in \{1,\ 2,\ \cdots,\ k\} \end{array} \right\} \end{array} \right\} \\
&\cup \left\{ (t, p_t) \,\middle|\, \begin{array}{l} t \in V^{\mathcal{T}},\ |p_t| \geq 2 \\ p_t = \left\{ s \in V^{\mathcal{S}} \,\middle|\, \begin{array}{l} m_{s,t} = 1 \\ s \in \{1,\ 2,\ \cdots,\ n\} \end{array} \right\} \end{array} \right\}
\end{aligned}
\qquad 6)
$$

The more detailed discussions can see [6, 25], and we give an example in section 5.7.2 to explain the reason that we should identify the set of split vertices.

## Similarity function

By Eq.4, let $\alpha = 0$, $\beta = 0$, the similarity of $\mathcal{S}$ and $\mathcal{T}$ with respect to a matching state *m* is defined by:

$$\boldsymbol{sim}_m(\mathcal{S}, \mathcal{T}) = \frac{f(descr(\mathcal{S}) \cap_m descr(\mathcal{T})) - g(splits(\boldsymbol{m}))}{f(descr(\mathcal{S}) \cup descr(\mathcal{T}))} \quad 7)$$

where *f* and *g* are two functions that are introduced to weigh features and splits, depending on the desired application.
Indeed, we can design different similarity function to compare two schemas based on the variants of Contrast Model, i.e., we can obtain the other similarity measures by Eq.4. Here, *f* and *g* are cardinality functions:

$$f(descr(\mathcal{S})) = w_{name} \cdot \left|r_{V^{\mathcal{S}} name}\right| + w_{concept} \cdot \left|r_{V^{\mathcal{S}} concept}\right| + w_{type} \cdot \left|r_{V^{\mathcal{S}} type}\right| + w_E \cdot \left|r_{E^{\mathcal{S}}}\right| \quad 8)$$

$$g(splits(\boldsymbol{m})) = w' \cdot \left|splits(\boldsymbol{m})\right| \quad 9)$$

Finally, the maximal similarity *sim*($\mathcal{S}$, $\mathcal{T}$) of two schemas $\mathcal{S}$ and $\mathcal{T}$ is the greatest similarity with respect to all possible matching states:

$$\boldsymbol{sim}(\mathcal{S}, \mathcal{T}) = \max_{m \subseteq M} \frac{f(descr(\mathcal{S}) \cap_m descr(\mathcal{T})) - g(splits(\boldsymbol{m}))}{f(descr(\mathcal{S}) \cup descr(\mathcal{T}))} \quad 10)$$

The denominator $f(descr(\mathcal{S}) \cup descr(\mathcal{T})$ of Eq.7 does not depend on the matching states, which is introduced to normalize the similarity value to the 0-1 range [6]. Hence, to compute the maximum similarity between two graphs $\mathcal{S}$ and $\mathcal{T}$, one has to find the matching state *m* that maximizes the score function:

$$score(\mathcal{S}, \mathcal{T}) = f(descr(\mathcal{S}) \cap_m descr(\mathcal{T})) - g(splits(m)) \quad 11)$$

## The Best Matching State Based on Contrast Model

Based on the similarity principles of Contrast Model, for multi-labeled graph, the more common features a matching state *m* has, the better the matching state is. Therefore, we propose the concept of best matching result:

**Definition 7.** (*The best matching state*) Suppose that $\mathcal{S} = (V^{\mathcal{S}}, E^{\mathcal{S}}, Lab^{\mathcal{S}}, r_{V^{\mathcal{S}}}, r_{E^{\mathcal{S}}})$ is the source schema, $\mathcal{T} = (V^{\mathcal{T}}, E^{\mathcal{T}}, Lab^{\mathcal{T}}, r_{V^{\mathcal{T}}}, r_{E^{\mathcal{T}}})$ is the target schema, there exist a best matching state *m*, such that:

$$\boldsymbol{sim}_m(\mathcal{S}, \mathcal{T}) \geq \boldsymbol{sim}_{m'}(\mathcal{S}, \mathcal{T}),\ m \in M,\ m' \in M,\ m \neq m'$$

where, $\boldsymbol{sim}_m(\mathcal{S}, \mathcal{T})$ and $\boldsymbol{sim}_{m'}(\mathcal{S}, \mathcal{T})$ can be computed by Eq.7.

In other words, the best matching state is one that maximizes the common features and minimizes the distinctive features.

## The algorithmic model of SMP

Based on Eq.10, we have the optimization function for the multi-labeled graph matching, then, we present the algorithmic model for SMP in Fig. 4. The purpose of the algorithm is to find the best matching state between two schemas.

**Input:** Schemas $\mathcal{S}$ and $\mathcal{T}$
**Object:** Find a matching state that maximize the similarity of graph $G_1$ and $G_2$, namely, to find the best matching state
**Output:** The semantic correspondences between $\mathcal{S}$ and $\mathcal{T}$
1. $G_1$ = Multi_labeled_Graph($\mathcal{S}$ ); $G_2$ = Multi_labeled_Graph ($\mathcal{T}$);
2. Iteratively search the matching space to find a matching state $m$, such that similarity$(G_1,G_2)_m$ is the maximum one among the matching states (Eq.10);
3. $m$ is the matching result of $G_1$ and $G_2$.

*Fig. 4 Algorithm model of SMP based on Multi-labeled graph matching*

## Examples

To illustrate the similarity computation, we take a matching state for example. For two schemas in Fig. 1, based on Fig. 2 and Table 2, we can obtain all the features of two schemas:

$$descr(\mathcal{S}) \cup descr(\mathcal{T}) = descr(\mathcal{S}) \cup descr(\mathcal{T})|_{concept} \cup descr(\mathcal{S}) \cup descr(\mathcal{T})|_{name} \cup descr(\mathcal{S}) \cup descr(\mathcal{T})|_{type} \cup descr(\mathcal{S}) \cup descr(\mathcal{T})|_{edge}$$

$$f(descr(\mathcal{S}) \cup descr(\mathcal{T})) = |r_{V\,name}| + 4 \cdot |r_{V\,concept}| + |r_{V\,type}| + |r_E| = 21 + 4 \times 21 + 21 + 19 = 145,$$

where, $w_{concept} = 4$, $w_{name} = 1$, $w_{type} = 1$, $w_E = 1$.

