Data Privacy Preserving using Random Grouping

Edara Rudrani Devi[1], Mr K.Raja Sekhar[2]

[1] Student, K.L.University,Vaddeswaram,Guntur(dt),, Andhra Pradesh, India
[2] Assistant Professor, K.L.University,Vaddeswaram,Guntur(dt), Andhra Pradesh, India

Abstract: Data Anonymization mitigates privacy and security concerns and comply with legal requirements. Data publishing involves balancing many aspects such as privacy, security, and legal interests. Anonymization countermeasures are still vulnerable as they still can expose protected information in released information. Earlier k-anonymity, l-diversity procedures ensured privacy, their limitations such as inability to handle high dimensional data, failure to maintain clear separation between quasi-identifying attributes and sensitive attributes forced researchers to explore other alternatives. Later a novel technique called Slicing, which partitions the data both horizontally and vertically is developed that can handle high dimensional data and promoting better data utility. Slicing algorithm consists of three phases: attribute partitioning, column generalization, and tuple partitioning. It's sensitive attribute disclosures are based on random grouping which is not very effective as randomly generating the associations between column values of a bucket significantly lowers data utility. In our quest for efficiency we propose to replace random grouping with an optimized tuple grouping algorithms such as Tuple Space Search algorithm that are driven by hashing techniques. Experiments on varying datasets indicated better data privacy, utility and anonymity compared to prior approaches. The computed and obtained anonymized data from high dimensional sensitive attributes based on the proposed technique offers significant performance gains. A feasible practical implementation on dynamic data validates our claim.

Keywords—Privacy preservation, data anonymization, data publishing, data security

I. INTRODUCTION

Data mining has emerged as a means for identifying patterns and trends from a large amount of data. We need to collect data, to conduct data mining computations. Without privacy concerns, data can be directly collected. The privacy preserving data mining problem has gained considerable importance in recent years because of the vast amounts of personal data about individuals stored at different commercial vendors and organizations. Micro data contains records each of which contains information about an individual entity, such as a person, a household, or an organization. Most of the microdata anonymization techniques have been proposed. The most popular ones are generalization, for k-anonymity and bucketization, for diversity. Microdata publishing enables researchers and policy-makers to analyze the data and learn important information. Privacy is a key parameter in sensitive attribute disclosures. For privacy in Microdata publishing generalization and bucketization techniques based on k-anonymity, l-diversity approaches were used. Generalization fails to handle high dimensional data. Bucketization fails to maintain clear separation between quasi-identifying attributes and sensitive attributes. we will analyze the k-anonymity approach for the high dimensional case. k-anonymity protects against identity disclosures, but it does not provide sufficient protection against attribute disclosures. l-diversity protects against attribute disclosures but fails to prevent probabilistic attacks. So a better system is required that can with stand these failures and offers significant performance rise. For privacy in Microdata publishing a novel technique called slicing is used, which partitions the data both horizontally and vertically. Slicing preserves better data utility than generalization and can be used for membership disclosure protection. Slicing can handle high-dimensional data. For Sliced data to obey the diversity requirement random grouping methods were used. Slicing algorithm consists of three phases: attribute partitioning, column generalization, and tuple partitioning. Involves the following procedures to attain data anonymity

- Attribute Partition and Columns
- Tuple Partition and Buckets
- Slicing
- Column Generalization

