5-2002

Product Schema Integration for Electronic Commerce: A synonym comparison approach

Guanghao YAN
State University of New York at Stony Brook

Wee-Keong NG
Nanyang Technological University

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

DOI: https://doi.org/10.1109/TKDE.2002.1000344

IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 14, NO. 3, MAY/JUNE 2002 pp. 583-598
https://doi.org/10.1109/TKDE.2002.1000344

Product Schema Integration for Electronic Commerce—A Synonym Comparison Approach

Guanghao Yan, *Student Member*, *IEEE*, Wee Keong Ng, *Member*, *IEEE Computer Society*, and Ee-Peng Lim, *Senior Member*, *IEEE*

Abstract—In any electronic commerce system, the heterogeneity of product descriptions is a critical impediment to efficient business information exchange. In the ABECOS electronic commerce system, buyer agents, seller agents, and directory agents liaise with one another in e-commerce activities. Only when agents have a common ontology of product descriptions (also called product schemas) are they able to interact seamlessly in e-commerce activities. This gives rise to the Product Schema Integration problem (PSI); the problem of integrating heterogeneous schemas of a certain product into one globally compatible schema. In this paper, we adopt an integration approach based on product attribute synonyms. We give a formal definition of the problem and show that it is NP-complete. We contrast our approach of study to conventional schema integration in federated databases. We also propose a set of approximate algorithms for PSI and evaluate their performance.

Index Terms—Electronic commerce, product description, relational schema, schema integration.

1 Introduction

As information on the Internet becomes more and more dynamic and heterogeneous, software agents have been touted as the new building blocks for a new Internet structure. From an electronic commerce perspective, agent technologies are useful for making providers aware of consumer needs and consumers aware of providers' offerings [1]. Organizations and individuals will not only establish their presence via web sites alone, they will also provide agents to interact with other agents. In order to realize this, the ABECOS (**A**gent-**B**ased **E**lectronic **CO**mmerce **S**ystem) Project at the School of Computer Engineering, Nanyang Technological University, Singapore, started in July 1997 with the key objective of designing and implementing a software infrastructure for a very large, distributed, agent-based web commerce system [21], [23], [24], [37], [38]. The project aims to construct generic prototypes for three types of agents: *buyer* agent, *seller* agent, and *directory* agent. Buyer agents and seller agents interact and transact on behalf of sellers and buyers. In addition, directory agents, corresponding to conventional web search engines, locate seller agents for buyer agents.

In ABECOS, directory agents keep information about seller agents. Since seller agents distinguish themselves by the products they sell, directory agents use the buyer's specifications of desired product as the search query to find relevant seller agents. In this case, directory agents no longer follow the simple keyword search strategy adopted by traditional search engines; they use users' product specifications as the search criteria, compare them with descriptions provided by different seller agents, and return the addresses of relevant seller agents. Once buyer agents obtain the addresses, they communicate with each relevant seller agent, query for the details of products, negotiate the prices, and complete the deal.

To describe a product, a set of attributes is used. For example, we may use (*Celine Dion*, *Falling into you*, *Sony*) to represent the singer's name, the title, and the company of a music CD, respectively. In relational database terminology, a set of column names of a relation is called the *schema* of that relation. Correspondingly, we may call the set of attribute names of a product the schema of that product. In this way, the schema of music CDs is (*artist*, *album*, *company*). Different instantiations of this schema correspond to different music CD records. However, different sellers may differ in the way they describe their products. They may adopt different sets of attributes or vocabularies to describe the same product. For example, (*year*, *classification*, *singer*, *title*, *company*) may be another schema for music CDs. We refer to such a seller-specific schema as a *local* product schema.

Since different sellers use different local product schemas, the directory agent in ABECOS is required to be "intelligent" enough to understand the semantics of different local product schemas and to identify the correspondence between the local schema attributes and the attributes involved in the buyer agents' search queries. Moreover, when buyer agents and seller agents in ABECOS interact directly, they should also understand each other's product descriptions. This is an *ontology heterogeneity* problem. Data integration is a solution to this problem; we can construct an integrated schema (or global schema) for a product before the user specifies the search query. As A. Levy has argued, a data-integration system lets users focus on specifying what they want, rather than thinking

- *G. Yan is with the Computer Science Department, State University of New York at Stony Brook, Stony Brook, NY 11794-4400. E-mail: guanghao@cs.sunysb.edu.*
- *W.K. Ng and E.-P. Lim are with the School of Computer Engineering, Nanyang Technological University, Nanyang Ave., Singapore 639798. E-mail: wkn@acm.org, aseplim@ntu.edu.sg.*

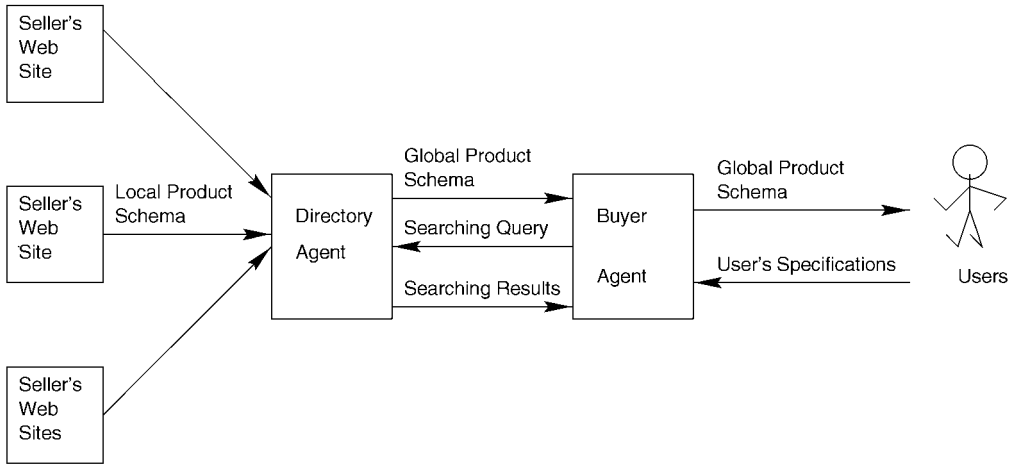

Fig. 1. Interactions among users, buyer agents, and directory agent.

about how to obtain the answers [14]. This is the main function of a global product schema. Using the global schema as the common description of products, the complex interaction among heterogeneous software agents (seller and buyer agents) can be carried out. As shown in Fig. 1, in ABECOS, the directory agent integrates different local product schemas into a global product schema; the buyer agent obtains the global schema from the directory agent when a user wants to buy such a product. Then, the user specifies his or her requirements for the product according to the global product schema, and the buyer agent sends these specifications as a search query to the directory agent. Finally, when the directory agent has located relevant seller agents, it returns the addresses of those seller agents as search results to the buyer agent.

Product schema integration is a subfield of data integration. In the Asilomar report on database research [3], in J. Gray's brief essay on database research [12], and in another paper on database research in the 21st century by A. Silberschatz et al. [34], heterogeneous data integration has been identified as one of the major research directions in the future. It has also been pointed out that, in electronic commerce systems, heterogeneous information sources must be integrated and the integration is an extremely difficult problem [34]. Although significant progress has been made in data integration, this problem is by no means solved, name matching across distributed sources remains one major problem [14]. Thus, we need schema management facilities such as directory agents in ABECOS which can adapt to the dynamic nature of heterogeneous data.

1.1 Characteristics of Product Schema Integration

As in other schema integration problems (such as in database schema integration), the heterogeneity among local product schemas in ABECOS can be classified into two categories, namely, *naming conflicts* and *missing attributes*. Naming conflicts include *synonyms*, words similar in meaning but different in spelling, and *homonyms*, words similar in spelling but different in meaning in different context. For example, *album* and *title* are synonyms in the local product schemas of music CDs. In addition, some product attributes used by one seller agent in ABECOS may not be used by another seller agent in ABECOS. This results in missing product attributes. For example, some sellers may use *chassis* as an attribute to describe a PC while some others do not.

It has been pointed out by D. Florescu et al. in a survey on database techniques for the World Wide Web [10] that web data integration has to, in addition, deal with a large and evolving number of web sources, little metadata about the characteristics of the sources, and a large degree of source autonomy. Specifically, product schema integration in the ABECOS commerce system has the following characteristics:

- **Limited knowledge of local schemas**. Since product information is proprietary, a directory agent may only obtain product schemas without much information about attribute domains or data types from the sellers' web pages. Thus, conventional schema integration methods built on the availability of attribute domain information no longer work. This results in additional difficulties in understanding the semantics of local product schemas.
- **Large number of local schemas**. The number of seller agents in the system can be quite large. In this situation, human intervention is hardly feasible. A scalable and fully automated solution is therefore required.
- **Fast local schema evolution**. Whenever new features of a product are added or old features of a product are removed, local schemas of the product must be updated. For example, a multimedia PC product can be extended to include additional peripherals which lead to new product schemas. Although the updates to a single local schema may not be very frequent, the total updates to all local schemas in the system can be so frequent that human intervention is also hardly feasible. Thus, the integration is again desired to be fully automated, low-cost, and efficient.

To successfully integrate product schemas, the approach we adopt should address the above characteristics. Note that most of these characteristics also apply to heterogeneous information integration in other non-agent-based distributed e-commerce systems. Because of the large

number of sellers and the frequent changes of information, an automatic integration is generally desired in the future. We may restate the problem as a set of basic questions:

- What is a description of a product? What is its schema?
- What is the information needed for identifying the correspondence between attributes from heterogeneous local product schemas? How is such information represented?
- How can we integrate different local product schemas? Is it possible to do the integration automatically?

We shall refer to this set of problems as the Product Schema Integration problem (PSI). Although there are many existing methodologies for schema integration in multidatabase systems, PSI is different because of the characteristics involved.

1.2 Paper Organization

The organization of this paper is as follows: The next section provides an overview of related work in the area of database schema integration and electronic commerce. Sections 3 and 4 elaborate on the formal definition and complexity of PSI, respectively. In Section 5, we compare the performance of three algorithms for PSI. Conclusions and future work are briefly discussed in Section 6 and some complexity results are given in the Appendix.

2 Related Work

2.1 Database Schema Integration

Schema integration is an important issue in the context of view integration (in database design) and database integration (in multidatabase management) [2]. Much work has been done during the past decades in these two areas [6], [7], [13], [17], [18], [19], [22], [26], [31], [35], [39]. It has been pointed out that success in schema integration depends on understanding the semantics of the schema components, i.e., attributes, relations, entity sets, etc., and the ability to reason with semantics. There are three levels of information that can be used to determine the semantics of attributes: attribute *names*, attribute *field specifications*, and attribute *domains*.

2.1.1 Levels of Schema

Attribute Name. At the attribute name level, it is found that having only attribute names is not sufficient for complete semantic schema integration [26]. The main problems of schema integration at this level are the presence of homonyms and synonyms. Homonyms can be detected by comparing concepts with the same name in different schemas, and synonyms can only be detected after external specifications [2]. According to the characteristics of PSI given in Section 1.1, product schema integration should be performed at this level.

