2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications

Efficient Contextual Transaction Trust Computation in E-Commerce Environments

Haibin Zhang, Yan Wang
Department of Computing
Macquarie University
NSW 2109, Australia
{*haibin.zhang, yan.wang*}*@mq.edu.au*

Xiuzhen Zhang
School of Computer Science and IT
RMIT University
Melbourne, Victoria 3001, Australia
xiuzhen.zhang@rmit.edu.au

***Abstract*—In e-commerce environments, trust is a dominating factor in seller selection. Most existing trust evaluation studies compute a single value to reflect the "general" or "global" trust level of a seller provider without any contextual transaction information taken into account. As a result, a buyer may be easily deceived by a malicious seller in a forthcoming transaction. For example, with the notorious "value imbalance problem", a malicious seller can build up a high trust level by selling cheap products and then starts to deceive buyers in selling expensive products. To detect this problem and avoid massive monetary losses of buyers, trust evaluation should be associated with both past transactions and the new one, and take transaction context into account. In particular, the computed trust result should outline the seller's reputation profile indicating the trust level in a specific product or a product category, a price range, a time period or any necessary combination of them. However, this need requires complex computation and thus new data structures and efficient algorithms. In this paper, we design a new data structure to support the CTT computation in e-commerce environments. In addition, based on the new data structures, we further propose an approach for promptly responding to a buyer's CTT query. The conducted experiments illustrate that our proposed structure and approach can yield much shorter computation time than the existing approaches.**

***Keywords*-E-Commerce, Contextual Transaction Trust, Data Warehouse, OLAP**

I. Introduction

The trust of sellers is a crucial issue in open e-commerce environments [9, 7]. With a simple trust management system (such as eBay), buyers are vulnerable to some frauds from malicious sellers. For example, with the notorious *value imbalance problem* [6], a seller could try to build up a high trust level by honestly selling good products with low values (price); once obtaining a high trust level, he/she begins to deceive buyers when selling expensive products. Several real world cases with this problem have been revealed [9]. The value imbalance problem is one type of the transaction context imbalance problem. Any type of the problems in this category may cause fraud and lead to monetary losses of customers. At Alibaba.com [1], following a few cases of fraud, buyers are explicitly suggested to manually check if the products offered by a supplier are in the same categories as the products that the supplier usually sells.

Clearly, a single trust value can only reflect the "general" or "global" trust level of a seller, and it is static to any forthcoming transaction rather than being specific to it. However, different transactions have different *nature* and *contexts*; even the same seller needs to be regarded differently with respect to the trustworthiness in different forthcoming transactions, instead of using the same and thus static trust value [18, 15, 19].

Recognizing the importance of context in transaction trust evaluation, an immediate question to ask is what is transaction context. Briefly, the context of a transaction can be described by contextual transaction attributes. More detailed discussion of these contextual transaction attributes is presented in Section III. Here, we only emphasize that some contextual attributes have hierarchical structures and a transaction's context can be represented at multiple hierarchical levels (i.e., *different granularities*). For example, a buyer plans to buy a 'Cannon EOS T3i Digital Camera' at the price '$700' from a seller. Besides the trustworthiness of selling this specific product, he/she could be also concerned about the trustworthiness of this seller in selling 'Digital Cameras' with a price range of '$500-$900' (i.e., an query w.r.t a higher layer in the hierarchical product category with a specified price range) and time range (e.g., in the last one week). Therefore, given *contextual transaction trust* (CTT for short) queries with different granularities, the trust management system should promptly provide the computation results to potential buyers.

In our earlier work [17], we have introduced a trust vector which contains a set of trust values corresponding to CTT with different granularities, and it includes (a) the trust value of a seller in selling a specific product in a time period, (b) the trust value of a seller at a specific product category, in a price range and a time period and (c) the trust value of a seller in a specific price range. The above vector is bound to each forthcoming transaction and the past transactions. Even for the same seller, it may vary with different forthcoming transactions, as different forthcoming transactions possess different products and prices. Compared with existing trust management systems, which provide a single and static trust value, this vector based trust evaluation has the following advantages:

- This trust vector can represent the reputation profile of a seller with trust values in different product categories, price ranges and time periods. More importantly, it may dynamically vary with different forthcoming transactions. As the result, the potential buyer could know trust level of the seller more precisely.
- Potentially malicious transactions with the *transaction context imbalance* problem [17] (the *value imbalance*

problem is one type of the *transaction context imbalance* problem) can be identified.