*Remark*: For Eq.8, $w_{concept} = 4$, because the semantic matching is stronger than other label matchings [24].

Suppose that there is a mapping state of $\mathcal{S}$ and $\mathcal{T}$:

$$m_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e., $m_1$= {($s_1$, $t_1$), ($s_2$, $t_2$), ($s_3$, $t_3$), ($s_3$, $t_4$), ($s_4$, $t_5$), ($s_7$, $t_6$), ($s_8$, $t_7$), ($s_9$, $t_8$), ($s_{10}$, $t_9$), ($s_{11}$, $t_{10}$)}.

### *Similarity of Matching State*

#### 1. Common Features of Matching State

At first, we calculate the common features between two schemas based on $m_1$.
*a*. To compare the name labels of schemas to obtain the name features. There are many methods to measure similarity of names [7, 13, 19]. For example we use the Levenshtein distance (i.e., edit distance) to compare the name string of vertices [7], $sim_{name}(s_1, t_1) = 0.875$. Suppose the threshold of name matching $th_{name}$ = 0.4, the name common features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$):

$descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{name}$ = {($s_1$, Schema $\mathcal{S}$), ($s_2$, AccountOwner), ($s_3$, AccountOwner.Name), ($s_4$, AccountOwner.Address), ($s_7$, Address), ($s_8$, Address.Street), ($s_9$, Address.City), ($t_1$, Schema $\mathcal{T}$), ($t_2$, Customer), ($t_3$, Customer.Cfname), ($t_4$, Customer.Clname), ($t_5$, Customer.CAddress) ($t_6$, CustomerAddress), ($t_7$, CustomerAddress.Street), ($t_8$, CustomerAddress.City)}

*b*. To obtain the common concept features, we need compare the concept labels of two vertices. We can use some semantic distances to compare similarity of two concepts, such as *hso*, *wup*, *res*, *lin*, and *jcn*, etc [17]. By *wup* [22], $sim_{concept}(s_2, t_2) = 0.67$. If $th_{concept}$ = 0.55, for $m_1$, the intersection features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$):

$descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{concept}$ = {($s_1$, schema), ($s_2$, account + owner), ($s_3$, name), ($s_4$, address), ($s_7$, address), ($s_8$, street), ($s_9$, city), ($s_{10}$, state), ($s_{11}$, ZIP), ($t_1$, schema), ($t_2$, customer), ($t_3$, first name), ($t_4$, last name), ($t_5$, address) ($t_6$, address), ($t_7$, street), ($t_8$, city), ($t_9$, province), ($t_{10}$, postal code)}

Unlike the semantic matching method proposed by Giunchiglia *et al*. [11], our method based on the structural relations of two concepts in WordNet. The semantic matching result of two elements is a real value between 0 and 1.

*c*. The common type features of two schemas:

$descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{type}$ = {($s_1$, Schema), ($s_2$, ElementType), ($s_3$, element), ($s_4$, element), ($s_7$, ElementType), ($s_8$, element), ($s_9$, element), ($s_{10}$, element), ($s_{11}$, element), ($t_1$, Schema), ($t_2$, ElementType), ($t_3$, element), ($t_4$, element), ($t_5$, element) ($t_6$, ElementType), ($t_7$, element), ($t_8$, element), ($t_9$, element), ($t_{10}$, element)}

*d*. The common edge features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$):

$descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{edge}$ = {($s_1$, $s_2$, part-of), ($s_2$, $s_3$, part-of), ($s_2$, $s_4$, part-of), ($s_1$, $s_7$, part-of), ($s_7$, $s_8$, part-of), ($s_7$, $s_9$, part-of), ($s_7$, $s_{10}$, part-of), ($s_7$, $s_{11}$, part-of), ($t_1$, $t_2$, part-of), ($t_2$, $t_3$, part-of), ($t_2$, $t_4$, part-of), ($t_2$, $t_5$, part-of), ($t_1$, $t_6$, part-of), ($t_6$, $t_7$, part-of), ($t_6$, $t_8$, part-of), ($t_6$, $t_9$, part-of), ($t_6$, $t_{10}$, part-of)}

*e*. If we use all of the vertex and edge features of schemas together, we can get the intersection features of *descr*($\mathcal{S}$) and *descr*($\mathcal{T}$) as follows:

$$descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T}) = descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{concept} \cup descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{name} \cup descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{type} \cup descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})|_{edge}$$

$$f(descr(\mathcal{S}) \cap_{m_1} descr(\mathcal{T})) = 15 + 4 \times 19 + 19 + 17 = 127$$

**2. Splits of Matching State**
The matching state $m_1$ has the following splits:

$$splits(m_1) = \{(s_3, \{t_3, t_4\})\} \quad g(splits(m_1)) = | splits(m) | = 3$$

**3. Score and Similarity of Matching State**
By Eq.11, the score of $\mathcal{S}$ and $\mathcal{T}$ based on $m_1$ is:

$$score_{m_1}(\mathcal{S}, \mathcal{T}) = (127 - 3) = 124$$

Then, by Eq.10, the similarity of two schemas $\mathcal{S}$ and $\mathcal{T}$ based on $m_1$ is:

$$Sim_{m_1}(\mathcal{S}, \mathcal{T}) = (127 - 3)/145 = 0.855$$

### *Comparison of Two Matching States*

Now, we expound the reason why we calculate the splits.
Suppose that we add a matching couple ($s_7$, $t_5$) into $m_1$, then we obtain the matching state: $m_2$= {($s_1$, $t_1$), ($s_2$, $t_2$), ($s_3$, $t_3$), ($s_3$, $t_4$), ($s_4$, $t_5$), ($s_7$, $t_5$), ($s_7$, $t_6$), ($s_8$, $t_7$), ($s_9$, $t_8$), ($s_{10}$, $t_9$), ($s_{11}$, $t_{10}$)}. If we do not consider the splits of $m_2$, the common features of $m_2$ more than $m_1$:

$$f(descr(\mathcal{S}) \cap_{m_2} descr(\mathcal{T})) = 16 + 4 \times 20 + 19 + 17 = 132$$