These methods compromise on overall data utility to maintain diversity requirement. A better system is required that can that can with stand high-dimensional data handling and sensitive attribute disclosure failures.In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. Generalization transforms the QI-values in each bucket into “less specific but semantically consistent” values so that tuples in the same bucket cannot be distinguished by their QI values. Where as in bucketization, one separates the SAs from the QIs by randomly permuting the SA values in each bucket. In generalization for kanonymity losses considerable amount of information, especially for high-dimensional data due to three reasons. First, generalization for k-anonymitysuffers from the curse of dimensionality. For effective generalization , records in the same bucket must be close to each other so that generalizing the records would not lose too much information. Hence, for the high dimensional data, most data points have similar distances with each other, forcing a great amount of generalization to satisfy k-anonymity even for relatively small k’s. Second, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution. This assumpts that every value in a generalized interval/set is equally possible.Hence it significantly reduces the generalized data for data utility. Third, because each attribute is generalized separately, correlations between different attributes are lost. The data analyst has to assume that every possible combination of attribute values is equally possible in attribute correlations on the generalized table. While bucketization has better data utility than generalization, it has several limitations. Initially bucketization does not prevent membership disclosure , while bucketization publishes the QI values in their original forms. A microdata usually contains many other attributes besides those three attributes. Which refers to the membership information of most individuals can be inferred from the bucketized table. Secondary as the bucketization requires a clear separation between QIs and SAs. However it is unclear which attributes are QIs and which are SAs in many data sets. Next by separating the sensitive attribute from the QI attributes, bucketization breaks the attribute correlations between the QIs and the SAs. We introduce a novel data anonymization technique called slicing to improve the current state of the art. The data has been partitioned horizontally and vertically by the slicing. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Every coloumn has highly correlated with a subset of attributes. Horizontal partitioning is done by grouping tuples into buckets. Finally, values in each column are randomly permutated (or sorted) to break the linking between different columns in the each bucket.

The basic idea of slicing is to break the association cross columns, but to preserve the association within each column. Slicing preserves utility because it groups highly correlated attributes together, and preserves the correlations between such attributes. It protects privacy because it breaks the associations between uncorrelated attributes, which are infrequent and thus identifying. When the data set contains QIs and one SA, bucketization has to break their correlation. slicing, on the other hand, can group some QI attributes with the SA, preserving attribute correlations with the sensitive attribute. slicing provides privacy protection that the slicing process ensures that for any tuple, there are generally multiple matching buckets. Consider a tuple $t = \langle v_1, v_2, \ldots, v_c \rangle$, ' , where c is the number of columns and vi is the value for the ith column, a bucket is a matching bucket for t if and only if for each i ($1 \le i \le c$), vi appears at least once in the i’th column of the bucket. At the same time, a matching bucket can be due to containing other tuples each of which contains some but not all vi’s.

We present a novel technique called slicing for privacy-preserving data publishing. Our contributions include the following.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

First, we introduce slicing as a new technique for privacy preserving data publishing. Comparing to the generalization and bucketization, scliling has the advantage. It preserves better data utility than generalization and more attribute correlations with the SAs than bucketization.

Second, we show that slicing can be effectively used for preventing attribute disclosure, based on the privacy requirement of *l* -diversity. Hence , We introduce a notion called *l* -diverse slicing, which ensures that the adversary cannot learn the sensitive value of any individual with a probability greater than 1/*l*.

Third, we develop an efficient algorithm for computing the sliced table that satisfies ‘-diversity. Our algorithm partitions attributes into columns, applies column generalization, and partitions tuples into buckets. The associations between uncorrelated attributes are broken .This provides better privacy as the associations between such attributes are lessfrequent and potentially identifying.

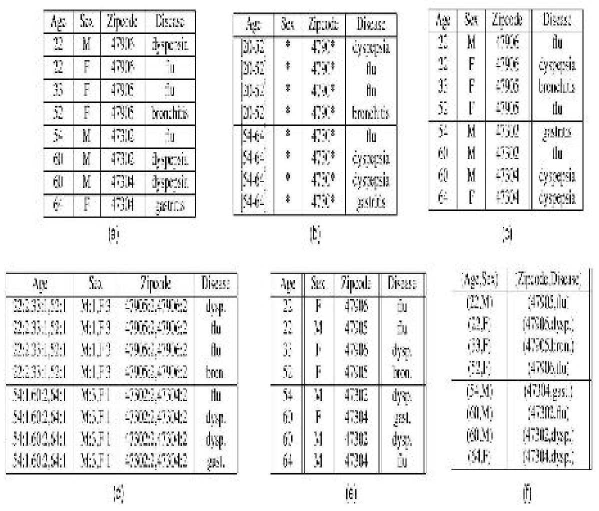
Fourth, we describe the intuition behind membership disclosure and explain how slicing prevents membership disclosure. A bucket of size k can potentially match k^c tuples where c is the number of columns. Because only k of the kc tuples are actually in the original data, the existence of the other k^c - k tuples hides the membership information of tuples in the original data.