As stated above, a buyer's CTT queries on price ranges can vary from product to product, and each time range varies from a previous point (e.g., one month ago) to the present time. However, in real applications, a seller usually has a large number of transactions within a long period (e.g. one year). Therefore, how to compute CTT and promptly respond to CTT queries of different granularities with a set of trust values, is a big challenge. Towards solving this challenging problem, we propose our solutions and our contributions in this paper, which can be briefly summarized as follows:

(1) We first model the CTT computation as the RA (range aggregate) problem in spatial data warehouse with some modification. The RA problem is to compute an aggregate function over all spatial objects that fall into a query region.

(2) In literature, some existing methods aim to solve the RA query problem in two-dimensional spatial databases, such as aR-tree [8] and aP-tree [12]. According to our analysis, we firstly point out the limitations of these methods in solving our targeted problem, and then propose a new approach to fast indexing rating aggregates for CTT computation. In particular, we propose a hybrid structure of aP$^+$-tree based on aP-tree and aB$^+$-tree based on B$^+$-tree to further reduce the computation time of CTT queries.

(3) We have conducted experiments on a real dataset from eBay with the transactions of 90 days of the most popular seller for selling 'Cannon EOS T3i Digital Camera' and a synthetic dataset for the transactions of 12 months with 10 times transactions as much as the above popular seller on each day, respectively. The experiments compare our approach with the one based on traditional aR-tree or aP-tree for answering CTT queries. The experimental results illustrate that our proposed approach is much faster than them in responding to CTT queries.

(4) To the best of our knowledge, this is the first solution in the literature to the computation of CTT in e-commerce environments.

The rest of the paper is organized as follows: Section II provides a brief overview of both existing trust evaluation approaches and the OLAP technology. We give a detailed explanation on CTT query with different granularities in Section III. In Section IV, we introduce the trust metric for our proposed trust vector. Section V provides our method for supporting CTT computation. Section VI evaluates our approach experimentally, and Section VII concludes our paper.

II. Related Work

A. Trust Evaluation without Contextual Information

In the literature, trust evaluation models are studied in some application fields. For example, in Peer-to-Peer (P2P) information sharing networks, a "global" trust value of a given peer is calculated via collecting binary trust ratings [5]. However, these existing trust evaluation models focus on computing a single *"final trust level"* (e.g. a value in the range of $[0, 1]$ [10, 18, 16, 7]) to reflect the "general" or "global" trust status of every seller, and they do not take any contextual information into account. With such a result, a buyers can hardly know under what kind of circumstances the seller has obtained a high trust value. So it fails to predict the likelihood of the seller for a successful forthcoming transaction.

B. Contextual Trust Evaluation

There are also some existing studies considering the relationship between trust evaluation and context information. Griffiths [3] proposed a Multi-Dimensional Trust (MDT) model, which studied contextual trust from multiple facets. Given the same seller, the trust results computed for different buyers may be different. Similarly, in REGRET [10] and RATEweb systems [7], the multi-attribute structure was adopted when calculating a seller's reputation. But these models still focus on how to compute a single general or global trust value, and overlook the fact that transaction context may vary in historical transactions. Therefore, they also fail to predict the likelihood of the seller in a successful forthcoming transaction.

In the literature, context similarity calculation is regarded as an important means to deal with contextual trust evaluation problem. Uddin *et al*. proposed a CAT (Context-Aware Trust) model to compare the similarity of contexts by using key values that could describe certain context to some extent [14]. For example, in task A: "My brother drives me to the airport", the keywords are $\{$my brother, drive, car$\}$; in task B: "My brother flies the plane", the keywords are $\{$my brother, fly, plane$\}$. While task A is trustworthy, task B may be untrustworthy, because two out of three keywords are different. Toivonen *et al*. use a more complex ontology structure to calculate similarity [13]. This idea works for the situation when there are no or insufficient ratings available to infer the trust level of a seller in certain transaction context. Following this idea, in [19], we proposed detailed formulae to calculate transaction context similarity so as to infer the trust level of forthcoming transactions in e-commerce environments.