Therefore, the similarity of $m_2$ is higher than the similarity of $m_1$. However, if we compute the splits of the matching states, then we get:

$$splits(m_2) = \{(s_3, \{t_3, t_4\}), (s_7, \{t_5, t_6\}), (t_5, \{s_4, s_7\})\}$$

$$Sim_{m_1}(\mathcal{S}, \mathcal{T}) = (132 - 9)/145 = 0.848$$

We can see that, although the common features of $m_2$ is greater than $m_1$, the similarity of $m_2$ is lower than $m_1$, because the splits is greater than $m_1$. In fact, we have the matching state $m_1$ is better than $m_2$.

## Hybrid Search algorithm for Multi-labeled Graph Matching

### Complete Search

As we known it, to compute the maximum similarity of labeled graphs, which is highly combinatorial, and is a NP-hard problem [2, 20]. It can be explored in an exhaustive way with a "branch and bound" approach, which is an algorithmic technique to find the optimal solution by keeping the best solution found so far. If a partial solution cannot improve on the best, it is abandoned. Such a complete approach is actually tractable if there exists a "good" bounding function that can detect as soon as possible when a node can be pruned [6], i.e., when the score of

all the matching states that can be constructed from the current state is worse than the best score found so far.
In Definition 5, we present the concept of matching matrix, and all of these matching matrixes are the matching states that constitute the matching space. Suppose the current matching state is $m$, all the matching states $m'$ such that $|m' - m| \geq 0$ are the potential successors of $m$, and $m'$ is the superset of $m$, $m' \supseteq m$.
The score function (Eq.11) is not monotonic with respect to set inclusion, i.e., the score of a mapping may either increase or decrease when one adds a new couple to it [6]. Indeed, this score is defined as a difference between a function of the common features and a function of the splits, and both sides of this difference may increase when adding a couple to a mapping. In [6], Champin and Solnon study the bounding function, for every matching state $m'$, $m' \supseteq m$:

$$score_{m'}(\mathcal{S}, \mathcal{T}) = f(descr(\mathcal{S}) \cap_{m'} descr(\mathcal{T})) - g(splits(m'))$$
$$\leq f(descr(\mathcal{S}) \cup descr(\mathcal{T})) - g(splits(m))$$

If $f(descr(\mathcal{S}) \cup descr(\mathcal{T})) - g(splits(m))$ is smaller or equal to the score of the best matching state $m_{best}$ found so far, then the search path can be pruned. In other words, all the supersets of $m$ will not be explored as their score cannot be higher than the best score found so far.
Obviously, although the branch and bound approach can find the best matching state, it is a costly search process. Therefore, we want to find other approximate methods to solve schema matching. Local search has been applied to solve NP-hard optimization problems [1, 20, 25]. The principle of local search is to refine a given initial solution point in the solution space by searching through the neighborhood of the solution point. However, the performance of local search relies on the initial state, we need obtain a good initial state for local search. As a result, we first use a greedy matching algorithm to find a good initial state.

## Greedy Matching Algorithm

The greedy strategy is a fundamental technique to solve NP-hard problem [23]. Based on [6], Zhang *et al*. [25] designed a greedy algorithm to solve SMP: iteratively picks the couple that most increase the score function and has the greatest looked-ahead common edge features. The algorithm stops iterating when every couple neither directly increases the score function nor has looked-ahead common edge features. Now, we discuss the computation of incremental score and the greedy strategies in detail.
Here, the functions $f$ and $g$ are the cardinality functions, so we define the increment of common features as follows:

$$\Delta descr_m(s_i, t_j) = f(descr(\mathcal{S}) \cap_{m \cup (s_i, t_j)} descr(\mathcal{T})) - f(descr(\mathcal{S}) \cap_m descr(\mathcal{T})) \quad 12)$$

$$\Delta descr_m(s_i,\ t_j) \le \begin{cases} e+e & \sum_{c=1}^{n} m_{i,c} = 1,\ \sum_{c=1}^{k} m_{i,c} = 1 \\ 0+e & \sum_{c=1}^{n} m_{i,c} \ge 2,\ \sum_{c=1}^{k} m_{i,c} = 1 \\ e+0 & \sum_{c=1}^{n} m_{i,c} = 1,\ \sum_{c=1}^{k} m_{i,c} \ge 2 \\ 0+0 & \sum_{c=1}^{n} m_{i,c} \ge 2,\ \sum_{c=1}^{k} m_{i,c} \ge 2 \end{cases} \quad 13)$$

where, $e$ is the number of vertex and edge features (i.e., the number of vertex and edge labels), $e = w_{name} + w_{concept} + w_{type} + w_E$ . For the example in section 5.7, $e = 1+4+1+1 = 7$ . If all the features of $s_i$ and $t_j$ are all matched, the $\Delta descr(s_i,\ t_j) = e+e$ ; If partial of features of $s_i$ and $t_j$ are matched, then $\Delta descr(s_i,\ t_j) \le e+e$ .