In MUVIS (Multi-User View Integration System) [13], a synonyms lexicon has been used to identify the equivalence of attribute names. It overcomes the drawbacks of simple attribute name comparison, but to construct a synonyms lexicon is a tough and time consuming task. Furthermore, the relationships of words in a synonyms lexicon are static whereas the relationships of attribute names are dynamic; the equivalence relationship of some words may not hold from one context to another. For example, in the schema of an "automobile," *model* is an attribute name equivalent to *type* or *classification*, but this relationship does not apply in the schema of a "computer." To alleviate this situation, *common concepts*, i.e., properties or characteristics possessed by certain objects, can be used to help the understanding of attribute semantics [39]. For example, "horizontal-position" is a common concept for attributes like *x-axis* or *horizontal-axis*. In this way, homonyms can be easily detected since their concepts are different. For synonyms, when they have been identified in a synonyms lexicon, whether their concepts agree decides whether they are equivalent attributes names.

Field Specification. The second level of attribute correspondence is attribute field specifications, i.e., characteristics of attributes such as uniqueness, cardinality, domain, semantic integrity constraints, security constraints, allowable operations, and scale [22]. They are used to describe the properties of different attributes in database schemas. In the Semint project [19], [30], a classifier is used to categorize attributes according to their field specifications such as data type, length, and key fields, and an algorithm has been proposed to determine the degree of similarity and dissimilarity among attributes. However, this approach is based on the assumption that most database designers have the same knowledge about designing a "good" database. This assumption makes it possible to use attribute field specifications to help determine the likelihood that attributes are equivalent. One of the weaknesses of this approach is that attribute field information may not always be available, as is the case of ABECOS.

Attribute Domain. At the attribute domain level, relationships among attributes, entities, and relations are defined formally into five types: EQUAL, CONTAINS, OVERLAP, CONTAINED-IN, and DISJOINT [17]. Several formulations of different attribute integration strategies were given according to different relationships of attributes to be integrated. With this approach, entities and relations can be integrated by identifying the relationships of attributes of these entities and relations. This approach has been used to implement a tool for schema integration [32]. However, it has been pointed out that to determine such relationships can be time consuming and tedious [31], even with complete information of the domain of every attribute. The Carnot [7] project and MIT's COIN project [4] also address schema integration at this level. A large knowledge base has been established to store the information of different semantic contexts. This approach is based on the availability of the complete domain of each attribute.

2.1.2 Metadata Representation

Metadata are data that describe information of other data. In multidatabase systems, the metadata of schemas of component databases are critical to the whole process of database integration. Such metadata describe the correspondences among attributes of different schemas. Various metadata representations have been proposed [15]:

- *Tables*. Attribute correspondence can be maintained in a simple table structure. Each tuple in the table represents a static correspondence between attribute names in different databases. This is the simplest representation of metadata. For example, one tuple from the table representing metadata of music CD schemas can be (*singer*, *artist*).
- *Ontologies*. An ontological representation of attribute correspondence uses an ontology, a knowledge base consisting of entities and relationships with abstraction, inference, and typing mechanisms, to represent semantics of the component databases [7]. This is the most powerful and complex among the three. In the Carnot multidatabase system [7], a knowledge base known as Cyc is used to store ontologies [18]. Articulation axioms like $ist(G, \Phi) \Leftrightarrow ist(S_i, \Psi)$ are used to represent the logical relationship that logical expression Φ in global schema G is equivalent to logical expression Ψ in local schema S_i.

2.1.3 Relevance

In product schema integration in ABECOS, we need metadata to represent the semantic information of attribute names. We have the following considerations when we design the metadata representation: First, the fast evolution of local product schemas makes tables no longer suitable since tables can only represent static correspondences. Any change in one tuple may lead to changes in other tuples; thereby, making the maintenance of a table very expensive. Second, the diversity of alternative attribute names employed by different sellers makes it impossible to build a complete synonym lexicon for all kinds of products. Third, to predict all possible semantic context situations and to set up and maintain an ontology knowledge base is difficult because of the lack of complete attribute domain information. Last, in electronic commerce transactions fast response time is very important. The use of large knowledge bases may incur undesirable overheads that undermine the performance of the system. Therefore, we should find a simple but flexible way to represent the metadata of local product schemas.

2.2 Standardization Efforts in Electronic Commerce

Much effort has been expended to provide related standards on the issues of heterogeneity in electronic commerce [5], [8], [9], [16], [25], [27], [36]. If these standards are adopted widely by e-commerce participants, then heterogeneity problems will be largely resolved. However, it will be some time before the standards are widely used, and we foresee a future whereby multiple standards exist. Thus, product schema integration is still meaningful, though it will be simplified to some extent by the adoption of standards.

The eCo Framework Project [9], conducted by CommerceNet [8], has addressed some of the heterogeneity issues. They created a base set of common terms and mappings among existing terms for e-commerce specifications. The eCo working group consider a list of related specifications among which the RosettaNet Specification [27] and the Common Business Library (CBL) [5] shall be briefly discussed next.

2.2.1 RosettaNet Specification

RosettaNet [27] creates "property" definitions for various entities in electronic commerce, such as property definitions for a certain product. The word "property" is similar to "attribute" as discussed in Section 1. For example, "Modem" is a property (or an attribute) for computer products. Once these property definitions are completed, they will be distributed to some standards maintenance organizations that will then enumerate possible values for those properties. Then, property definitions, as well as their values, are distributed to companies in the industry supply chain as standards for business information format, say for product descriptions.

Let us take a look at how one property of a product is defined by RosettaNet. Consider the definition of the "Central Processor Unit" property for a laptop given by the RossettaNet Laptop Technical Specification [28]. There are several fields for the property "Central Processor Unit": *Property Name, Synonym, Property Definition, Dictionary References, Where Used, Property Type*, etc. Moreover, some of these fields contain subfields. For example, *Property Name* has *Abbreviation* and *Acronym* as its subfields. All these fields serve as metadata for the product property (attribute). In our product schema integration work, we use some of them to add semantics to product attributes. In fact, we have examined a mechanism based on "product attribute synonyms" to integrate product schema. We shall elaborate on this later.

2.2.2 Common Business Library (CBL)

The Common Business Library (CBL) that is developed by Veo Systems, Inc. [36] is a set of building blocks with common semantics and syntax to ensure interoperability among XML applications. CBL consists of information models for generic business concepts including:

- business description primitives like companies, services, and products,
- business forms like catalogs, purchase orders, and invoices, and
- standard measurements, date and time, location, classification codes.

Specifically, CBL consists of an extensible, public set of XML DTDs and modules. These building blocks can be assembled to create complete XML documents representing a business interaction such as a purchase order or an inventory stock query. Where possible, CBL takes advantage of other standards using, for example, relevant ISO standards for dates, currencies, and names. CBL is closely related to the work of RosettaNet, and the property definitions given by RossettaNet can be referenced by CBL to compose DTDs and modules for various e-commerce transactions, including product descriptions.

To use CBL, an organization starts by creating a CBL document describing its offer and services. Then, it integrates a CBL system with its back-end system by writing custom code that interprets information between the CBL format and the organization's previous format. It is like building a "wrapper" for back-end systems by using CBL blocks. After that, organizations interact on the basis of CBL semantics and syntax.

3 Problem Formulation

3.1 Synonym Set

We use *synonym set* to represent the metadata of product schemas. A synonym set is a set of alternative names for an attribute. These alternative names include synonyms, abbreviations, and acronyms for the attribute. It is different from *value set*, which is the domain (or set of possible values) of the attribute. For instance, {*type*, *sort*, *category*, *class*} is a synonym set representing the real world conceptualization of the genre of a music CD, such as {*rock*, *classical*, *jazz*, *country*, *Christian*, *R&B*, *misc*}. The synonym set refers to attributes that represent the genre of a music CD record while the value set contains the possible values of that attribute. In this way, the metadata of a product schema is represented as a set of synonym sets, which we call a *description*. For example, ({*artist*, *singer*}, {*album*, *title*}, {*publisher*, *company*}) is a description of music CD records. In ABECOS, when sellers describe local product schemas, they not only give the attribute names they are using, but also provide synonym sets as metadata to facilitate the understanding of descriptions by other sellers.

We say two synonym sets are *semantically coherent* if their intersection is not empty. Here, note that semantic coherence does not mean semantic equivalence; it only implies close semantic relationship. The size of the intersection of two synonym sets is an indicator of the degree of proximity between them. We observe that semantically coherent synonym sets probably refer *conceptually* to the "same attribute." For instance, synonym sets {*type*, *model*, *sort*, *category*, *suit*, *genus*, *genre*, *class*, *classification*} and {*sort*, *genre*, *style*, *class*, *classification*, *group*, *make*, *lot*, *type*, *kind*} are semantically coherent. However, it is still possible that some semantically coherent synonym sets do not refer to the same attribute, such as the case of homonyms. To handle this situation, we consider other issues like the degree of coherence of the synonym sets. We shall elaborate on these issues when we formally define the integration problem.

The approach of using synonym sets to represent the metadata of local product schemas is simple and flexible. There is no complicated knowledge or context information to be stored in knowledge bases. Generally, sellers are familiar with the terminology of the products they sell. Thus, they are able to provide synonym sets easily and accurately. When there are updates to local product schemas, such as adding or removing synonym sets or attribute names, sellers only need to change the synonym sets. Any change to one synonym set does not lead to additional changes in other synonym sets. Furthermore, the correspondence among local product schemas can be identified by examining the intersection, if any, of their synonym sets. In this way, the semantic identification process is transformed to the process of scalable and fully automated synonym set matching.

To further illustrate the use of synonym sets in describing products, we present an example of PC product descriptions, which is more complicated than the previous example of music CD product descriptions. These product schemas are abstracted and synthesized from the websites of some major computer vendors. As shown in Table 2, there are four different PC product descriptions in four separate small tables D_1, D_2, D_3, and D_4, respectively. Each cell of the small tables is a synonym set for a certain attribute. For example, synonym set {`memory,main memory`} in table D_1 represents the memory of a PC product while synonym set {`memory, RAM`} in table D_2 represents the same attribute of the product. In addition, we note that the numbers of synonym sets in these four tables are also different.

3.2 Problem Definition

Suppose the schema of a product is $D_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,t_i}\}$ $(1 \le i \le n,\ t_i \ge 1)$, where $A_{i,j}$ is the synonym set for the jth product attribute, and t_i is the number of synonym sets used by vendor i. We make a few observations. First, the number of synonym sets employed by each seller may differ due to the different number of attributes found in the local product schemas. Second, synonym sets from the same seller may contain common attribute names because an attribute name may have multiple semantics in different contexts. Last, attribute synonym sets from different sellers may contain common attribute names because the sellers may choose the same attribute name for an attribute. We formally express these observations below: For all $1 \le i, j \le n$, $i \ne j$, we have

1. $|t_i - t_j| \ge 0$;
2. $|A_{i,u} \cap A_{i,v}| \ge 0$ for $1 \le u, v \le t_i$;
3. $|A_{i,u} \cap A_{j,v}| \ge 0$ for $1 \le u \le t_i$, $1 \le v \le t_j$.