II. RELATED WORK

SLICING

We first give an example to illustrate slicing. We then formalize slicing, compare it with generalization and bucketization, and discuss privacy threats that slicing can address

TABLE 1
An Original Microdata Table and Its Anonymized Versions Using Various Anonymization Techniques



(a)

Age	Sex	Zipcode	Disease
22	M	47906	dyspepsia
22	F	47906	flu
33	F	47905	flu
52	F	47905	bronchitis
54	M	47302	flu
60	M	47302	dyspepsia
60	M	47304	dyspepsia
64	F	47304	gastritis

(b)

Age	Sex	Zipcode	Disease
[20-52]	*	4790*	dyspepsia
[20-52]	*	4790*	flu
[20-52]	*	4790*	flu
[20-52]	*	4790*	bronchitis
[54-64]	*	4730*	flu
[54-64]	*	4730*	dyspepsia
[54-64]	*	4730*	dyspepsia
[54-64]	*	4730*	gastritis

(c)

Age	Sex	Zipcode	Disease
22	M	47906	flu
22	F	47906	dyspepsia
33	F	47905	bronchitis
52	F	47905	flu
54	M	47302	gastritis
60	M	47302	flu
60	M	47304	dyspepsia
64	F	47304	dyspepsia

(d)

Age	Sex	Zipcode	Disease
22:2,33:1,52:1	M:1,F:3	47905:2,47906:2	dysp.
22:2,33:1,52:1	M:1,F:3	47905:2,47906:2	flu
22:2,33:1,52:1	M:1,F:3	47905:2,47906:2	flu
22:2,33:1,52:1	M:1,F:3	47905:2,47906:2	bron.
54:1,60:2,64:1	M:3,F:1	47302:2,47304:2	flu
54:1,60:2,64:1	M:3,F:1	47302:2,47304:2	dysp.
54:1,60:2,64:1	M:3,F:1	47302:2,47304:2	dysp.
54:1,60:2,64:1	M:3,F:1	47302:2,47304:2	gast.

(e)

Age	Sex	Zipcode	Disease
22	F	47906	flu
22	M	47905	flu
33	F	47906	dysp.
52	F	47905	bron.
54	M	47302	dysp.
60	F	47304	gast.
60	M	47302	dysp.
64	M	47304	flu

(f)

(Age,Sex)	(Zipcode,Disease)
(22,M)	(47905,flu)
(22,F)	(47906,dysp.)
(33,F)	(47905,bron.)
(52,F)	(47906,flu)
(54,M)	(47304,gast.)
(60,M)	(47302,flu)
(60,M)	(47302,dysp.)
(64,F)	(47304,dysp.)

(a) The original table, (b) the generalized table, (c) the bucketized table, (d) multiset-based generalization, (e) one-attribute-per-column slicing, (f) the sliced table.

The above table shows an example microdata table and its anonymized versions using various anonymization techniques.

The original table is shown in Table 1a. The three QI attributes are {Age; Sex;Zipcode} and the sensitive attribute SA is Disease. Table lb shows generalized table that satisfies 4-anonymity. A bucketized table that satisfies 2-diversity is shown in Table 1c and a generalized table where each attribute value is replaced with the multiset of values in the bucket is shown in Table 1d, and two sliced tables are shown in Tables 1e and 1f.

The first partition of attributes in slicing is to coloumns . Each column contains a subset of attributes. This vertically partitions the table. We can observe, the sliced table in Table 1f contains two columns: the first column contains {Age; Sex} and the second column contains {Zipcode; Disease}. Slicing also partition tuples into buckets. Each bucket contains a subset of tuples. This horizontally partitions the table. Within each bucket, values in each column are randomly permutated to break the linking between different columns.

Formalization of Slicing

Let T be the microdata table to be published. T contains d attributes: A = {A1;A2; . . .;Ad} and there attributes domains are (D[A1];D[A2]; . . .;D[A]}. Then the tuple *t* can be represented as t

={t[A1]; t[A2]; . . . ; t[Ad]} where t = [Ai] (a ≤ I ≤ d) is the Ai value of t.