At the same time, the splits will increase with $(s_i,\ t_j) \in V^S \times V^T$ enter $m$. We evaluate the increment of splits as follows:

$$\Delta splits_{\boldsymbol{m}}(s_i,\ t_j) = g(splits(\boldsymbol{m} \cup (s_i,\ t_j))) - g(splits(\boldsymbol{m})) \quad 14)$$

Since $g$ is the cardinality function, we show the evaluation of splits in Eq.15:

$$\Delta splits_m(s_i,\ t_j) = \begin{cases} 0 & \sum_{c=1}^{k} m_{i,c} = 1,\ \sum_{c=1}^{n} m_{c,j} = 1,\ m_{i,j} = 1 \\ 3 & \sum_{c=1}^{k} m_{i,c} = 2,\ \sum_{c=1}^{n} m_{c,j} = 1,\ m_{i,j} = 1 \\ 3 & \sum_{c=1}^{k} m_{i,c} = 1,\ \sum_{c=1}^{n} m_{c,j} = 2,\ m_{i,j} = 1 \\ 1 & \sum_{c=1}^{k} m_{i,c} > 2,\ \sum_{c=1}^{n} m_{c,j} = 1,\ m_{i,j} = 1 \\ 1 & \sum_{c=1}^{k} m_{i,c} = 1,\ \sum_{c=1}^{n} m_{c,j} > 2,\ m_{i,j} = 1 \\ 2 & \sum_{c=1}^{k} m_{i,c} > 2,\ \sum_{c=1}^{n} m_{c,j} > 2,\ m_{i,j} = 1 \end{cases} \quad 15)$$

Based on Eq.9 and Eq.11, we get the incremental score:

$$\Delta score_m(s_i,\ t_j) = \Delta descr_m(s_i,\ t_j) - \Delta splits_m(m \cup (s_i,\ t_j)) \quad 16)$$

As the complete search methods are not feasible, by Eq.16, we can design the greedy search strategies:

1. *The matching candidates are the zero elements in the current matching state* $m$. *If* $m_{i,j} = 0$ , *and* $\Delta score_m(s_i,\ t_j)$ *is the maximum one among the matching candidates*, *we will let* $m_{i,j} = 1$, *i.e.*, $(s_i,\ t_j)$ *will enter* $m$.

2. *In addition*, *if there are several matching candidates have the maximum score value*, *then we will choose the one which has more potential common edge features*, *i.e.*, *look_ahead*$(s_i,\ t_j)$ *is the maximal one*.

$$look\_ahead(s_i,\ t_j) = \{(s_i,\ s,\ l)\ \in r_{E^{\mathcal{S}}} \mid \exists t \in V^{\mathcal{T}},\ (t_j,\ t,\ l) \in r_{E^{\mathcal{T}}}\} \cup \{(t_j,\ t,\ l) \in r_{E^{\mathcal{T}}} \mid \exists s \in V^{\mathcal{S}},\ (s_i,\ s,\ l) \in r_{E^{\mathcal{S}}}\} \cup \{(s,\ s_i,\ l)\ \in r_{E^{\mathcal{S}}} \mid \exists t \in V^{\mathcal{T}},\ (t,\ t_j,\ l) \in r_{E^{\mathcal{T}}}\} \cup \{(t,\ t_j,\ l) \in r_{E^{\mathcal{T}}} \mid \exists s \in V^{\mathcal{S}},\ (s,\ s_i,\ l) \in r_{E^{\mathcal{S}}}\} - descr(\mathcal{S}) \cap_{m \cup \{(s_i,\ t_j)\}} descr(\mathcal{T}) \quad 17)$$

*The look_ahead means that this matching couple will increase the common edge features*, *therefore*, *we pick the one has the greatest looked-ahead value*.

3. *If there are several matching couples both have the maximum score and look_ahead value*, *then the algorithm can pick a matching couple randomly in the matching candidates*.

In [25], the authors used an example to present the greedy matching process in detail.

For this greedy algorithm, the computations of the *f* function has a polynomial time complexity of $O((|V^{\mathcal{S}}| \times |V^{\mathcal{T}}|)^2)$; *g* functions has a linear time complexities with respect to the size of the schemas $O(\mathrm{Max}(|V^{\mathcal{S}}|,|V^{\mathcal{T}}|)$; The computation of "*look ahead*" sets has a polynomial time complexity of $O(|V^{\mathcal{S}}| \times |V^{\mathcal{T}}|)$, and can be computed in an incremental way [6]. Therefore, the greedy algorithm has a polynomial time complexity of $O((|V^{\mathcal{S}}| \times |V^{\mathcal{T}}|)^2)$.

Just as the greedy algorithm of knapsack problem, at each step of search process, the greedy algorithm only iteratively selects the matching couple which make the score function is the maximum one. The algorithm only tracks one search path and do not compare with matching results that obtained by the other search paths. Therefore, the greedy algorithm is not a complete algorithm. Based on [25], We design a local search algorithm to improve the matching result.

## Local Search for Multi-labeled Graph Matching

Local search is class of effective approximation algorithms for combinatorial optimization [1], which tries to improve a solution by locally exploring its neighborhood. The neighbors of *m* are the mapping states that can be obtained by adding or removing one couple of vertices to *m*, $\forall m \in M$. The size of *N*(*m*) is $C^1_{n \times k} = n \times k$.

$$N(m) := \left\{ m' \left| |m' - m| = \sum_{i=1}^{i=n} \sum_{j=1}^{j=k} |m'_{i,j} - m_{i,j}| \le 1,\ m' \in M \right. \right\} \quad 18)$$

If we define *l* elements of current state *m* can be changed at the same time, then we obtain the neighborhood:

$$N_l(m) := \left\{ m' \left| |m' - m| = \sum_{i=1}^{i=n} \sum_{j=1}^{j=k} |m'_{i,j} - m_{i,j}| \le l,\ m' \in M \right. \right\} \quad 19)$$

Therefore, the size of $N_l(m)$ is $C_{n\times k}^1 + C_{n\times k}^2 + \cdots + C_{n\times k}^l$. Fig. 5 shows the local search algorithm, which randomly select a matching couple in $N(m)$.

```
function Local_Search(𝒮, 𝒯)
begin
  k ← 0;
  m ← Greedy(𝒮, 𝒯) ;
  M ← N(m) ; m_best ← m ;
  while ( k++ < Max_iteration ) do /* search */
    if sim_m(𝒮, 𝒯) > sim_{m_best}(𝒮, 𝒯) then
      m_best ← m ;
    end if
    choose randomly m ∈ M ;
    M ← M − m ;
    if M = ∅ then return m_best ;
  end
  return m_best ;
end
```

*Fig. 5 Local Search for Schema Matching*

Provided that we want to obtain all the feasible mapping states for users, we can set a threshold of schema similarity to obtain the matching states which greater than the given similarity value. For instance, given a matching state *m*, if $sim_m(\mathcal{S}, \mathcal{T}) \geq th_{sim}$, then *m* is a possible matching result.