The problem that we address may be informally described as follows: Given a set of heterogeneous descriptions $D_1, D_2, \ldots, D_n$ by n sellers for the same product, we wish to find a description $\mathbf{D} = \{I_1, I_2, \ldots, I_m\}$ $(m \ge t_i,\ 1 \le i \le n)$ for the product that integrates all the different schemas from the sellers. Here, I_k, $1 \le k \le m$ is an integrated synonym set for a certain attribute. Specifically, I_k is an intersection of some semantically coherent synonym sets belonging to different sellers. I_k is derived from D_is as follows: We first define a *column* P_k of $U = \bigcup_{i=1}^{n} D_i$ to be a subset of U such that $|P_k \cap D_i| \in \{0, 1\}$ for $1 \le i \le n$ and $\bigcap_{B \in P_k} B \ne \emptyset$. All names in one column are assumed to be semantically coherent and at most one synonym set from each D_i may be included in the column. Then, I_k $(1 \le k \le m)$ is defined as the intersection of synonym sets in P_k, i.e., $I_k = \bigcap_{B \in P_k} B$.

Therefore, to find an integrated description $\mathbf{D}$ is to find a set of columns each corresponding to a group of semantically coherent synonym sets. Using the concept of *partitions* in set theory, the set of columns is a partition of U. We denote it as $\mathbf{P} = \{P_1, P_2, \ldots, P_\ell\}$ $(\ell \ge m)$. However, such a partition may not be unique from the three observations mentioned previously. We introduce two notions for characterizing a partition: A partition is *correct* if synonym sets within each column of the partition have nonempty intersection. A partition is *good* if synonym sets within each column of the partition have a high degree of semantic coherence. In general, we want to find partitions that are correct and have a high degree of goodness. Let us examine factors that determine the goodness of a partition.

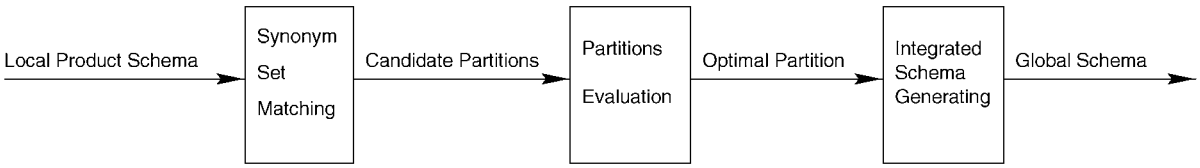

Fig. 2. Steps of Product Schema Integration.

Column Similarity. This measures the degree of similarity among synonym sets in the column. Let $P_k = \{A_1, A_2, \ldots, A_{k_m}\}$ for $1 \leq k_m \leq n$. Let $W = \bigcup_{A \in P_k} A$ such that $|W| = w$. We define a w-dimensional vector

$$\mathbf{W} = (a_1, a_2, \ldots, a_w),$$

where $a_u \in W$ for $1 \leq u \leq w$. Then, a w-dimensional vector C_v is derived from A_v $(1 \leq v \leq k_m)$ as follows: Let $C_v(u)$ $(1 \leq u \leq w)$ be the uth component of vector C_v. Then, $C_v(u) = 1$ if $\mathbf{W}(u) \in A_v$; $C_v(u) = 0$ otherwise. Let C_{P_k} be the mean vector of all synonym sets in P_k, i.e., $C_{P_k} = \sum_{v=1}^{k_m} C_v / k_m$. The *similarity* of a column P_k is given by $\sum_{v=1}^{|P_k|} \tau(C_v, C_{P_k})/|P_k|$, where function τ is defined as follows: For vectors $C_X = (x_1, x_2, \ldots, x_w)$ and $C_Y = (y_1, y_2, \ldots, y_w)$,

$$\tau(C_X, C_Y) = \frac{\sum_{u=1}^{w} x_u y_u}{\sqrt{\sum_{u=1}^{w} x_u^2 \cdot \sum_{u=1}^{w} y_u^2}}.$$

This function is generally used to compute vector similarity in information retrieval [29]. From its definition, we have $0 \leq \tau(C_X, C_Y) \leq 1$. Since each vector C_v has a similarity value of C_{P_k}, the mean of these similarities is the column similarity of P_k, i.e., the similarity of all the synonym sets in P_k. The more similar synonym sets in P_k are, the larger is the value of column similarity; when all the synonym sets are identical, the column similarity is maximum.

Column Popularity. This measures the extent to which the column is contributed by each seller. The *popularity* of a column P_k is given by the ratio $0 \leq (|P_k| - 1)/(n - 1) \leq 1$. By definition, the number of synonym sets in P_k lies between 1 and n. When $|P_k| = n$, number of sellers, the popularity of P_k is maximum. When $|P_k| = 1$, the attribute referred by P_k is used by only one seller. We observe that a popular column contributes to a better partition.

Column Evenness. This measures the extent to which synonym sets in P_k are semantically coherent. The *evenness* of a column P_k is given by $0 \leq |\bigcap_{A \in P_k} A|/|P_k| \sum_{A \in P_k} (1/|A|) \leq 1$. For an arbitrary synonym set A in P_k, $|\bigcap_{B \in P_k} B|/|A|$ is the degree of semantic coherence between A and the other synonym sets. When the alternative attribute names of A are all employed by the other synonym sets in the same column—the ideal case, $|\bigcap_{B \in P_k} B|/|A|$ has a maximal value of 1. When no other synonym sets employ any common words in A, $|\bigcap_{B \in P_k} B|/|A|$ has a minimal value of 0. Let this value be the *coherence scale* of A. Then, the arithmetic mean of the coherence scale of synonym sets in P_k is the evenness of P_k. We note that its value also lies between 0 and 1. The higher the semantic coherence among the synonym sets, the higher is the evenness.

The combination of similarity, popularity, and evenness yields the following measure:

$$\varphi(P_k) = h_1\left(\sum_{v=1}^{|P_k|} \frac{\tau(C_v, C_{P_k})}{|P_k|}\right) + h_2\left(\frac{|P_k| - 1}{n - 1}\right) + h_3\left(\frac{|\bigcap_{A \in P_k} A|}{|P_k|} \sum_{A \in P_k} \frac{1}{|A|}\right)$$

for computing the goodness of P_k, where the weights h_1, h_2, h_3 sum to 1. For simplicity, we may give equal importance to the three measures by having equal values for the weights. For the entire partition $\mathbf{P}$, the following function:

$$\xi(\mathbf{P}) = \frac{1}{\ell} \sum_{k=1}^{\ell} \varphi(P_k)$$

is an evaluation of the goodness of the partition. Since $0 \leq \varphi(P_k) \leq 1$, we have $0 \leq \xi(\mathbf{P}) \leq 1$. In general, we would like $\xi(\mathbf{P})$ to be as large as possible. With the above preliminaries, we are now ready to define the Product Schema Integration Problem. We give the formal definition of the problem as follows:

Definition 1. *Given a set of descriptions $D_1, D_2, \ldots, D_n$ $(n \geq 1)$, where $D_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,t_i}\} (1 \leq i \leq n, t_i \geq 1)$ and $A_{i,j} (1 \leq j \leq t_i)$ is a synonym set, find a partition $\mathbf{P} = \{P_1, P_2, \ldots, P_\ell\} (\ell \geq \min(t_i))$ of $\bigcup_{i=1}^{n} D_i$ satisfying the following conditions:*

1. $|\bigcap_{A \in P_k} A| \geq 1$ *for* $1 \leq k \leq \ell$,
2. $|P_k \cap D_i| \in \{0, 1\}$ *for* $1 \leq i \leq n$ *and* $1 \leq k \leq \ell$,
3. $P_k \cap P_j = \emptyset$ *for* $1 \leq k, j \leq \ell$ *and* $k \neq j$, *and*
4. $\xi(\mathbf{P})$ *is maximal.*

Once a partition $\mathbf{P}$ exists, we may compute the integrated schema as follows: $\mathbf{D} = \{I_k \mid I_k = \bigcap_{B \in P_k} B, 1 \leq k \leq \ell\}$, which is trivial compared to the process to compute $\mathbf{P}$. Thus, the Product Schema Integration Problem is essentially the problem of finding $\mathbf{P}$. Fig. 2 shows the three steps of Product Schema Integration: synonym set matching, partition evaluation, and integrated schema generation. At the end of these three steps, the input local product schemas can be integrated into an output global schema. Note that the Schema Integration Problem defined above is an optimization problem. The difficulty lies in the fact that when we find one partition satisfying the first three conditions of the problem, we do not know whether it is the optimum partition, i.e., whether ξ is maximum. In order to analyze the complexity of the problem in a more convenient way, we redefine the problem as a decision problem:

TABLE 1
Optimal Partition Resulting from the Schema Integration Process

D_1	D_2	D_3	D_4
{type,sort,classification}	{type,classification,model}	{type,kind,model}	{type,classification}
{processor,CPU,chipset}	{chip,processor,CPU}	{chipset,processor}	{CPU,processor}
{model,serial number,brand}		{model,manufacturer}	
{memory,main memory}	{memory,RAM}	{RAM,memory}	{memory,main memory}
{cache}	{cache}	{cache}	{cache}
{CD-Rom,CD-drive}	{CD-Rom}	{CD-Rom}	{CD-Rom}
{hard drive,hard-drive}	{h-drive,hard drive}	{hard drive}	
{floppy drive,f-drive}		{floppy drive,f-drive}	
{modem}	{modem,modem/fax}		{modem}
{open bays,bays,bay}	{total bays,bays}	{bays,open bays}	{bay,bays}
{expansion slots,slots}	{I/O slots,slots}	{slots,slot}	{slots,I/O slots}
{mouse}	{mouse}	{mouse}	{mouse}
{keyboard}	{keyboard}	{keyboard}	{keyboard}
{ports,bus-ports}	{ bus-ports,ports}	{busports,ports}	
{chassis}		{chassis,chassis style}	
{warranty}	{warranty}	{warranty}	
{weight,bulk}	{weights,weight}		
	{color,colors}	{color,coloration}	{color}

Definition 2 (Product Schema Integration). *Given a number* $M(0 \leq M \leq 1)$*, and a set of descriptions* $D_1, D_2, \ldots, D_n$ $(n \geq 1)$*, where* $D_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,t_i}\}$ $(1 \leq i \leq n, t_i \geq 1)$ *and* $A_{i,j}$ $(1 \leq j \leq t_i)$ *is a synonym set, does there exist a partition* $\mathbf{P} = \{P_1, P_2, \ldots, P_\ell\}$ $(\ell \geq \min(t_i))$ *of* $\bigcup_{i=1}^{n} D_i$ *satisfying the following conditions:*

1. $|\bigcap_{A \in P_k} A| \geq 1$ *for* $1 \leq k \leq \ell$,
2. $|P_k \cap D_i| \in \{0, 1\}$ *for* $1 \leq i \leq n$ *and* $1 \leq k \leq \ell$,
3. $P_k \cap P_j = \emptyset$ *for* $1 \leq k, j \leq \ell$ *and* $k \neq j$, *and*
4. $\xi(\mathbf{P}) \geq M$?

3.3 An Example

So far, we have discussed the definition of Product Schema Integration Problem and steps to do such an integration. Now, we shall illustrate PSI by integrating the four PC product schemas given in Table 2 in Section 3.1. To perform the Product Schema Integration in this example, we should first find a partition **P**. Table 1 shows such an example of a partition: Synonym sets in the same row refer to the "same" attribute and each row is a column (as defined in Section 3.2).