However, these contextual trust evaluation models focus on combining all the contextual attributes (factors) into one model and using a single value for trust evaluation which can not distinguish the contribution of each contextual attribute specifically. In our paper, first of all, we discuss and model the contextual transaction attributes in e-commerce environments. In addition, compared with single value based approach, our propose trust vector could more precisely reflect the trust status of a seller in a forthcoming transaction.

C. OLAP (On-Line Analytical Processing)

In OLAP applications, it is common to include hierarchies for several dimensions. For instance, a sales data warehouse for a company contains the dimension of *Location* which is a hierarchical structure. The data can be grouped by *city*, but the users may have queries that involve grouping by *country*. Some researchers precalculated some of these results (such as pre-aggregate the results of the sales data in a specific location) so as to accelerate the response to such queries [4]. This idea is designed to support static hierarchies only (i.e. predefined hierarchies). But, in some cases, the hierarchies are often not known in advance. In section III, we will discuss that transaction dimensions have dynamic hierarchies that should be taken into account in CTT computation.

III. Transaction context and CTT queries with different granularities

A. Transaction Context

In our work, transaction context is modeled with some contextual transaction attributes, all of which have influence on the trustworthiness of a forthcoming transaction.

• **Transaction item**: *Transaction item* refers to the product traded in a transaction, the properties (e.g. product quality and product category) of which determine the nature of the transaction. A seller, who sells high quality 'handbags', may provide poor quality 'Notebook Computer'.

A transaction item belongs to a predefined (static) hierarchy in product category. The classification of product category is multilayer, for instance, at eBay, they are at least two layers in the hierarchy of category. If a specific product is 'Cannon EOS T3i Digital Camera', then its ancestors in category hierarchy are 'Digital Camera' and 'Cameras & Photo' in order. Hence, for CTT queries in transaction item, in addition to the trustworthiness of this specific product, a buyer may be also concerned about the trustworthiness of the seller in selling various products in a subcategory or category. In some existing products and services categorization standards, such as United Nations Standard Products and Services Code (UNSPSC)[1] and eCl@ss[2], the number of layers can be up to four.

• **Transaction amount**: *Transaction amount* refers to the sum of prices of all products in a transaction. The transactions of about US\$10 are obviously different from those of about US\$10K in nature. The larger the transaction amount, the more likely for a fraud to happen since the benefits of cheating are greater. For the sake of simplicity, in this paper, *each item in a transaction is considered separately in trust computation*. A transaction with multiple transaction items are taken as several transactions with one item each. Consequently, in the rest context of this paper, the transaction amount equals to the price of a product.

Furthermore, there is no predefined hierarchy in transaction amount, because it varies from product to product; and a buyer's CTT query on price may also vary. For instance, the transaction amount between a buyer and a seller is around \$500, and a buyer may be concerned about the trustworthiness of the seller on selling products at the price range of \$400-\$600. If the transaction amount changes to \$1500, the corresponding price range query may change to \$1000-\$2000.

• **Transaction time**: *Transaction time* is the time when a transaction happens. Trust evaluation is time-sensitive, because the transaction quality may change with time [11].

Transaction time has a specific feature in trust computation. Any query on temporal dimension should start from a previous point (e.g. one month ago) and end at the present time.

B. CTT Queries

As stated before, the context of a transaction can be described at different granularities. Suppose that a buyer plans to buy 'Cannon EOS T3i Digital Camera', his/her CTT query with transaction context at higher levels can be <product-subcategory: Digital Camera, price-range: 600-800, time-range: from a month ago to present>, and a query at a further higher levels can be <product-subcategory: Digital Camera, price-range: 600-800> or <Price-range: 600-800>.