Attribute Partition and Columns : An attribute partition consists of several subsets of A, such that each attribute belongs to exactly one subset. Each subset of attributes is called a column. For simplicity of discussion, we consider only one sensitive attribute S. If the data contain multiple sensitive attributes, one can either consider them separately or consider their joint distribution. Exactly one of the c columns contains S.

Tuple Partition and Buckets : A tuple partition consists of several subsets of T, such that each tuple belongs to exactly one subset. Each subset of tuples is called a bucket.

Slicing : Given a microdata table T, a slicing of T is given by an attribute partition and a tuple partition. Often times, slicing also involves column generalization.

Column Generalization : Column generalization ensures that one column satisfies the k-anonymity requirement. Now we can say it is a multidimensional encoding and can be used as an additional step in slicing. Generally slicing algorithm consists of the following three phases: attribute partition, column generalization, and tuple partition.

Matching Buckets : Let {C1; C2; . . . ; Cc } be the c columns of a sliced table. Let t be a tuple, and t[Ci] be the Ci value of t. Let B be a bucket in the sliced table, and B[Ci] be the multiset of Ci values in B. We say that B is a matching bucket of t iff for all 1 ≤ i ≤ c, t[Ci] Є B[Ci].

Comparison with Generalization

There are several types of recodings for generalization. The recoding that preserves the most information is local recoding. We now show that slicing preserves more information than such a local recoding approach, assuming that the
same tuple partition is used. Using a generalized value to replace more specific attribute values, one uses the multiset of exact values in each bucket. Table 1b is a generalized table, and Table 1d is the result of using multisets of exact values rather than generalized values. The multiset of exact values provides more information about the distribution of values
in each attribute than the generalized interval. We observe that this multiset-based generalization is equivalent to a trivial slicing scheme where each column contains exactly one attribute, because both approaches preserve the exact values in each attribute but break the association between them within one bucket. Consider that the table le is equivalent to table ld and now compare Table 1e with the sliced table shown in Table 1f, we observe that while one-attribute-per-column slicing reserves attribute distributional information. It does not preserve attribute correlation, because each attribute is in its own
column. In slicing, one groups correlated attributes together in one column and preserves their correlation.

Another important advantage of slicing is its ability to handle high-dimensional data. By partitioning attributes into columns, slicing reduces the dimensionality of the data. Each column of the table can be viewed as a subtable with a lower dimensionality. The idea of slicing is to achieve a better trade-off between privacy and utility by preserving correlations between highly correlated attributes and breaking correlations between uncorrelated attributes.

Comparison with Bucketization

To compare slicing with bucketization, first we want to note that the bucketization can be viewed as a special case of slicing. It contain exactly two columns: one column contains only the SA, and the other contains all the QIs. The advantages of slicing over bucketization can be understood in the following explanation , Initially by partitioning attributes into more than two columns, slicing can be used to prevent membership disclosure. Second, unlike bucketization, which requires a clear separation of QI attributes and the sensitive attribute, slicing can be used without such a separation. Finally, by allowing a column to contain both some QI attributes and the sensitive attribute, attribute correlations between the sensitive attribute and the QI attributes are preserved. For workloads that consider attributes in isolation, one can simply publish two tables, one containing all QI attributes and one containing the sensitive attribute.

Privacy Threats

When publishing microdata, there are three types of privacy disclosure threats.

- *Membership disclosure.*
  When the data set to be published is selected from a large population and the selection criteria are sensitive
- *identity disclosure*
  It occurs when an individual is linked to a particular record in the released table. In some situations, one wants to protect against identity disclosure when the adversary is uncertain of membership. Here protection against membership disclosure helps protect against identity disclosure.
- *attribute disclosure*
  Identity disclosure leads to attribute disclosure. It occurs when new information about some individuals is revealed. we need to consider adversaries who already know the membership information. Once there is identity disclosure, an individual is reidentified and the corresponding sensitive value is revealed.

For slicing, we consider protection against membership disclosure and attribute disclosure. For data anonymized by bucketization iIt is a little unclear how identity disclosure should be defined. Because identity disclosure leads to attribute disclosure, protection against attribute disclosure is also sufficient protection against identity disclosure.