Note that in D_1 and D_2, {`processor, CPU, chipset`} and {`chip, processor`} are two synonym sets referring to the CPU of a PC. Although these two synonym sets are different, their intersection is not empty. Thus, they are semantically coherent. However, we observe that two synonym sets {`type, sort, classification`} and {`model, serial number, brand`} in D_1 intersect with synonym set {`type, classification, model`} in D_2. Conceptually, {`type, sort, classification`} refers to the classification of a PC such as *desktop* or *notebook*, and {`model, serial number, brand`} refers to the brand of the PC such as *IBM* or *Apple*. Thus, {`type, classification, model`} refers to the same attribute of a PC as the former one of the two synonym sets in D_1.

TABLE 2
Different Descriptions of PCs from Different Sellers

D_1
{type,sort,classification}
{processor,CPU,chipset}
{model,serial number,brand}
{memory,main memory}
{cache}
{CD-Rom,CD-drive}
{hard drive,hard-drive}
{floppy drive,f-drive}
{modem}
{open bays,bays,bay}
{expansion slots,slots}
{mouse}
{keyboard}
{ports,bus-ports}
{chassis}
{warranty}
{weight,bulk}

D_2
{memory,RAM}
{type,classification,model}
{total bays,bays}
{h-drive,hard drive}
{CD-Rpm}
{modem,modem/fax}
{chip,processor,CPU}
{I/O slots,slots}
{bus-ports,ports}
{keyboard}
{mouse}
{warranty}
{color,colors}
{weights,weight}

D_3
{CD-Rpm}
{type,kind,model}
{chassis,chassis style}
{hard drive}
{chipset,processor}
{RAM,memory}
{f-drive,floppy drive}
{bays,open bays}
{slots,slot}
{mouse}
{keyboard}
{warranty}
{color,coloration}
{model,manufacturer}
{warranty}
{busports,ports}

D_4
{color}
{type,classification}
{CPU,processor}
{memory,main memory}
{cache}
{CD-Rom}
{modem}
{bay,bays}
{slots,I/O slots}
{mouse}
{keyboard}

From Table 1, we compute the intersections of the synonym sets in each row and derive an integrated description

$\mathbf{D} = \{\{\texttt{type}\}, \{\texttt{processor}\}, \{\texttt{model}\}, \{\texttt{memory}\}, \{\texttt{cache}\},$
$\{\texttt{CD-Rom}\}, \{\texttt{hard drive}\}, \{\texttt{floppy drive}, \texttt{f-drive}\},$
$\{\texttt{modem}\}, \{\texttt{bays}\}, \{\texttt{slots}\}, \{\texttt{mouse}\}, \{\texttt{ports}\}, \{\texttt{keyboard}\},$
$\{\texttt{chassis}\}, \{\texttt{warranty}\}, \{\texttt{weight}\}, \{\texttt{color}\}\}.$

4 Complexity of PSI

The problem of integrating heterogeneous product schemas is originally an optimization problem (as defined in Definition 1). The difficulty comes from the large number of possible candidate solutions (partitions). In the worst case, the intersection of one synonym set and any synonym set from the other sellers can be nonempty. We shall analyze the complexity of the problem in this case.

Suppose there are n sellers each having m synonym sets and the size of each column in any partition is n. Then, let us compute the number of possible ways to construct a solution (partition): To constitute the first column in one partition, there are m^n possible ways since we can select one synonym set from each seller. For the second column, there are $(m-1)^n$ ways since each seller now has $m-1$ synonym sets after the first selection. In this way, we have $m^n(m-1)^n \cdots 2^n 1^n$, i.e., $(m!)^n$ ways to construct one candidate partition.

However, the actual number of candidate partitions is less than $(m!)^n$ because some of these combinations result in the same partition; a partition is only a combination of columns, not the permutation of them. As the number of permutations of columns is $m!$, the number of possible partitions is thus $(m!)^n/m!$, i.e., $(m!)^{n-1}$. According to Stirling's series, we have

$$m! = \sqrt{(2\pi m)}\Big(\frac{m}{e}\Big)^m e^{\mu(m)},$$

where

$$\mu(m) = \frac{1}{12m} - \frac{1}{360m^3} + \frac{1}{1260m^5}.$$

Clearly, $(m!)^{n-1}$ is an exponential function of n. In the general case when the size of each column varies between 1 and n, the complexity of the problem is far higher.

In order to analyze the complexity of PSI, we have redefined it as a decision problem in Definition 2. In the rest of this section, we shall prove that PSI is NP-complete. To show that PSI is NP-complete, we first define a version of PSI called UPSI and show that it is NP-complete. Then, we reduce UPSI to PSI (Lemma 3). In order to show that UPSI is NP-complete, we show that n-dimensional matching (nDM) reduces to UPSI (Lemma 2). The complexity results of nDM is presented in the Appendix. We begin by showing that PSI is NP in the following lemma:

Lemma 1. PSI $\in NP$.

Proof. We can solve PSI with a nondeterministic algorithm: For all the possible partitions of $\bigcup_{i=1}^{n} D_i$, compute the value of ξ to see whether it satisfies the requirements given in the definition. It is obvious that the time complexity for checking each of the first three requirements in the definition is polynomial. For the fourth requirement, suppose there are n sellers and the average number of synonym sets of each seller is $m \geq 1$. Then, for a partition $\mathbf{P}$, $\max(|\mathbf{P}|) = n \cdot m$, and for each column of $\mathbf{P}$, the time complexity of computing function φ in all the cases is $O(n^3)$. Thus, the time complexity to compute $\xi(\mathbf{P})$ is $O(mn^4)$. This means that PSI can be solved in polynomial time by a nondeterministic algorithm. □

Next, we define UPSI (Uniform Product Schema Integration problem), a subproblem of PSI and show that it is in NP-complete:

Definition 3 (UPSI). *Given a number M $(0 \leq M \leq 1)$, and a set of descriptions $D_1, D_2, \ldots, D_n$ $(n \geq 1)$, where $D_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,m}\}$ $(1 \leq i \leq n, m \geq 1)$ and $A_{i,j}$ $(1 \leq j \leq m)$ is a synonym set, does there exist a partition $\mathbf{P} = \{P_1, P_2, \ldots, P_m\}$ of $\bigcup_{i=1}^{n} D_i$ satisfying the following conditions:*

1. $|P_k| = n$ *for* $1 \leq k \leq m$,
2. $|\bigcap_{A \in P_k} A| \geq 1$ *for* $1 \leq k \leq m$,
3. $|P_k \cap D_i| \in \{0, 1\}$ *for* $1 \leq i \leq n$ *and* $1 \leq k \leq m$,
4. $P_k \cap P_j = \emptyset$ *for* $1 \leq k, j \leq m$ *and* $k \neq j$, *and*
5. $\xi(\mathbf{P}) \geq M$*?*

Lemma 2. UPSI $\in NP$*-complete.*

Proof. It is easy to see that UPSI $\in$ NP since a nondeterministic algorithm need only to guess a candidate partition and check in polynomial time whether that partition can satisfy all the given requirements.

We transform nDM to UPSI. First, we consider an instance of nDM. Let $W_1, W_2, \ldots, W_n$ be n disjoint sets each having the q elements such that $W_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,q}\}$ for $1 \leq i \leq n$. Let n-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k})$, $1 \leq i, j, k \leq q$, be an arbitrary element of the set $S \subseteq W_1 \times W_2 \times \cdots \times W_n$. We shall construct n sets $D_1, D_2, \ldots, D_n$ in the corresponding instance of UPSI.

Suppose $S = (s_1, s_2, \ldots, s_m)$, $m \geq 1$. We use a one-to-one function θ to assign a distinguished value to each element of S, such that $\theta(s_i) = y_i$. Note that $y_i \neq w_{j,k}$, $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq k \leq q$. Set D_ℓ $(1 \leq \ell \leq n)$ contains q elements as is $A_{\ell,i}$ $(1 \leq i \leq q)$ which is constructed as follows: $A_{\ell,i} = \{w_{\ell,i}\} \cup Y$, where $Y = \{y_j\}$ for $1 \leq j \leq m$ if and only if $w_{\ell,i}$ is the ℓth element of s_j; $Y = \emptyset$ if $w_{\ell,i}$ is not an element of any n-tuple s_j.

To another parameter M in the instance of UPSI, we let $M = 0$. Thus, we have constructed an instance of UPSI from the instance of nDM. We shall prove that each of these two instances is satisfiable if and only if the other one is satisfiable.

If the instance of nDM is satisfiable, then there exists a set $S' \subseteq S$ satisfying the following conditions:

1. $|S'| = q$,
2. $\forall s_1 = (w_{1,i}, w_{2,j}, \ldots, w_{n,k})$ and $s_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}) \in S'$ for $1 \leq i, j, k, i', j', k' \leq q$, if $s_1 \neq s_2$, then $w_{1,i} \neq w_{1,i'}$, $w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'}$.

For each n-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}) \in S'$ there is a corresponding set $P_t = \{A_{1,i}, A_{2,j}, \ldots, A_{n,k}\}$, $1 \leq t \leq q$ having the following properties:

1. The first element of each of $A_{1,i}, A_{2,j}, \ldots, A_{n,k}$ are $w_{1,i}, w_{2,j}, \ldots, w_{n,k}$, respectively.
2. If $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}) = s_r$, $1 \leq r \leq m$, then according to the construction of $D_1, D_2, \ldots, D_n$, $\{y_r\} \subseteq \bigcap_{A \in P_t} A$;.
3. $\{y_r\} = \bigcap_{A \in P_t} A$ because of the second condition satisfied by S' (if there is $y_{r'} \in \bigcap_{A \in P_t} A$, then there is an n-tuple $s_{r'} \in S'$ such that $s_{r'} = s_r$, which conflicts with the second condition satisfied by S').

Since $|S'| = q$, there exist q such sets $P_1, P_2, \ldots, P_q$ comprising a partition $\mathbf{P} = \{P_1, P_2, \ldots, P_q\}$ of $D_1 \cup D_2 \cup \cdots \cup D_n$, satisfying the following conditions:

1. $|P_k| = n$ for $1 \leq k \leq q$,
2. $|\bigcap_{A \in P_k} A| = 1$ for $1 \leq k \leq q$,
3. $|P_k \cap D_i| = 1$ for $1 \leq i \leq n$ and $1 \leq k \leq q$,
4. $P_k \cap P_j = \emptyset$ for $1 \leq k, j \leq q$ and $k \neq j$, and
5. $\xi(\mathbf{P}) > 0$.

The first condition is satisfied because $P_t = \{A_{1,i}, A_{2,j}, \ldots, A_{n,k}\}$ for $1 \leq t \leq q$. The second and fourth conditions are satisfied from the previous description of the third property of P_t. The third condition is obviously satisfied from the construction of P_t. Because the second condition is satisfied, we have $\phi(P_t) > 0$ and $\xi(\mathbf{P}) > 0$. Then, the fifth condition is also satisfied. Therefore, the instance of UPSI is satisfiable.