# Evaluation of Algorithm

## Experimental Design

We have carried out some experiments to evaluate our approach. We tested the hybrid algorithm on seven samples: Biztalk (Fig. 1), Library (XML) [14], University (XML) [14], Property Listing (XML) [14], Purchase order (relational & XML) [8], Financial (XML) [27], Student (XML) [27]. The seven schemas are classified into three different kinds:

1. matching of XML schemas (Biztalk, Library, Property Listing, Financial, Student)

2. matching of XML schemas using XML data instances (University)

3. matching of relational schema and XML schemas (Purchase order)

For these samples, Table 3 shows the scales of them, includes the numbers of vertex, edge, and the depths of schema structure:

| | Scale of $\mathcal{S}$ | | | Scale of $\mathcal{T}$ | | |
|---|---|---|---|---|---|---|
| | vertex | edge | depth | vertex | edge | depth |
| Biztalk | 11 | 10 | 3 | 10 | 9 | 3 |
| Library | 15 | 14 | 3 | 16 | 15 | 3 |
| University | 10 | 9 | 3 | 7 | 6 | 3 |
| Property | 12 | 11 | 4 | 13 | 12 | 4 |
| Purchase | 13 | 12 | 2 | 9 | 8 | 3 |
| Financial | 14 | 13 | 3 | 14 | 13 | 5 |
| Student | 18 | 17 | 4 | 15 | 14 | 6 |

*Table 3 The number of vertex and edge*

## Parameter Tuning

Because schema matching is a heuristic operation, we should use some meta-strategies to change the parameters of similarity evaluation function during the search process. The optimization function (Eq.10) is very important to achieve the optimal matching result. To obtain the optimal matching results, we should adjust the functions *f* and *g*, i.e., tune different weights of Eq.8 and Eq.9, including the weight of splits *w'*, the weight of vertex $w_{name}$, $w_{concept}$ and $w_{type}$, and edge features $w_E$.

First, for function *f*, the concept feature has greater weight than name and type feature. If the concepts of two vertices are matchable, then the matching probability of two vertices is higher than only name or type matchable. For example, if $w_{concept} = 4$, $w' = 1$, the greedy algorithm will obtain the couples ($s_3$, $t_3$) and ($s_3$, $t_4$), however, if $w_{concept} \le 2$, $w' = 1$, the algorithm cannot obtain both ($s_3$, $t_3$) and ($s_3$, $t_4$), and will obtain ($s_5$, $t_4$). Second, the function *g* determines the number of multivalent mappings. The greater weight of *g* is, the more difficult to obtain multivalent mapping. For example, if $w_{concept} = 4$, $w' = 3$, then the algorithm cannot obtain *many-to-many* matching, and will obtain *one-to-one* mapping result.

By different weights, we obtain a reasonable proportion of functions of *f* and *g*, and then we can obtain the desired multivalent correspondences. Table 4 shows the weights of *f* and *g* that are used in greedy and local search algorithms.

| Schema | $w_{na}$ | $w_{concept}$ | $w_{type}$ | $w_E$ | $w'$ |
|---|---|---|---|---|---|
| Biztalk/Purchase order | 1 | 4 | 1 | 1 | 1 |
| Library/Financial | 1 | 4 | 2 | 1 | 1.5 |
| University/Property | 1 | 4 | 1 | 2 | 1.5 |
| Student | 1 | 4 | 2 | 1 | 1 |

*Table 4. The weights of function f and g*

In fact, we only consider the three features of schemas (i.e., name, concept, and type) in this paper, and in the near future, we can consider the more features of schemas for matching. Therefore, we should modify the function *f* and *g*, and adjust the weights of different features.

The similarity measure of schemas based on Contrast Model and the multi-labeled graph is an open framework for schema matching. In light of different applications, we can choose appropriate features to describe schemas and encode these features as the labels of multi-labeled graph, then, we can use the hybrid matching method to obtain the desired matching result.

## Experimental Results

We evaluate the "accuracy" of the algorithm by counting the number of needed adjustments, therefore, in this paper, provided that the best matching state of two schemas is fixed in the matching tests. Under these conditions, the performance of hybrid search algorithm is very well.
Fig. 6 shows the average Precision of seven matching samples by the hybrid algorithm, the total average Precision is 87%.