III. EXISTING SYSTEM

For privacy in Microdata publishing a novel technique called slicing is used, which partitions the data both horizontally and vertically. Slicing preserves better data utility than generalization and can be used for membership disclosure protection. Slicing can handle high-dimensional data. For Sliced data to obey the diversity requirement random grouping methods were used. Slicing algorithm consists of three phases: attribute partitioning, column generalization, and tuple partitioning. Involves the following procedures to attain data anonymity

- Attribute Partition and Columns
- Tuple Partition and Buckets
- Slicing
- Column Generalization

These methods compromise on overall data utility to maintain diversity requirement. A better system is required that can that can with stand high-dimensional data handling and sensitive attribute disclosure failures.

IV. PROPOSED SYSTEM

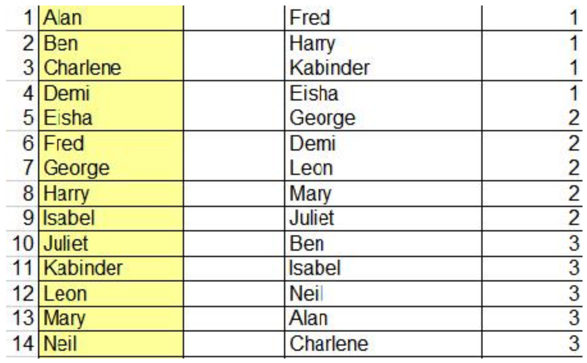
For privacy in Microdata publishing we still use slicing, which partitions the data both horizontally and vertically. Existing Slicing methods compromise on overall data utility to maintain diversity requirement. So we propose to replace random grouping with more effective tuple grouping algorithms such as Tuple Space Search algorithm based on hashing techniques. A tuple is defined as a vector of ***k*** lengths, where ***k*** is the number of fields in a filter. For example, in a 5-field filter set, the tuple [7, 12, 8, 0, 16] means the length of the source IP address prefix is 7, the length of the destination IP address prefix is 12, the length of the protocol prefix is 8 (an exact protocol value), the length of the source port prefix is 0 (wildcard or "don't care"), and the length of the destination port prefix is 16 (an exact port value). We can partition the filters in a filter set to the different tuple groups. Since the filtes in a same tuple group have the same tuple specification, they are mutual exclusive and none of them overlaps with others in this tuple group. Now we can perform the packet classification across all the tuples to find the best matched filter. If multiple tuple groups report matches, we resolve the best matched filter by comparing their priorities. The filters in a tuple can be easily organized into a hash table, where we use the tuple specification to extract the proper number of bits from each field as the hash key. This key can be used for faster indexing, sorting and aprimarily for accurate comparisons. The efficiency of tuple grouping algorithms enables its application to handle slicing problems that were previously prohibitive due to high-dimensional data handling and sensitive attribute disclosures.

A better system is required that can that can with stand high-dimensional data handling and sensitive attribute disclosure failures.In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. Generalization transforms the QI-values in each bucket into "less specific but semantically consistent" values so that tuples in the same bucket cannot be distinguished by their QI values. Where as in bucketization, one separates the SAs from the QIs by randomly permuting the SA values in each bucket. In generalization for k-anonymity losses considerable amount of information, especially for high-dimensional data due to three reasons. First, generalization for k-anonymitysuffers from the curse of dimensionality. For effective generalization , records in the same bucket must be close to each other so that generalizing the records would not lose too much information. Second, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution as we seen in section 2. Hence it significantly reduces the generalized data for data utility. Because each attribute is generalized separately, correlations between different attributes are lost. The data analyst has to assume that every possible combination of attribute values is equally possible in attribute correlations on the generalized table.