Conversely, suppose the instance of UPSI is satisfiable, i.e., there exists a partition $\mathbf{P} = \{P_1, P_2, \ldots, P_q\}$ of $D_1 \cup D_2 \cup \cdots \cup D_n$ satisfying the five situations given by UPSI's definition. Let $P_t = \{A_{1,i}, A_{2,j}, \ldots, A_{n,k}\}$, $1 \leq t \leq q$. Then, the first element of each of $A_{1,i}, A_{2,j}, \ldots, A_{n,k}$, which are $w_{1,i}, w_{2,j}, \ldots, w_{n,k}$, construct an

$$n\text{-tuple}(\mathrm{w_{1,i}}, \mathrm{w_{2,j}}, \ldots, \mathrm{w_{n,k}}) \in \mathrm{W_1} \times \mathrm{W_2} \times \cdots \times \mathrm{W_n}.$$

Consequently, there are q such n-tuples, and they compose a set S'. Since $D_1, D_2, \ldots, D_n$ are derived from $S \subseteq W_1 \times W_2 \times \cdots \times W_n$, it is not difficult to see that $S' \subseteq S$. The set S' has the following properties:

1. $|S'| = q$,
2. $\forall s_1 = (w_{1,i}, w_{2,j}, \ldots, w_{n,k})$ and $s_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}) \in S'$, $1 \leq i, j, k, i', j', k' \leq q$, if $s_1 \neq s_2$, then $w_{1,i} \neq w_{1,i'}$, $w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'}$.

The first property is obvious. As to the second one, we suppose that $s_1 \neq s_2$, but $w_{z,x} = w_{z,x'}$, $1 \leq z \leq n$, $1 \leq x, x' \leq q$. However, s_1 is derived from $P_t = (A_{1,i}, A_{2,j}, \ldots, A_{z,x}, \ldots, A_{n,k})$, and s_2 is derived from $P_{t'} = (A_{1,i'}, A_{2,j'}, \ldots, A_{z,x'}, \ldots, A_{n,k'})$, $1 \leq t, t' \leq q$. Thus, the first element of each of $A_{z,x}$ and $A_{z,x'}$ are $w_{z,x}$ and $w_{z,x'}$ respectively, which are the same. This tells us that $A_{z,x} = A_{z,x'}$ which conflicts with the condition that $P_t \cap P_{t'} = \emptyset$. Therefore, the second property of S' is valid. Now, it is clear that S' is the solution to the instance of nDM.

To see that the transformation from the instance of nDM to the instance of UPSI can be performed in polynomial time, it is suffices to observe that the number of synonym sets in the instance of UPSI is bounded by a polynomial in q, and the details of the construction itself are straightforward. □

```
foreach description D_i, 1⩽i⩽n
    P_i = ∅
    foreach synonym set A_{i,k} of D_i, 1⩽k⩽t_i
        P_k = {A_{i,k}}
        foreach description D_j, 1⩽j⩽n, j ≠ i
            foreach synonym set A_{j,u} of D_j, 1⩽u⩽t_j
                s_{j,u} = |A_{i,k} ∩ A_{j,u}|
            endfor
            P_k = P_k ∪ {A_{j,r}} where r = max_u{s_{j,u}}
            D_j = D_j − {A_{j,r}}
        endfor
        P_i = P_i ∪ {P_k}
    endfor
endfor
P = P_ℓ where ℓ = max_i{ξ(P_i)}
```

Fig. 3. Algorithm I for PSI.

Finally, we show that UPSI reduces to PSI:

Lemma 3. UPSI $\propto$ PSI.

Proof. We define a transformation from each instance of UPSI to a corresponding instance of PSI. Suppose there is an arbitrary instance of UPSI: $M = q$, n descriptions, and each has m element sets. The corresponding instance of PSI also has n descriptions identical to those in UPSI, i.e., $t_i = m, 1 \leq i \leq n$. The number M in PSI is now q.

It is easy to see that this transformation can be computed by a polynomial time algorithm. For each of the n descriptions in PSI that must be specified, it is exactly the same as each of those n descriptions in UPSI, and the value of M is also the same.

Consequently, if the answer to UPSI is "yes," i.e., there exists a partition satisfying the five requirements in the definition of UPSI, this partition is also a solution to PSI. Thus, the answer to PSI should also be "yes." If the answer to PSI is "no," i.e., there is no partition satisfying the requirements in the definition of PSI, it is also true that the answer to UPSI is "no." □

Since PSI is in NP (Lemma 1) and it is in NP-hard (Lemma 3), therefore PSI is in NP-complete:

Theorem 1. PSI $\in$ *NP-complete*.

5 Algorithms for PSI

Since PSI is NP-complete, it is not likely to find good polynomial time deterministic algorithms. Thus, we focus on finding approximate algorithms that are acceptable with respect to both time complexity and result accuracy. Figs. 3, 4, and 5 show three approximation algorithms for PSI.

5.1 Approximation Algorithms

Algorithm I. The following algorithm is repeated (first loop, index i) for each seller D_i: For each synonym set (second loop, index k) of the seller, we select the most semantically coherent (having the largest intersection) synonym set (third loop, index r) from each of the other sellers' descriptions (third loop, index j); these semantically coherent synonym sets constitute a column P_k. That is, $A_{i,k}$ $(1 \leq k \leq t_i)$ of D_i $(1 \leq i \leq n)$ is compared with $A_{j,u}$ $(1 \leq u \leq t_j)$ of D_j $1 \leq j \leq n$ by evaluating the size of their intersection (fourth loop).

```
foreach description D_i, 1⩽i⩽n
    P_i = ∅
    foreach synonym set A_{i,k} of D_i, 1⩽k⩽t_i
        P_k = {A_{i,k}}
        foreach description D_j, 1⩽j⩽n, j ≠ i
            foreach synonym set A_{j,u} of D_j, 1⩽u⩽t_j
                A'_{i,k} = ⋃_{A∈P'_k} A
                s_{j,u} = |A'_{i,k} ∩ A_{j,u}|
            endfor
            P_k = P_k ∪ {A_{j,r}} where r = max_u{s_{j,u}}
            D_j = D_j − {A_{j,r}}
        endfor
        P_i = P_i ∪ {P_k}
    endfor
endfor
P = P_ℓ where ℓ = max_i{ξ(P_i)}
```

Fig. 4. Algorithm II for PSI.

```
foreach description D_i, 1⩽i⩽n
    P_i = ∅
    foreach synonym set A_{i,k} of D_i, 1⩽k⩽t_i
        P_k = {A_{i,k}}
        foreach description D_j, 1⩽j⩽n, j ≠ i
            foreach synonym set A_{j,u} of D_j, 1⩽u⩽t_j
                A'_{i,k} = ⋂_{A∈P'_k} A
                s_{j,u} = |A'_{i,k} ∩ A_{j,u}|
            endfor
            P_k = P_k ∪ {A_{j,r}} where r = max_u{s_{j,u}}
            D_j = D_j − {A_{j,r}}
        endfor
        P_i = P_i ∪ {P_k}
    endfor
endfor
P = P_ℓ where ℓ = max_i{ξ(P_i)}
```

Fig. 5. Algorithm III for PSI.

Each iteration of the second loop results in a column, which constitutes partition $\mathbf{P}_i$ at the end of each of the first loop's iteration. This results in n candidate partitions. The final step in the algorithm chooses the partition having the largest goodness value.

When determining the time complexity of the above algorithm, we make the following two assumptions:

- The average value of t_i $(1 \leq i \leq n)$, the number of synonym sets in each seller, is m $(m \geq 1)$.
- The time spent on matching any two synonym sets is constant.

Consider the fourth loop, which iterates through every synonym sets of a seller. Based on the first assumption, the number of iterations is m. However, when a synonym set has been selected to join a column P_k (of a synonym set in the second loop), it is removed from considerations in subsequent iterations. Hence, the number of iterations of the fourth loop decreases: $m, m-1, \ldots, 1$ as we iterate through each synonym set of the second loop.

As there are $n-1$ other sellers to match against for a given seller, the third loop iterates $n-1$ times. Last, the first and second loops iterate n and m times, respectively. Therefore, the complexity of the whole algorithm is $n(n-1)\sum_{g=1}^{m} g = n(n-1)m(m+1)/2 = O(n^2m^2)$. Note that the time complexity in the best, worst, and average cases is the same since the number of comparisons in these three cases remain the same.

In reality, we expect the complexity of the algorithm to be way below $O(n^2m^2)$ because the average number of attributes to describe a product is not very large, say around 50 at most. Thus, the algorithm reduces to $O(n^2)$.

Algorithm II. Fig. 4 shows Algorithm II. For PSI, the difference between Algorithms I and II is in the fourth loop: In Algorithm I, we compute the size of the intersection of synonym sets $A_{j,u}$ and $A_{i,k}$, while in Algorithm II, we compute the size of the intersection of synonym set $A_{j,u}$ and the union of all the existing synonym sets already in P_k. In other words, in Algorithm I, we select the synonym set having the largest intersection with $A_{i,k}$ to join the column P_k, while in Algorithm II, we select the synonym set having the largest intersection with the union of all existing synonym sets in P_k to join P_k.

Based on the second assumption made in computing the time complexity of Algorithm I, we know that the time complexity of Algorithm II is the same as Algorithm I since the time to compute the union of the synonym sets in column P_k can be treated as constant. Hence, the time complexity of Algorithm II is also $O(n^2m^2)$, as we can see from the algorithms themselves.

Algorithm III. Fig. 5 shows Algorithm III. For PSI, Algorithm III is almost the same as Algorithm II except for one operation in the fourth loop: In Algorithm III, we compute the size of the intersection of synonym set $A_{j,u}$ and the intersection of all the existing synonym sets already in column P_k, while in Algorithm II, we compute the size of the intersection of synonym set $A_{j,u}$ and the union of all the existing synonym sets already in column P_k. Algorithm III seems to be more "strict" in constructing a column; it ensures that the evenness of the solution is larger than zero.

The time complexity of Algorithm III is also $O(n^2m^2)$ since it has a structure similar to Algorithms I and II.

Therefore, the time complexity of the three approximation algorithms are the same. The difference among them is their performance, i.e, the goodness of the resulting partitions. We shall elaborate on this in the next section.

5.2 Performance Comparison

The performance of each algorithm on different input product descriptions with respect to different parameters are shown in Figs. 6, 7, 8, and 9. We plot four sets of graphs where the vertical axes are goodness, similarity, popularity, and evenness of a solution. In each set, we use each of the following parameters in turn as the horizontal axis:

p The average percentage of input overlap, i.e., the average percentage of synonym set overlap within a seller.

n The number of input descriptions, i.e., the number of sellers.

m The average size of each description, i.e., the average number of synonym sets in each description.

w The average size of each synonym set.