[1]http://www.unspsc.org/
[2]http://www.eclass.de/

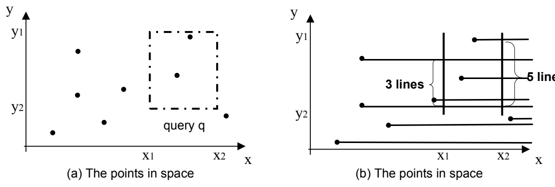


Figure 1. Transformation process of a RA query

In addition, CTT should also be evaluated from other aspects. For example, at eBay, apart from a general rating (denote as r_g) to reflect a seller's performance during the whole transaction, buyers' ratings also evaluate (i) shipping time (i.e. whether the seller delivers goods on time, denote as r_{st}), (ii) communication (i.e. whether the seller has prompt and friendly communication with buyers, denote as r_c), (iii) shipping charges (i.e. whether the seller charges a reasonable price for shipment, denote as r_{sc}) and (iv) the quality of delivered goods (denote as r_q). The design of our new data structures will take into account all the above facets.

IV. CTT metric in our proposed trust vector

In this section, we will present how to calculate each of the elements in our proposed trust vector [17], which includes CTT with different granularities. These trust vector elements include, (a) the trust value of a seller in selling a specific product in a time period, (b) the trust value of a seller at a specific product category, in a price range and a time period and (c) the trust value of a seller in a specific price range.

A. CTT metrics

In the literature, the researchers averaged the rating values as metric to evaluate a sellers' trust value [5, 18, 16]. Following this idea, our approach will calculate the trust level as the average of rating values in a specific transaction context to evaluate a sellers' CTT. In order to satisfy this requirement, two aspects of the aggregate data are stored respectively, i.e., *count* and *aggregate ratings* (denoted as agg_r). The field *count* records the number of transactions and agg_r records the sum of ratings that are given by buyers over these transactions. In section III, we have pointed out that CTT should also be evaluated from multiple aspects. Correspondingly, the agg_r field could be an array including several dimensions.

B. The General Idea of Our Approach

1) The Existing Work for RA Query: The range aggregation (RA) query on spatial data warehouse, as illustrated in Fig 1(a), is to compute the total number of points that fall into the query region q (i.e., the region surrounded by $[x_1, x_2]$ and $[y_1, y_2]$).

Previous research work focused on RA query in a two-dimensional space. Tao *et al.* [12] proposed an aP-tree structure, which converts each spatial point to an interval (see Fig 1(b)). If the region q is transformed to two vertical lines $x_1 : [y_1, y_2]$ and $x_2 : [y_1, y_2]$, and then an RA query will turn to retrieve the number of intervals that intersects two vertical lines. The total number of points in that region q equals to

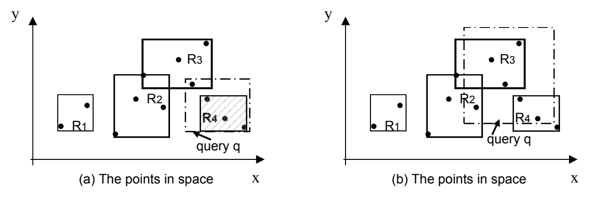


Figure 2. An aR-tree example

the difference of them (i.e. $5 - 3 = 2$). Tao *et al.* defined the number of lines intersecting in each border as a vertical range aggregate (VRA). In such a way, a RA query can be reduced to two vertical range aggregate (VRA) queries.

In our approach, the above transformation is adopted to deal with CTT queries in the two dimensions of transaction amount and transaction time, since it avoids the shortcoming of aR-tree [8]. The aR-tree maintains x-y coordinates for each minimum bounding rectangle (MBR) (see Fig 2, R_1, R_2, R_3, R_4 are all MBRs). At the same time, it records the total number (aggregate function) of objects that fall into that MBR. As shown in Fig 2(a), to answer the number of spatial objects falls into the query region q, the MBR R_4 will not be accessed, which is inside the query window, and its pre-aggregated result (i.e., 3) is directly used. In this case, only R_3 needs to be visited, which is overlapped by query region q. But a serious problem in the aR-tree is that the query cost depends on the size of the query region: the larger the query region, the more MBR overlaps with it (see Fig 2(b)).