While bucketization has better data utility than generalization, it has several limitations. Initially bucketization does not prevent membership disclosure , while bucketization publishes the QI values in their original forms. A microdata usually contains many other attributes besides those three attributes. Which refers to the membership information of most individuals can be inferred from the bucketized table. Secondary as the bucketization requires a clear separation between QIs and SAs. However it is unclear which attributes are QIs and which are SAs in many data sets. Next by separating the sensitive attribute from the QI attributes, bucketization breaks the attribute correlations between the QIs and the SAs. We introduce a novel data anonymization technique called slicing to improve the current state of the art. The data has been partitioned horizontally and vertically by the slicing. Vertical partitioning is done by grouping attributes into columns based on the correlations among the attributes. Every coloumn has highly correlated with a subset of attributes. Horizontal partitioning is done by grouping tuples into buckets. Finally, values in each column are randomly permutated (or sorted) to break the linking between different columns in the each bucket. Experiments on varying datasets indicated better data privacy, utility and anonymity compared to prior approaches. The computed and obtained anonymized data from high dimensional sensitive attributes based on the proposed technique offers significant performance gains. A feasible practical implementation on dynamic data validates our claim.

V. PERFORMANCE ANALYSIS

We will provide some experimental analysis of the behavior of the different data sets. We will show that the behavior discussed earlier in this paper is exhibited over a variety of real and synthetic data sets. The synthetic data sets were generated as Gaussian clusters with randomly distributed centers in the unit cube.



1	Alan		Fred	1
2	Ben		Harry	1
3	Charlene		Kabinder	1
4	Demi		Eisha	1
5	Eisha		George	2
6	Fred		Demi	2
7	George		Leon	2
8	Harry		Mary	2
9	Isabel		Juliet	2
10	Juliet		Ben	3
11	Kabinder		Isabel	3
12	Leon		Neil	3
13	Mary		Alan	3
14	Neil		Charlene	3

Fig 1: Example of Random Group Generator

By verifying these results we are describing the efficient performance in the tuple grouping . Random number generators are like antibiotics. Every type of generator has its unwanted sideeffects. There are no safe generators. Good random number generators are characterized by theoretical support, convincing empirical evidence, and positive practical aspects. They will produce correct results in many, though not all, simulations. Open questions in this field concern reliable parallelization, the creation of good generators on demand, the sensitivity of transformation methods (to obtain nonuniform random numbers) to defects of the uniform random number generators, the classification of empirical tests, and the mathematical foundation of forecasting the empirical performance by theoretical figures of merit.

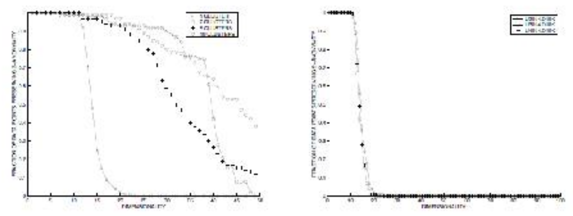


Fig 2: Data Points preserving examples

We have illustrated the behavior of a generalization approach in which each attribute is divided into only two ranges. The number of dimensions on the X-axis represents those which are partially specified using these two ranges, whereas all other dimensions are fully suppressed. On the Y - axis, we have illustrated the percentage of data points which maintain 2-anonymity using this generalization. We note that all other data points (which violate the 2-anonymity condition) would need to be suppressed. A high per- centage of suppression is never acceptable from a data mining point of view [14]. It is interesting to see that while a greater number of clusters (and corresponding skew) in the underlying data helps the anonymization, the percentage of data points which continue to pre- serve privacy falls of rapidly with increasing data dimensionality.

VI. REFERENCES

[1] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[2] P. Samarati, "Protecting Respondent's Privacy in Microdata Release," IEEE Trans. Knowledge

[3] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," Int'lJ. Uncertainty Fuzziness and Knowledge-Based Systems, vol. 10, no. 5,pp. 557-570, 2002.

[4] X. Xiao and Y. Tao, "Anatomy: Simple and Effective Privacy Preservation," Proc. Int'l Conf. Very Large Data Bases (VLDB),pp. 139-150, 2006.

[5] N. Koudas, D. Srivastava, T. Yu, and Q. Zhang, "Aggregate Query Answering on Anonymized Tables," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 116-125, 2007.

[6] D.J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J.Y.Halpern, "Worst-Case Background Knowledge for Privacy-Preserving Data Publishing," Proc.

[7]M.E. Nergiz, M. Atzori, and C. Clifton, "Hiding the Presence of Individuals from Shared Databases," Proc. ACM SIGMOD Int'lConf. Management of Data (SIGMOD), pp. 665-676, 2007.