We created synthetic descriptions of PCs as inputs for our experiments. For realism, we refer to actual PC descriptions from websites of PC sellers when generating synthetic

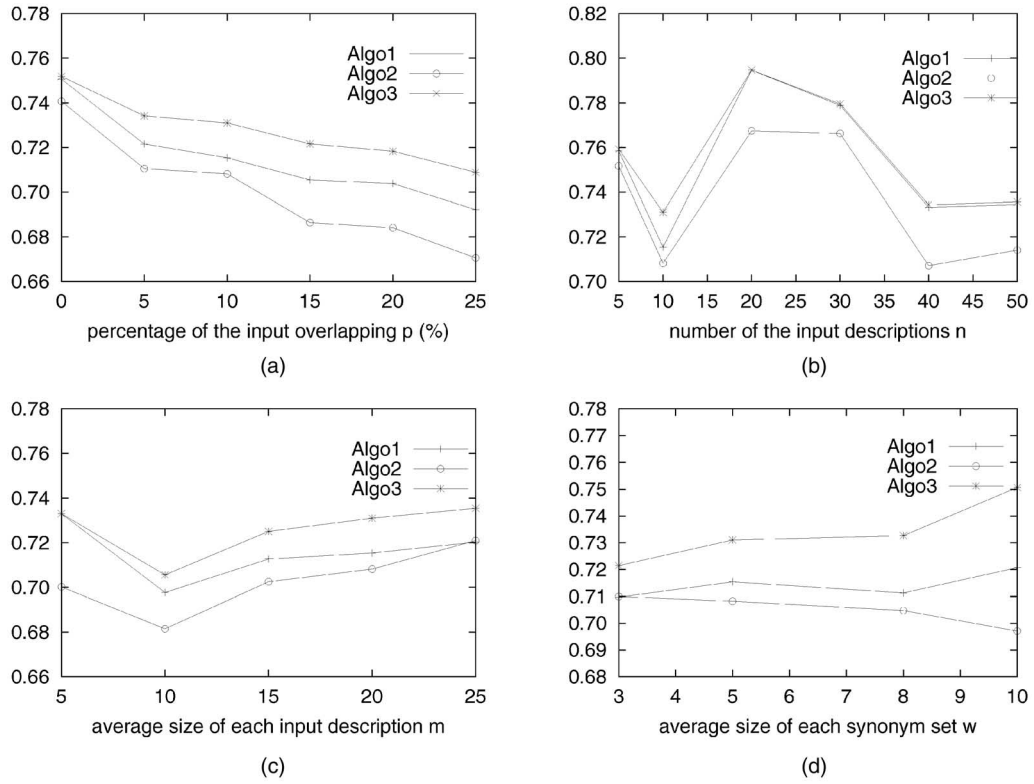

Fig. 6. Goodness of solution versus different values of percentage of input overlap, number of input descriptions, average size of input descriptions, and average size of each synonym set. (a) $n = 10, m = 20, w = 5$, (b) $p = 10\%, m = 20, w = 5$, (c) $p = 10\%, n = 10, w = 5$, and (d) $p = 10\%, n = 10, m = 20$.

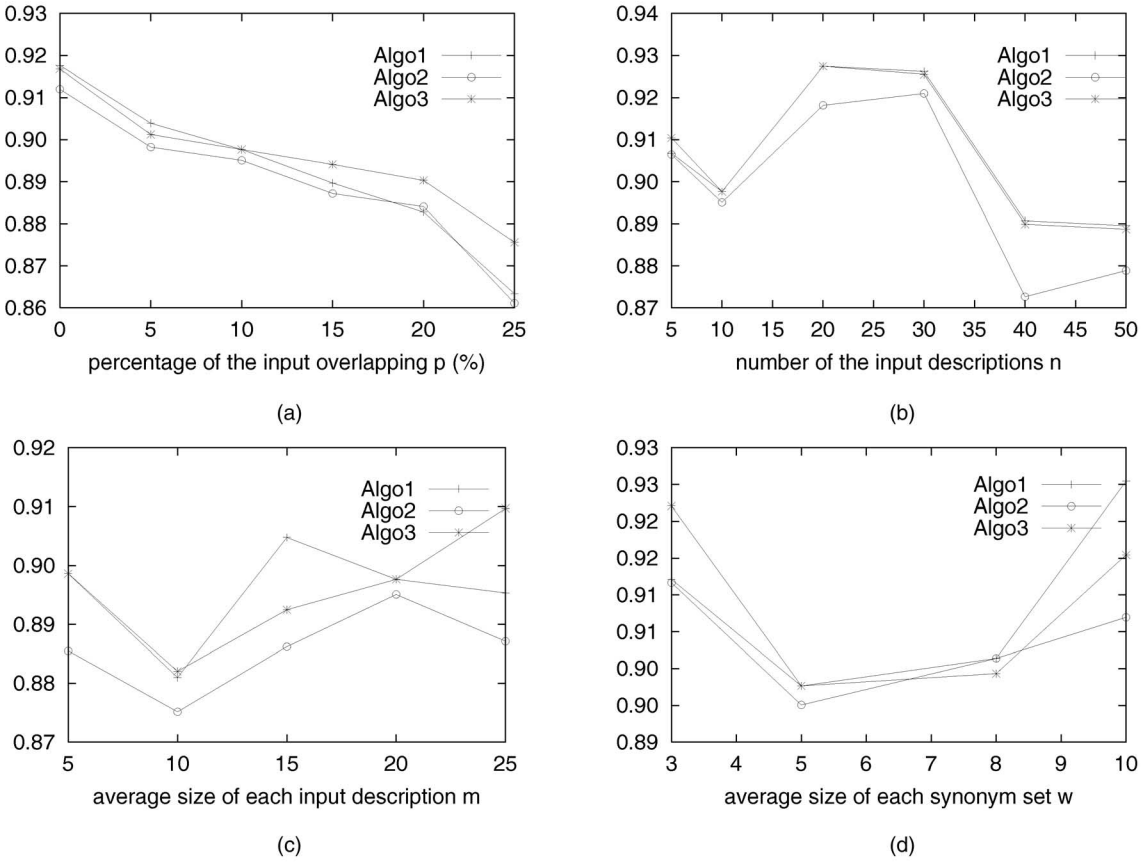

Fig. 7. Similarity of solution versus different values of percentage of input overlap, number of input descriptions, average size of input descriptions, and average size of each synonym set. (a) $n = 10, m = 20, w = 5$, (b) $p = 10\%, m = 20, w = 5$, (c) $p = 10\%, n = 10, w = 5$, and (d) $p = 10\%, n = 10, m = 20$.

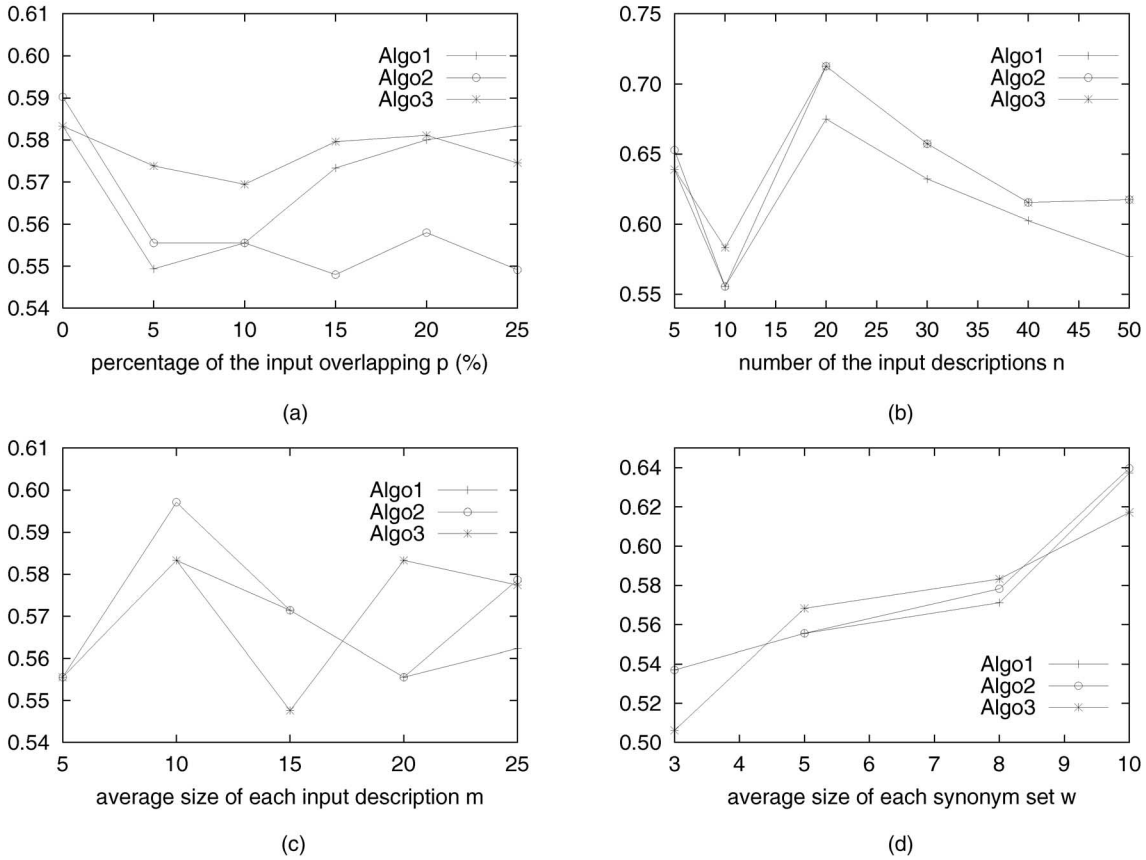

Fig. 8. Popularity of solution versus different values of percentage of input overlap, number of input descriptions, average size of input descriptions, and average size of each synonym set. (a) $n = 10, m = 20, w = 5$, (b) $p = 10\%, m = 20, w = 5$, (c) $p = 10\%, n = 10, w = 5$, and (d) $p = 10\%, n = 10, m = 20$.

descriptions. We arrange the PC descriptions according to different values of the four parameters above. We let $p = 10\%$, $n = 10$, $m = 20$, and $w = 5$ be the average case for the input descriptions. We make the following observations in the experiments:

1. The goodness of each solution generated by each algorithm increases with m when m is larger than 10, while it decreases when p increases. Parameters n and w do not have direct or inverse effects on the goodness of solutions (Fig. 6).
2. The similarity of each solution generated by each algorithm increases with m, while it decreases when p increases. Parameters n and w do not have direct or inverse effects on the similarity of solutions (Fig. 7).
3. The popularity of each solution generated by each algorithm increases with w. Parameters p, n, and m do not have direct or inverse effects on the popularity of solutions (Fig. 8).
4. The evenness of each solution generated by each algorithm increases with m, while it decreases when p or w increases. Parameter n does not have direct or inverse effects on the evenness of solutions (Fig. 9).

With regard to Observation 1 (Fig. 6), we note that:

- Algorithm I yields the best performance among the three algorithms when n is larger than 20; the goodness of the partition generated by Algorithm I in this case is the highest among the three. Algorithm I gives the second best performance in the other situations.
- Algorithm II always yields the worst performance in all situations. This is a result of the way it constructs columns for product schemas. Since it chooses the most semantically coherent synonym sets with the union of the synonym sets already in the column, it is more likely that the evenness of the columns is zero, i.e., not all the synonym sets in one column intersect. This reduces the goodness of the solution.
- Algorithm III yields the best performance in all situations. This is a result of the way it constructs columns for product schemas. Algorithm III is "strict" in selecting synonym sets for a column (see Section 5.1); it selects the synonym set having the largest intersection instead of the union with the existing synonym sets in the column. Thus, it selects synonym sets in a narrower range but this ensures the evenness of the partition to be larger than zero.