In order to index each VRA value, the aP-tree introduces an additional field *agg* in each entry of the original multiversion B-tree (MVBT) [2]; thus the entry format of the aP-tree is $< y, [x_{start}, x_{end}), agg, pointer >$. Since MVBT is suitable for indexing temporal databases, Tao *et al.* added a restrictive condition, i.e., the x-coordinates of the original data points should be in an ascending order. As a result, a spatial point with x-y coordinates are transformed to the point in time-key plane (i.e., x-axis is time space and y-axis is key space), and each interval in x-axis ($[x_{start}, x_{end})$) represents a time interval $[t_{start}, t_{end})$ which is also called $lifespan$. For a leaf entry, the field $pointer$ points to an actual record in data warehouse. For an intermediate entry, the field $pointer$ points to its child and the field agg equals to the number of leaf entries in its subtree alive (the entry with $x_{end} = “*”$ means it is alive) in $lifespan$.

2) RA Query in CTT computation: Considering our targeted problem as discussed in section III, we have illustrated that transaction context includes a dimension with the predefined hierarchy and two dimensions with undefined hierarchies. Obviously, the CTT query in transaction amount and transaction time dimensions can be converted to the RA problem, including specific feedback information (i.e. the average of rating values) to reflect a seller's trust level.

Meanwhile, taking into account the dimension of product category, we design a new data structure aP$^+$-tree based on aP-tree for efficient CTT computation. The new structure is to create the new root entries to store the static product category hierarchy. For each root entry, an additional *pointer* points to its child, which maintains the aggregated values obtained from the two dimensions of transaction amount and transaction time. Moreover, according to the feature of transaction time as described in section III, where the end time is fixed to "present" in CTT computation, we design a new data structure *aB$^+$-tree* based on B$^+$-tree for computing right border VRA query. Finally, we propose a hybrid structure of aP$^+$-tree and aB$^+$-tree to further reduce the computation time of CTT queries.

C. CTT metrics for each element in trust vector

1) The trust metrics for elements (a) and (b): In this subsection, we give a general metric for both elements (a) and (b) in our proposed trust vector, since the computation of element (a) can be regarded as a special case for computing element (b). More details on structure design will be illustrated in the following section.

Recall that a RA query is reduced to two vertical range aggregate (VRA) queries problem. Thus, a CTT query regarding a past time t to the present time t_P can be calculated as:

$$T_{CTT}^{[t,t_p]} = \frac{agg\_r_2 - agg\_r_1}{count_2 - count_1} \tag{1}$$

For the element (a), $count_1$ and agg_r_1 represent left border VRA CTT query in a specific product; $count_2$ and agg_r_2 present right border VRA CTT query in this product. For the element (b), $count_1$ and agg_r_1 are from the corresponding left border VRA CTT query at a specific product category and transaction amount range; $count_2$ and agg_r_2 are from the corresponding right border VRA CTT query at the same product category and transaction amount range.

2) The trust metric for element (c): CTT at a specific price range is important for identifying whether a seller cheat via value imbalance [15]. When a seller has a low averaged trust value at a specific price range, he/she may be untrustworthy. In such a case, our method only needs to compute for the right border VRA CTT query. If all the products sold by a seller are from k subcategory, $count_2^{(i)}$ denotes the number of transactions at a specific product subcategory ($i = 1, 2, 3..k$) and a specific transaction amount range, and $agg\_r_2{}^{(i)}$ denotes the corresponding *aggregate ratings*. Thus, a CTT query at a specific price range $[ta_1, ta_2]$ can be calculated as:

$$T_{CTT}^{[ta_1,ta_2]} = \frac{\sum_{i=1}^{k} agg\_r_2{}^{(i)}}{\sum_{i=1}^{k} count_2^{(i)}} \tag{2}$$

As stated before, product category hierarchy could have multilayer (e.g. subcategory, subsubcategory, etc).