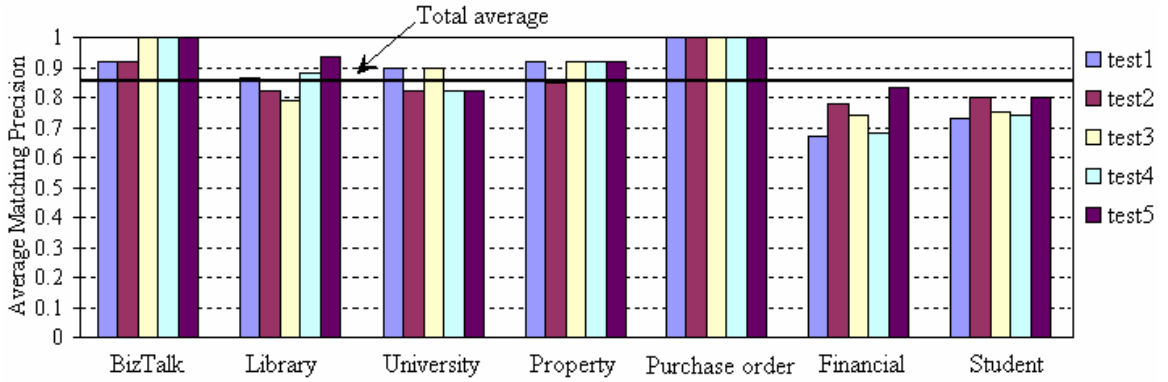


*Fig. 6 Average Precision of matching samples by hybrid search*

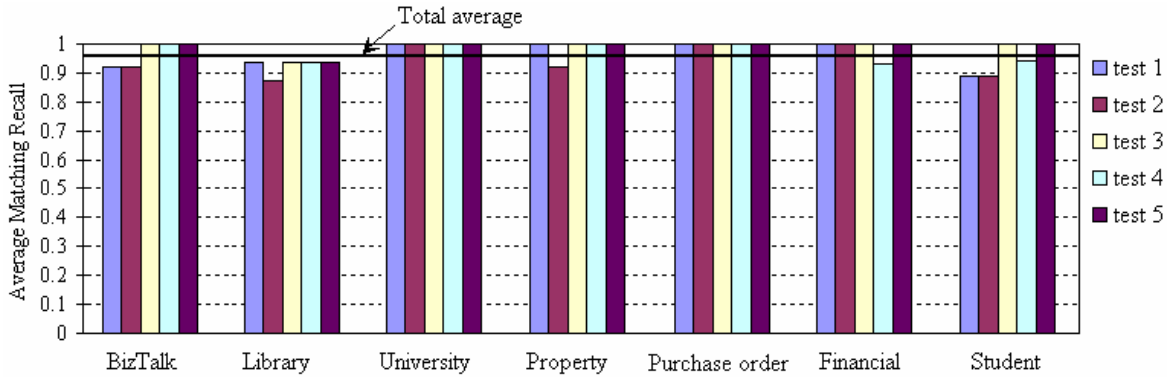


*Fig. 7 Average Recall of matching samples by hybrid search*

Moreover, the algorithm achieves the total average Recall nearly of 97.2% (See Fig. 7), and total average Overall of 83.6%. Fig. 8 presents average Precision, Recall, and Overall of samples by the hybrid algorithm.

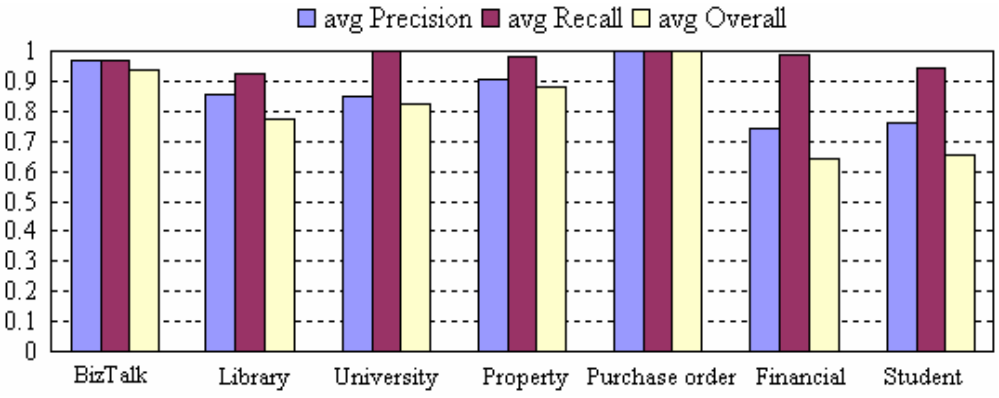


*Fig. 8 Average quality of matching samples by hybrid search*

The hybrid algorithm has been implemented in Visual C++. Experiment settings: P4 2.4G, 224M DDR RAM. On these tests, the algorithm is very fast and obtains feasible matching results.
For seven samples, Table 5 shows iteration times and average running time of greedy algorithm.

| | Biztalk | Library | University | Property | Purchase order | Financial | Student |
|---|---|---|---|---|---|---|---|
| Iteration | 12 | 20 | 10 | 21 | 9 | 18 | 25 |
| Time (s) | 0.310 | 0.712 | 0.045 | 0.325 | 0.312 | 0.452 | 0.798 |

*Table 5. The average iterations of greedy algorithm*

Table 6 shows the total average running times of hybrid algorithm, where, the maximum iteration of local search is 5000, i.e., Max_iteration = 5000 (see Fig. 5).

| | Biztalk | Library | University | Property | Purchase order | Financial | Student |
|---|---|---|---|---|---|---|---|
| Time (s) | 0.680 | 1.846 | 0.357 | 1.003 | 0.640 | 1.738 | 1.873 |

*Table 6. The total average times of hybrid search algorithm*

The similarity measure of multi-labeled graph can combine all the properties and features of two schemas, especially the matching method considers the edge features between two schemas, therefore, the matching performance is better than existed prototypes. To compare with Similarity Flooding [14], Automatch [3], and LSD [10], the average "accurate" is higher than these matching methods. In particular, for Biztalk, Library, University, and Property Listing, the matching results are better than Similarity Flooding. In addition, the time cost of our algorithm is lower than other algorithms.
By the multi-labeled graph matching, the algorithm model can implement not merely instance-level matching (University), but also schema-level matching (Biztalk, etc.). Moreover, by our matching method, the users can obtain element-level *n*:*m* matching result. If we tune the parameters of optimization function, the matching algorithm can obtain different matching results for users.

### A Comparison between Greedy Matching and Hybrid Search

We use an experiment to compare the performance of greedy matching and hybrid search. For seven samples in section 7.1, Fig. 9 shows the average Precision, Recall, and Overall of samples by using greedy matching algorithm. To compare with the matching result of hybrid algorithm (see Fig. 8), we can see that the quality of matching results is improved by local search. The comparison graph between greedy matching and hybrid search is shown in Fig. 10.