With regard to Observation 2 (Fig. 7), we note that:

- Algorithms I, II, and III have quite similar performance, i.e., similar values of similarity.
- Algorithm I has the best performance, i.e., the highest similarity, no matter what values n and m have. It also gives the best performance when p is smaller than 10 percent or when w is larger than five.
- Algorithm II always has the worst performance in all situations because it selects the synonym set having

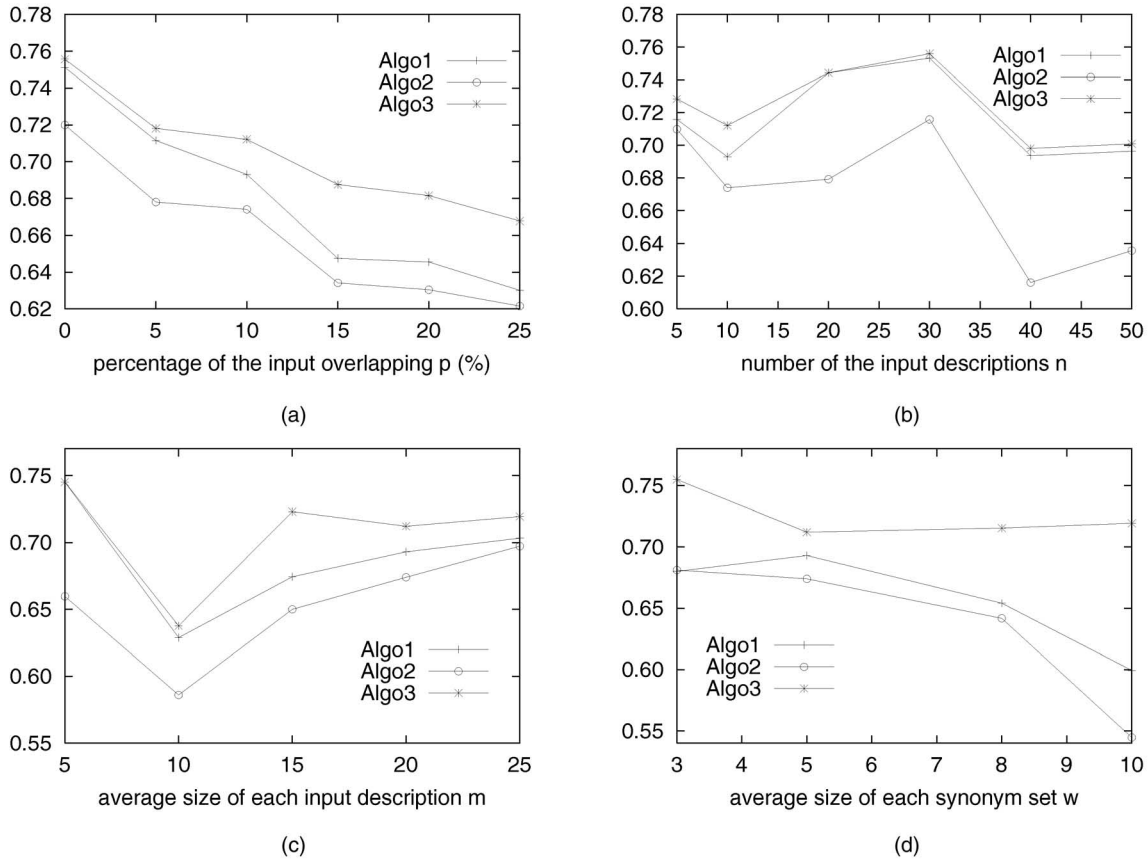

Fig. 9. Evenness of solution versus different values of percentage of input overlap, number of input descriptions, average size of input descriptions, and average size of each synonym set. (a) $n = 10, m = 20, w = 5$, (b) $p = 10\%, m = 20, w = 5$, (c) $p = 10\%, n = 10, w = 5$, and (d) $p = 10\%, n = 10, m = 20$.

the largest union instead of intersection with the existing synonym sets in the column. This results in columns having more diverse synonym sets, which reduce the similarity of the whole partition.

- Algorithm III has the best performance for all values of n. It also has the best performance when p is larger than 10 percent and when w is smaller than five.

With regard to Observation 3 (Fig. 8), we note that:

- Algorithm I yields the best performance, i.e., the highest popularity, when p is larger than 20 percent, or when w is larger than nine. It gives the worst performance when p is less than 10 percent, when n is larger than 10, or when w is between five and eight.
- Algorithm II gives the best performance when m is smaller than 15, when n is larger than 20, when w is smaller than four, or when w is larger than nine. It gives the worst performance when p is larger than 10 percent.
- Algorithm III gives the best performance when p is between 2 percent and 20 percent, when n is larger than two, when m is larger than 17 or when w is between five and eight. It gives the worst performance when m is smaller than 17, when w is smaller than four or when w is larger than nine.

With regard to Observation 4 (Fig. 9), we note that:

- In all the situations, Algorithm III has the best performance, i.e., the highest evenness, while Algorithm II has the worst and Algorithm I has the best.

In summary, we may make the following conclusions from the experiments:

- Algorithm III has the best performance in most cases. It is capable of solving PSI and it maximizes the evenness of the solutions. Algorithms I and II maximize the similarity and popularity of the solutions to some extent, but not as much as what Algorithm III does to the evenness of the solutions. We should thus adopt Algorithm III in our implementation of directory agents in our ABECOS system.
- Parameter p, the overlap of synonym sets within one description, inversely affects on the performance of each algorithm in all the cases except for the popularity of the solutions. Thus, we should reduce p as much as possible.
- Parameter n, the number of sellers, does not have any direct or inverse effects on the performance of each algorithm in all the cases. Thus, the algorithms are suitable for large number of sellers.
- Parameter m, the size of product description, directly affects on the performance of each algorithm with respect to the goodness and similarity; in the other cases, it does not affect the performance too much.

- Parameter w, the size of each synonym set, directly affects the performance of each algorithm with respect to the goodness and popularity; it has a direct effect on the performance of each algorithm with respect to the evenness of the solutions. Thus, we should raise w as much as possible.

Therefore, to increase the performance of product schema integration, we have some advice for sellers who provide synonym sets for their product descriptions:

- Sellers should try to avoid using identical attribute names in different synonym sets of a single product description (in order to reduce the value of parameter p). For example, when p is less than 10 percent, the goodness of the integration result is about 72 percent, which is quite good (see Fig. 6). This means that if there is a total of 50 attribute names in use by one seller, the number of attribute names that appear more than once should not be larger than five.
- Sellers are advised to make the size of synonym sets as large as possible (to raise the value of w). For example, when the average size of synonym sets is around eight, the integration result has a very high goodness value (see Fig. 6). However, sometimes it is not easy to provide many synonyms for one attribute name, say *CPU*. From the experiments, we know that the average size of synonym sets should be at least three in order to secure the performance of integration.

6 Conclusions and Future Work

In this paper, we address the problem (PSI) of integrating heterogeneous product descriptions in an agent-based electronic commerce system. Although several solutions to database schema integration have been proposed in multidatabase research work, PSI is distinct in the E-commerce context for several reasons: the need to overcome limited knowledge about local product databases, the requirement of automated integration, the large number of product schemas, and the evolution of local schemas. This paper describes our preliminary work in defining, analyzing, and solving PSI in the ABECOS commerce system [21], [23], [24], [37], [38]. We adopt synonym sets to represent the semantic information of local product schemas. Thus, the problem is defined as finding the optimum synonym set matching among local product schemas. We also prove that the problem is NP-complete. Experiments show that our approximation algorithm is feasible and computes solutions for PSI.

Future work can be focused on the following issues: First, it is not clear at this moment how complex attribute relationships such as *aggregate* relationship can be handled. For example, in the descriptions of PCs, an attribute *memory* has an aggregate relationship with another two attributes *standard memory* and *extensible memory*. Second, we should consider more complex product schemas such as hierarchical schemas. Attributes in a *hierarchical* schema are divided into several abstract levels; some attributes may have a set of child attributes. To integrate such schemas, we identify the correspondence of different attribute levels, as well as the attributes within each level. Third, the maintenance of integration results is also an important issue to be considered. To avoid recomputing the global schema from scratch, we should adopt incremental algorithms for adding, deleting, and updating local schemas. Fourth, we should study product schema integration in the context of XML documents. Last, further theoretical investigation on the approximation algorithms will be performed to determine their performance bounds. We will report the results of our work in future papers.

Appendix

Complexity Results

The problem of 3-Dimensional Matching (3DM) is defined as follows [11]:

Definition 4 (3DM). *Given a set $S \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets having the same number q of elements, does there exist a subset $S' \subseteq S$ such that $|S'| = q$ and no two elements of S' agree in any coordinate?*

We generalize 3DM to nDM as follows and show that it is NP-complete:

Definition 5 (nDM). *Given a set $S \subseteq W_1 \times W_2 \times \cdots \times W_n$, where W_i $(1 \leq i \leq n)$ are disjoint sets having the same number q of elements, does there exist a subset $S' \subseteq S$ such that $|S'| = q$ and no two elements of S' agree in any coordinate?*

Theorem 2. nDM $\in NP$*-complete.*

Proof. Since 3DM is NP-complete, let us suppose nDM, $n \geq 3$ to be NP-complete. We shall prove that (n+1)DM is also NP-complete under this assumption.

First, we consider an instance of nDM. Let $W_1, W_2, \ldots, W_n$ be n disjoint sets each having q elements, such that $W_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,q}\}$ for $1 \leq i \leq n$. Let n-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k})$, $1 \leq i, j, k \leq q$, be an arbitrary element of the set $S \subseteq W_1 \times W_2 \times \cdots \times W_n$. Let $W_1, W_2, \ldots, W_n$, W_{n+1} be the $n+1$ disjoint sets in the corresponding instance of (n+1)DM, where $W_1, W_2, \ldots, W_n$ are the same as those in nDM and $W_{n+1} = \{x_1, x_2, \ldots, x_q\}$.

Now, let us construct set T, the corresponding set of S in nDM, $T \subseteq W_1 \times W_2 \times \cdots \times W_n \times W_{n+1}$ as follows: If $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}) \in S$ for $1 \leq i, j, k \leq q$, then $(n+1)$-tuples $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_1)$, $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_2) \ldots$, $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_q)$ are elements of set T. We shall prove that each of these two instances is satisfiable if and only if the other one is satisfiable.

If the instance of nDM is satisfiable, then there exists a set $S' \subseteq S$, which satisfies the following conditions:

1. $|S'| = q$,
2. $\forall s_1 = (w_{1,i}, w_{2,j}, \ldots, w_{n,k})$ and $s_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}) \in S'$ for $1 \leq i, j, k, i', j', k' \leq q$ if $s_1 \neq s_2$, then $w_{1,i} \neq w_{1,i'}, w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'}$.