V. The Approach to efficient CTT computation

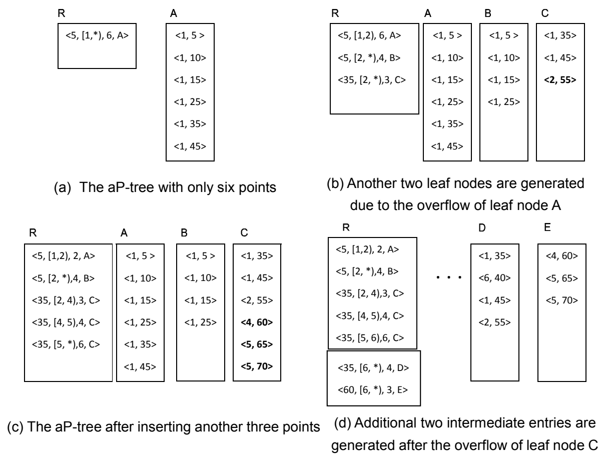
A. Our Proposed aP$^+$-tree

In our proposed the aP$^+$-tree for efficient CTT computation with different granularities, we designed three different formats (structures) for the entries: root entry, intermediate entry and leaf entry (see Table I). As discussed in subsection IV-B2, we introduce a new root entry in addition to original aP-tree, which contains hierarchy of transaction item categories. For each root entry, the field *pointer* points to an intermediate or a leaf node (each node contains 1 to b entries where b is the node capacity).

1) Tree construction: In the following, a simple example is used to illustrate the construction of an aP-tree. As shown in Fig 3(a), six points with x-y coordinates $(1, 5), (1, 10), (1, 15), (1, 25), (1, 35), (1, 45)$ are inserted to a leaf node A, assuming the capacity of the node is six ($b = 6$, b

Table I
THE ENTRY FORMAT

Entry	Format
Root entry	<subcategory,category,$[ta_{min}, ta_{max}]$, $count$,agg_r,$pointer$>
Intermediate entry	$<[ta_{min}, ta_{max}], [t_{start}, t_{end}), count, agg\_r, pointer>$
Leaf entry	<transaction amount, transaction time, $count$, agg_r, $pointer$>



(a) The aP-tree with only six points

(b) Another two leaf nodes are generated due to the overflow of leaf node A

(c) The aP-tree after inserting another three points

(d) Additional two intermediate entries are generated after the overflow of leaf node C

Figure 3. The construction of aP-tree

for node capacity). The intermediate entry $< 5, [1, *), 6, A >$ in root R (see Fig 3(a)) implies that there are six entries in leaf node A, which are alive from $x_{start} = 1$ and their keys are at least 5. When a new entry (2, 55) with different x-coordinate is inserted to A, the entry $< 5, [1, *), 6, A >$ dies, having its x_{end} modified to 2 (see root R in Fig 3(b)). Since the leaf node A is overflow after inserting the new entry $(2, 55)$, it splits into two leaf nodes B and C as shown in Fig 3(b). The additional two intermediate entries $< 5, [2, *), 4, B >$ and $< 35, [2, *), 3, C >$ are generated with the same $x_{start} = 2$, which point to leaf nodes B and C, respectively. Meanwhile, the whole process is also accompanied by keys (i.e. y-coordinates) split.

The way to locate a position inserted in an aP-tree is to *iteratively find the live entry* $[x_{start}, *)$ *whose y-coordinate is the largest among all the entries*. For example, Fig 3(c) illustrates the aP-tree after inserting another three entries $(4, 60), (5, 65), (5, 70)$. When the point $(4, 60)$ with different x-coordinate is inserted, the intermediate entry $< 35, [2, *), 3, C >$ dies, and a new the intermediate entry is duplicated from $< 35, [2, *), 3, C >$ with its x_{start} set to 4, and its agg incremented by one (see root R in Fig 3(c)). Fig 3(d) illustrates that two leaf nodes D and E are generated after inserting the entry $(6, 40)$. At this time, two additional intermediate entries make the root node R overfull and the two new roots R and R_1 (R_1 contains all the alive entries, i.e., $< 5, [2, *), 4, B >$, $< 35, [6, *), 4, D >$ and $< 60, [6, *), 3, E >$) are generated resulting from the spilt of the original root R. Additionally, different from MVBT, aP-tree has only a single parameter svo, which denotes the threshold of *strong version overflow*. It occurs when the number of entries in a new node exceeds $b * svo$, and more details on construction of an aP-tree can be found in [12].