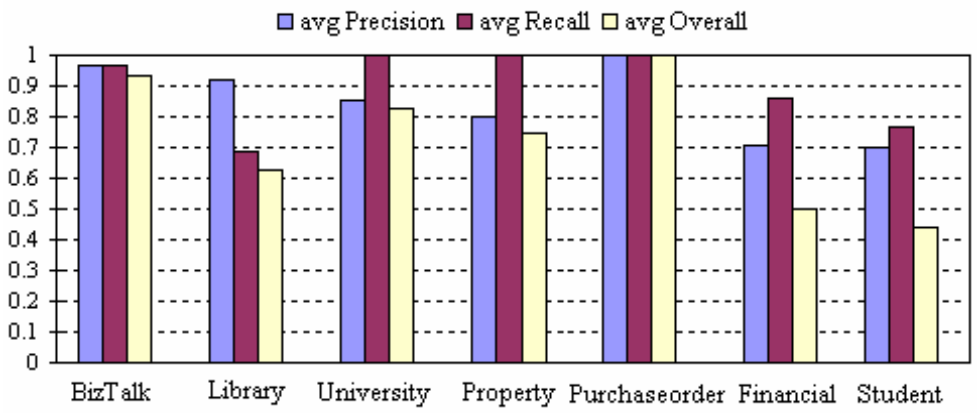


*Fig. 9 Average quality of matching samples by greedy matching*

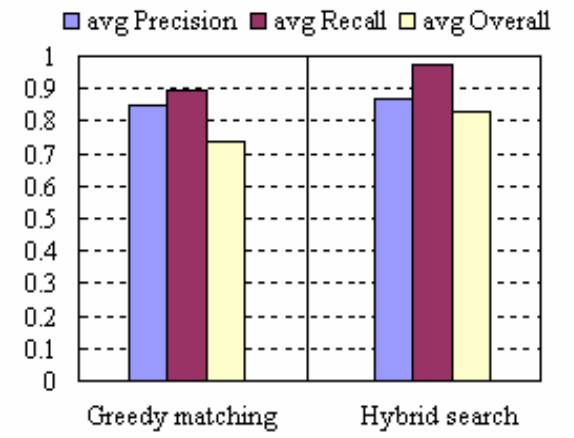


*Fig. 10 Comparison of greedy matching and hybrid search*

## Comparative Results and Conclusions

### Comparative Results

In Table 7, we compare the characteristics of five published matching methods with our Multi-labeled Graph Matching method.

| | Similarity Flooding [14] | LSD [10] | Cupid [13] | COMA [8] | S-Match [11] | Multi-labeled Graph Matching |
|---|---|---|---|---|---|---|
| Tested schema types | XML, relational | XML | XML, relational | XML | XML | XML, relational |
| Metadata representation | Directed labeled graph | XML schema trees | extended ER | DAG | tree | Multi-labeled graph |
| Match granularity | element / structure level | element / structure level | element / structure level | | element / structure level | element / structure level |
| Match cardinality | n:m | 1:1 | 1:1 and n:1 | 1:1 | 1:1 | n:m |
| Combination of matchers | hybrid | Composite matcher with automatic combination of matcher results | hybrid | hybrid, composite | semantic matcher | hybrid |
| Manual work / user input | user can adjust threshold weights | user-supplied matches for training sources; user can specify tuning parameters and integrity constraints to guide selection of match candidates | user can adjust threshold weights | user-Feedback matcher to capture match and mismatch information provided by the user including corrected match results from the previous match iteration. | - | user can adjust weights of objective function, threshold weights, choose intial matching candidates |
| Schema level match | syntactic | syntactic | syntactic | syntactic | semantic | syntactic / semantic |
| Instance level matchers | syntactic | syntactic | - | - | - | syntactic / semantic |
| Reuse / auxiliary information used | thesauri, glossaries | comparison with training matches; lookup for valid domain values | thesauri, glossaries | reuse, thesauri, glossaries | WordNet, thesauri, glossaries | WordNet, thesauri, glossaries |
| Pre-match effort | - | training, specifying domain synonyms, constraints | specifying domain synonyms | specifying domain synonyms | WordNet | specifying domain synonyms, WordNet |
| Subjectivity | 7 users | 1 user | 1 user | 1 user | 1 user | 5 users |
| Application area | metadata management | data integration with pre-defined global schema | data translation, but intended to be generic | integration of web data sources, data warehouse loading and XML message mapping | semantic integration | XML message mapping, semantic integration |
| Employed quality measures | Overall | Recall | - | Precision, Recall, Overall | Precision, Recall, Overall, F-measure | Precision, Recall, Overall |
| Precision | - | ~0.8 | - | 0.93 | ~1.0 | ~0.87 |
| Recall | - | 0.8 | - | 0.89 | ~0.88 | ~0.97 |
| Overall | ~0.6 | ~0.6 | - | 0.82 | ~0.88 | ~0.83 |
| F-measure | | | | | ~0.94 | |

*Table 7. Characteristics of proposed schema match approaches*

## Conclusion and Future work

In this paper, we focus on how to formulize SMP as a combinational optimization problem, and study the approximate matching algorithm to solve this optimization problem. We show the definition of multi-labeled graph at first. Therefore, we can

transform SMP into a multi-labeled graph matching problem. We present a similarity measure of multi-labeled graph based on Contrast Model, and we propose the best matching result based on features of two schemas. Then, by the objective function of multi-labeled graph matching (Eq.10), we discuss the branch and bound method briefly, which is a complete algorithm for SMP. Because it is a costly method, we propose a hybrid matching algorithm to solve graph matching problem, which combine the greedy matching algorithm and local search together. The experimental results confirm that the hybrid algorithm is effective.
In fact, we mainly use three kinds of label features, i.e., name, concept, type, and one kind of relation, i.e., part-of. Nevertheless, the other features also can be labeled to vertices and edges of multi-labeled graph. Our matching method also solves the extended multi-labeled graph effectively and easily. The multi-labeled graph model can integrate all of available features of schemas flexibly. Therefore, at first, we will use all features together to obtain more accurate matching result, such as data types and value ranges, uniqueness, optionality, relationship types and cardinalities, etc. Secondly, we will design some meta-heuristic strategies to tune the weights of functions during the search process. We also can introduce the fuzzy strategies to adjust the weights of Eq.8 and Eq.9. So we can find a desired matching state fast and accurately. Thirdly, for large-scale schema matching (XML, relational schema, etc.), we will design sub-labeled graph matching methods. We can use schema segmentation to obtain subschemas at first, and then use subgraph matching algorithms to achieve subschema matching. Moreover, we are going to design incremental algorithms for large-scale schema matching, and design reuse framework for schema matching based on CBR model.