For each n-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}) \in S'$, we extend it to an $(n+1)$-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_i)$. This $(n+1)$-tuple is an element of T. All such extended $(n+1)$-tuples compose a set $T' \subseteq T$, and T' satisfies the following conditions:

1. $|T'| = q$,
2. $\forall t_1 = (w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_i)$ and $t_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}, x_{i'}) \in T'$ for $1 \leq i, j, k, i', j', k' \leq q$ if $t_1 \neq t_2$, then

$$w_{1,i} \neq w_{1,i'}, w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'},\ x_i \neq x_{i'}.$$

The first condition is satisfied because $|S'| = q$ and $|T'| = |S'|$. From the second condition satisfied by S' and the fact that $W_1, W_2, \ldots, W_n$, W_{n+1} are $n+1$ disjoint sets, we can see that the second condition for T' is also satisfied. Thus, T' is the solution of (n+1)DM.

Conversely, if the instance of (n+1)DM is satisfiable, then there exists a set $T' \subseteq T$, and T' satisfies the following conditions:

1. $|T'| = q$,
2. $\forall t_1 = (w_{1,i}, w_{2,j}, , w_{n,k}, x_\ell)$ and $t_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}, x_{\ell'}) \in T'$ for $1 \leq i, j, k, \ell, i', j', k', \ell' \leq q$ if $t_1 \neq t_2$, then $w_{1,i} \neq w_{1,i'}, w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'}$, $x_\ell \neq x_{\ell'}$.

For each $(n+1)$-tuple $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}, x_\ell)$, we just discard its last element, i.e., x_ℓ to make it an n-tuple like $(w_{1,i}, w_{2,j}, \ldots, w_{n,k}) \in S$. Thus, all such n-tuples are composed of a set $S' \subseteq S$, and S' satisfies the following conditions:

1. $|S'| = q$,
2. $\forall s_1 = (w_{1,i}, w_{2,j}, \ldots, w_{n,k})$ and $s_2 = (w_{1,i'}, w_{2,j'}, \ldots, w_{n,k'}) \in S'$ for $1 \leq i, j, k, i', j', k' \leq q$ if $s_1 \neq s_2$, then $w_{1,i} \neq w_{1,i'}, w_{2,j} \neq w_{2,j'}, \ldots, w_{n,k} \neq w_{n,k'}$.

The first condition is satisfied because $|S'| = |T'|$. Since the second condition is satisfied by T' each n-tuple in S' is the first n elements of the $(n+1)$-tuple in T', we can see that the second condition is also satisfied by S'. Thus, the instance of nDM is satisfiable.

Therefore, we have proven that, if (n+1)DM is NP-complete, then nDM ($n \geq 3$) is NP-complete. Since 3DM is NP-complete, from the mathematical induction we conclude that nDM ($n \geq 3$) is NP-complete. □

REFERENCES

[1] N. Adam and Y. Yesha, "Strategic Directions in Electronic Commerce and Digital Libraries: Towards a Digital Agora," *ACM Computing Surveys,* vol. 28, no. 4, pp. 818–835, Dec. 1996.

[2] C. Batini, M. Lenzerini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys,* vol. 18, no. 4, pp. 323–364, Dec. 1986.

[3] P. Bernstein, M. Brodie, S. Ceri, and D. DeWitt, The Asilomar Report on Database Research, Sept. 1998, http://www.research.microsoft.com/Gray/Asilomar_DB_98.html.

[4] S. Bressan, C.H. Goh, K. Fynn, M. Jakobisiak, K. Hussein, H. Kon, T. Lee, S. Madnick, T. Pena, J. Qu, A. Shum, and M. Skegel, "The Context Interchange Mediator Prototype," *Proc. ACM SIGMOD/PODS Joint Conf.,* May 1997.

[5] Common Business Library (CBL), http://www.veosystems.com/xml/cbl/cbl.html.

[6] D. Clements, M. Ganesh, S.-Y. Hwang, E.-P. Lim, , K. Mediratta, J. Srivastava, and H.-R. Yang, "Myriad: Design and Implementation of Federated Database Prototype," *Proc. Int'l Conf. Computer Systems and Education,* June 1994.

[7] C. Collet, M.N. Huhns, and W.-M. Shen, "Resource Integration Using a Large Knowledge Base in Carnot," *Computer,* vol. 24, no. 12, pp. 55–62, Dec. 1991.

[8] CommerceNet, http://www.commerce.net. 1999.

[9] The eCo Framework Project, http://www.commerce.net/projects/currentprojects/eco, 1999.

[10] D. Florescu, A. Levy, and A. Mendelzon, "Database Techniques for the World-Wide Web: A Survey," *ACM Special Interest Group on Management of Data Record (SIGMOD Record '98),* vol. 27, no. 3, Sept. 1998.

[11] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* p. 46, W.H. Freeman and Co., 1979.

[12] J.N. Gray, Database Systems: A Textbook Case of Research Paying Off, http://www.cs.washington.edu/homes/lazowska/cra/database.html, 1999.

[13] S. Hayne and S. Ram, "Multi-User View Integration System (MUVIS): An Expert System for View Integration," *Proc. IEEE Sixth Int'l Conf. Data Eng. (ICDE '90),* pp. 402–409, Feb. 1990.

[14] M. Hearst, "Trends and Controversies: Information Integration," *IEEE Intelligent Systems J.,* pp. 12–24, Sept./Oct. 1998.

[15] R.D. Holowczak and W.-S. Li, "A Survey on Attribute Correspondence and Heterogeneity Metadata Representation," *Proc. First IEEE Metadata Conf.,* Ap. 1996.

[16] Internet Content Exchange (ICE), http://www.vignette.com/Products/ice, 1999.

[17] J.A. Larson, S.B. Navathe, and R. Elmasri, "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," *IEEE Trans. Software Eng.,* vol. 15, no. 4, pp. 449–463, Apr. 1989.

[18] D.B. Lenat, "Cyc: A Large-Scale Investment in Knowledge Infrastructure," *Comm. ACM,* pp. 33–38, Nov. 1995.

[19] W.-S. Li and C. Clifton, "Semantic Integration in Heterogeneous Databases Using Neural Networks," *Proc. 20th Int'l Conf. Very Large Data Bases,* Sept. 1994.

[20] W.-S. Li and C. Clifton, "Using Field Specifications to Determine Attribute Equivalence in Heterogeneous Databases," *Proc. IEEE Third Int'l Workshop Research Issues on Data Eng.: Interoperability in Multidatabase Systems,* pp. 174–177, Apr. 1993.

[21] E.-P. Lim and W.-K. Ng, "Overview of the Agent-Based Electronic Commerce (ABECOS) Project," *IEEE Data Eng. Bull.,* vol. 23, no. 1, pp. 49–54, Mar. 2000.

[22] S. Navathe, R. Elmasri, and J. Larson, "Integrating User Views in Database Design," *IEEE Computer,* vol. 19, no. 1, pp. 50–62, Jan. 1986.

[23] W.-K. Ng, G. Yan, and E.-P. Lim, "Heterogeneous Product Description in Electronic Commerce," *ACM SIGecom Exchanges,* vol. 1, no. 1, pp. 7–13, Aug. 2000.

[24] W.-K. Ng, G. Yan, and E.-P. Lim, "Standardization and Integration in Business-to-Business electronic Commerce," *IEEE Intelligent Systems,* Jan./Feb. 2001.

[25] Open Buying on the Internet (OBI), http://www.openbuy.org, 2000.

[26] W.J. Premerlani and M.R. Blaha, "An Approach for Reverse Engineering of Relational Databases," *Comm. ACM,* vol. 37, no. 5, pp. 42–49, May 1994.

[27] RosettaNet, http://www.rosettanet.org, 1999.

[28] RosettaNet Laptop Technical Specification, http://www. rosettanet. org/general/finished_projects/laptop. html, May 1998.

[29] G. Salton, *Automatic Text Processing,* chapter 10, Addison-Wesley Pub., 1989.

[30] P. Scheuermann, W.-S. Li, and C. Clifton, "Multidatabase Query Processing with Uncertainty in Global Keys and Attribute Values," *J. Am. Soc. for Information Science,* vol. 49, no. 3, pp. 283–301, 1998.

[31] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases," *ACM Computing Surveys,* vol. 22, no. 3, pp. 183–236, Sept. 1990.

[32] A. Sheth, J. Larson, A. Cornelio, and S.B. Navathe, "A Tool for Integrating Conceptual Schemas and User Views," *Proc. IEEE Fourth Int'l Conf. Data Eng. (ICDE '88),* Feb. 1988.

[33] A. Silberschatz and S. Zdonik, "Strategic Directions in Database Systems—Breaking Out of the Box," *ACM Computing Surveys,* vol. 28, no. 4, pp. 764–778, Dec. 1996.

[34] A. Silberschatz, M. Stonebraker, and J. Ullman, "Database Research: Achievements and Opportunities Into the 21st Century," *ACM Special Interest Group on Management of Data Record (SIGMOD Record '96),* vol. 25, no. 1 Mar. 1996.

[35] J.M. Smith, P.A. Bernstein, U. Dayal, N. Goodman, T. Landers, T. Lin, and E. Wang, "Multibase-Integrating Heterogeneous Distributed Database Systems," *Proc. Nat'l Computer Conf.,* pp. 487–499, 1981.

[36] Veo Systems, Inc., http://www.veosystems.com, 1999.

[37] G. Yan, W.-K. Ng, and E.-P. Lim, "Toolkits for a Distributed, Agent-Based Web Commerce System," *Proc. Int'l IFIP Working Conf. Trends in Distributed Systems for Electronic Commerce (TrEC '98)*, June 1998.
[38] G. Yan, W.-K. Ng, and E.-P. Lim, "Incremental Maintenance of Product Schema in Electronic Commerce—A Synonym-Based Approach," *Second Int'l Conf. Information, Comm., and Signal Processing (ICICS '99)*, Dec. 1999.
[39] C. Yu, W. Sun, S. Dao, and D. Keirsey, "Determining Relationships Among Attributes for Interoperability of Multi-Database System," *Proc. Workshop Multi-Database and Semantic Interoperability*, Nov. 1990.

Guanghao Yan recived the BSc degree from Peking University, China, in 1997 and the MSc degree from the Nanyang Technological University, Singapore, in 1999. He is currently a graduate student at the Department of Computer Science, the State University of New York at Stony Brook. He works in the area of data integration for electronic product descriptions. He is a student member of the ACM and IEEE Computer Society.

Wee Keong Ng received the MSc and PhD degrees from the University of Michigan, Ann Arbor in 1994 and 1996, respectively. He is an assistant professor of the School of Computer Engineering at the Nanyang Technological University, Singapore. He works and publishes widely in the areas of Web warehousing, information extraction, electronic commerce and data mining. He has organized and chaired international workshops, including tutorials, and has actively served in the program committees of numerous international conferences. He is a member of the ACM and the IEEE Computer Society.

Ee-Peng Lim is an associate professor of the School of Computer Engineering at the Nanyang Technological University, Singapore. His research interests are in Web warehousing, database integration, and digital libraries. He is currently the director of the Centre for Advanced Information Systems (CAIS), a research centre focusing on web computing and database research. He is also the guest editor of the special issue on mobile commerce of the *Journal of Database Management*. His research has been published in a number of international journals including *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, *Decision Support Systems*, and *Distributed and Parallel Databases*. He is a member of the ACM and a senior member of the IEEE.