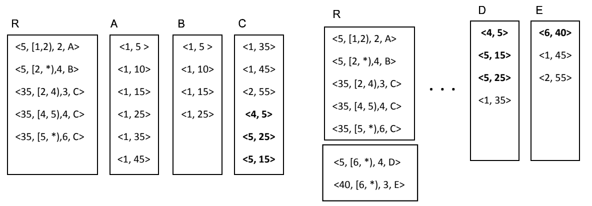
The process of aP$^+$-tree construction is similar to aP-tree. However, different from using only a key (the field y) as the index in aP-tree, we use the field $[ta_{min}, ta_{max}]$ in aP$^+$-tree to record the minimum and maximum the transaction amounts in its subtree. Although each intermediate entry increases a data field, it could reduce the number of nodes to be accessed, and we will give a further explanation in the following subsection. Furthermore, compared with aP-tree, the leaf entry format in aP$^+$-tree still has the fields $count$ and agg_r. That is because a seller may have the same transactions in a transaction time unit (the transactions have the same x-coordinate (see Fig 1)), i.e., it is normal for a seller selling a number of the same product within a day (the minimum unit of transaction time is set to a day). But for the different products with the same transaction amount, we use two leaf entries to differentiate them so as to position each specific product.

2) Less nodes to be accessed in aP$^+$-tree: To answer a VRA query $5 : [10, 20]$ in Fig 3(c), first of all, the entries in root should be considered, whose lifespans include 5; thus, entries $< 5, [1, 2), 2, A >$, $< 35, [2, 4), 3, C >$ and $< 35, [4, 5), 4, C >$ are eliminated immediately, because they are already dead before 5. Secondly, among the remaining alive entries $< 5, [2, *), 4, B >$ and $< 35, [5, *), 6, C >$, $< 35, [5, *), 6, C >$ is eliminated because the keys in its subtree are no less than 35 (i.e. in $[35, +\infty)$), which does not intersect with the query range $[10, 20]$ in y-axis. In such a case, only the leaf node B in aP-tree need to be visited pointed by $< 5, [2, *), 4, B >$, and the result 2 is returned.

Analysis: According to the description above, an important observation is that the efficiency of aP-tree depends on whether the key space (y-coordinates of all records) is inserted in order. In our example, the key space is also in ascending order. In such a case, it is easy to infer the range of key space in its subtree. However, towards our targeted problem, it is difficult to satisfy that keys are inserted in order, and it is quite common that the transaction amounts at the start time and end time are the same, i.e., a seller sells the products with the same price but at different transaction time. The range of key space in its subtree is difficult to infer as above, so there are still a large number of nodes to be accessed in original aP-tree.

An Example: In subsection V-A1, we have pointed out that the way to locate a position inserted in aP-tree is to iteratively find the live entry whose y-coordinate is the largest among all the entries. For instance, in Fig 4(a), if the inserted entries are $(4, 5), (5, 25), (5, 15), (6, 40)$ instead of $(4, 60), (5, 65), (5, 70), (6, 40)$ as in Fig 3(c), the additional two intermediate entries will be $< 5, [6, *), 4, D >$ and $< 40, [6, *), 3, E >$ (see Fig 4(b)), rather than $< 35, [6, *), 4, D >$ and $< 60, [4, *), 3, E >$ as shown in Fig 3(d). Assume that a VRA query is $6 : [5, 30]$, and then two of three alive intermediate entries (i.e. $< 5, [2, *), 4, B >$, $< 5, [6, *), 4, D >$) should be visited. If the VRA query changes to $[20, 50]$, and then all the leaf nodes pointed by three alive intermediate entries should be visited.

By contrast, in our proposed aP$^+$-tree, the generated two intermediate entries are $< [5, 35], [6, *), 4, D >$ and $< [40, 55], [6, *), 3, E >$, instead of $< 5, [6, *), 4, D >$ and $< 40, [6, *), 3, E >$ (see Fig 4(b)). As a result, for the same VRA query $6 : [5, 30]$ as the above example, the leaf node



(a) Different three points inserted to aP-tree (b) Two different intermediate entries are generated after inserting four points

Figure 4. Another example for construction of aP-tree

pointed by $< [5, 25], [2, *), 4, B >$ is not accessed due to being covered by $[5, 30]$ in aP$^+$-tree, but it will be visited in aP-tree. Clearly, aP$^+$-tree can further reduce the number of accessed nodes.