## Acknowledgment

This work was supported by the National 973 Information Technology and High-Performance Software Program of China under Grant No.G1998030408.

## References

[1] E. Aarts, J. K. Lenstra. Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, 1997.
[2] E. Bengoetxea. Inexact Graph Matching Using Estimation of Distribution Algorithms. Ecole Nationale Supérieure des Télécommunications. 2002, PhD thesis.
[3] J. Berlin, A. Motro, Database Schema Matching Using Machine Learning with Feature Selection. LNCS 2348: 452-466.
[4] V. Blondel, L. Ninove, P. V. Dooren, Convergence of graph similarity algorithms, Proceedings of the 23rd Benelux Meeting on Systems and Control, Helvoirt, The Netherlands, paper FrP06-4, March 17-19, 2004.
[5] P. Bouquet, B. Magnini, L. Serafini, S. Zanobini, A SAT-Based Algorithm for Context Matching. LNAI 2680: 66-79.
[6] P. A. Champin, C. Solnon. Measuring the similarity of labeled graphs. Springer-Verlag, ICCBR 2003, LNAI 2689: 80–95.
[7] W. W. Cohen, P. Ravikumar, S. E. Fienberg, A Comparison of String Distance Metrics for Name-Matching Tasks. 2003, IJCAI-03: 3-78.

[8] H. H. Do, E. Rahm. COMA - A system for flexible combination of schema matching approaches. VLDB 2002.
[9] H. H. Do, S. Melnik, E. Rahm. Comparison of schema matching evaluations. LNCS 2693: 221–237.
[10] A. Doan, P. Domingos, A. Halevy, Learning to Match the Schemas of Data Sources: A Multistrategy Approach. Machine Learning. Kluwer Academic Publishers, 2003 (60):279-301.
[11] F. Giunchiglia, P. Shvaiko, M. Yatskevich. S-Match: An algorithm and an implementation of semantic matching. In Proceedings of ESWS'04.
[12] R. L. Goldstone. Similarity. MIT encylopedia of the cognitive sciences. Cambridge, MA: MIT Press, 763-765.
[13] J. Madhavan, P. A. Bernstein, E. Rahm, Generic Schema Matching with Cupid. 27th VLDB Conference.
[14] S. Melnik, Generic model management - concepts and algorithms, Springer, 2004, LNCS 2967.
[15] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm. ICDE 2002.
[16] R. J. Miller, L.M. Haas, M.A. Hernández, Clio: Schema Mapping as Query Discovery. VLDB 2000.
[17] T. Pedersen, S. Patwardhan, S. Patwardhan, WordNet::Similarity - Measuring the Relatedness of Concepts. Proceedings of the Nineteenth National Conference on Artificial Intelligence, 2004, San Jose, CA.
[18] E. Rahm, P. A. Bernstein, On matching schemas automatically. Microsoft Research, Redmon, WA. Technical Report MSR-TR-2001-17, 2001.
[19] E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching. The VLDB Journal, 2001(10):334-350.
[20] S. Sorlin, C. Solnon: Reactive Tabu Search for Measuring Graph Similarity. GbRPR 2005: 172-182.
[21] A. Tversky. Features of similarity. Psychological Review, 84, 327-352.
[22] Z. B. Wu, M. Palmer. Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994, 133-138.
[23] V. V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin, 2001.
[24] Z. Zhang, H. Y. Che, P. F. Shi, Y. Sun, J. Gu. An algebraic framework for schema matching. WAIM 2005, LNCS 3739.
[25] Z. Zhang, H. Y. Che, P. F. Shi, Y. Sun, J. Gu. Multi-labeled graph matching - An algorithm model for schema matching. ASIAN'05.
[26] Z. Zhang, H. Y. Che, P. F. Shi, Y. Sun, J. Gu. Schema homomorphism – An algebraic framework for schema matching. ASIAN'05.
[27] http://www.almaden.ibm.com/software/km/clio/clioxmldemo.shtml

Zhi Zhang received the B.S. and M.S. degrees in Mechanical Engineering from Sichuan University in 1998 and Southwest Jiaotong University in 2002, respectively. He is currently working towards the Ph.D. degree in Pattern Recognition and Intelligent system with Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, China. His research interests include Metadata Management, Semantic Interoperability, Information/Data Integration, and Approximation Algorithm.

Haoyang Che received the B.S. and M.S. degrees in Electrical Engineering from Beijing Normal University in 1998 and 2001, respectively. He is currently working towards the Ph.D. degree in Computer Science with Institute of Software, the Chinese Academy of Sciences, China. His research interests include trust management, software engineering, and P2P networks.

Pengfei Shi received the Bachelor's and Master's degrees in electrical engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 1962 and 1965, respectively. In 1980, he joined the Institute of Image Processing and Pattern Recognition (IPPR), SJTU. During the past 23 years, he worked in the area of image analysis, pattern recognition, and visualization. He has published more than 80 papers. He is currently the director of the Institute of IPPR at SJTU and a professor of pattern recognition and intelligent systems on the Faculty of Electronic and Information Engineering. He is a senior member of the IEEE.

Jun Gu received the BS degree in electrical engineering from the University of Science and Technology of China in 1982 and the PhD degree in computer science from the University of Utah in 1989. He has been the associate editor-in-chief of the IEEE Computer Society Press Editorial Board, an associate editor of the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on VLSI Systems, the Journal of Global Optimization, the Journal of Combinatorial Optimization, and the Journal of Computer Science and Technology, and is on the advisory board of International Book Series on Combinatorial Optimization. He was a chair of the 1995 National Academy of Sciences Information Technology Forum and was a chair of the 1996 National Science Foundation special event in celebration of 25 years of research on the satisfiability problem. He is a member of

the ACM, the ISA, the ISTS, the INNS, a senior member of the IEEE, and a life member of the AAAI.