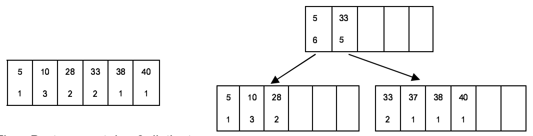
B. Our Proposed aB$^+$-tree in Transaction Amount Space

Now, let us consider the feature of transaction time in trust computation as described in section III, where each range query should start from a previous point and end at the "present". Hence, the right border VRA query is fixed to "present" (i.e. the end time) in CTT computation. In subsection V-A2, we could see that if the right border VRA query is taken as the end time, all the alive entries (i.e. the entries with $x_{end} = ``*''$) are checked whether their key-ranges intersect with $[y_1, y_2]$. Hence, there are still a large number of nodes to be accessed due to disordered key space. At this time, another observation is that, at the right border, CTT query has been transformed to one-dimensional transaction amount range CTT queries. Following this idea, in our approach, we construct a B$^+$-tree in transaction amount space. Meanwhile, the B$^+$-tree contains additional fields to store aggregate information (i.e., $count$ and agg_r), and we term this new tree as *aB$^+$-tree*. In this way, the right border VRA query will be converted to search in fully ordered transaction amount space, while aP$^+$-tree for the left border VRA CTT query.

In order to build up the tree, another entry format $<$ transaction amount, $count$, agg_r, $pointer$ $>$ is introduced. Both intermediate and leaf entry in aB$^+$-tree have this format. The fields $count$ and agg_r record the number of transactions within a transaction amount range and the corresponding aggregate rating of these transactions. Here, we emphasize that both aP$^+$-tree and aB$^+$-tree share the same root entry (see Table I). In addition, the same as aP$^+$-tree, for the different products with the same transaction amount, we use two leaf entries to differentiate such a case even if they are in the same subcategory.

Due to space constraints, we briefly illustrate construction process of aB$^+$-tree in transaction amount space (with node capacity 6) as in Fig. 5. Assume that 10 transactions with 6 distinct values of transaction amounts \$28, \$10, \$33, \$33, \$38, \$10, \$40, \$28, \$10, \$5 are inserted to a node. The number 1 in the $count$ field of leaf entry <\$5, 1,agg_r,pointer> (the figure only presents transaction amount and $count$ fields in an entry for sake of simplicity) implies that there are only one transaction in the transaction amount range (0,\$5]. When a new transaction with a different transaction amount \$37 is inserted, the leaf node is overflow as shown in Fig. 5(b).

For a transaction amount range query, the depth-first traversal (DFT) is performed to search all the leaf nodes for answering a transaction amount range query. For instance, a range query is [\$10,\$30], which is covered by the left subtree (i.e. (0, \$33]) in Fig. 5(b), and then all the entries in left subtree are visited by accumulating the aggregate fields $count$ and agg_r along the way. Specifically, in this example, the result 5 is returned for field $count$ (i.e. 3+2=5), which indicates there are 5 transactions within the transaction amount [\$10,\$30] in the query.



(a) The aB+-tree contains 6 distinct values of transaction amount (b) The split of the leaf node

Figure 5. The construction process of aB$^+$-tree in transaction amount space

C. Summary

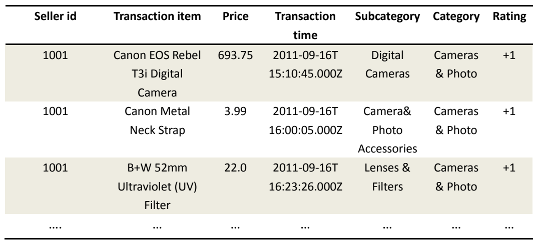
(1) Though we introduce the aB$^+$-tree, in addition to the aP$^+$-tree, to record the aggregate information in transaction amount, there is not much increase in space. Let us suppose that a seller has N transactions in a time period, and there are M distinct values of transaction amount ($M \leq N$). It requires the additional space with $O(M)$ entries in the aB$^+$-tree. In a typical case, for example, on average, if every 5 transactions have the same price (for the same or different products), then $M = N/5$.

(2) For an update operation (insertion or remove) in both an aP-tree and an aR-tree, the time complexity is $O(logN)$ (the height of the tree), while it is $O(logM)$ in aB$^+$-tree. Overall, in our approach, the time complexity is $O(logN) + O(logM) = O(logN)(M \leq N)$. Similarly, for n update operations, our approach has the same time complexity as aP-tree or aR-tree ($O(nlogN)$). In real operations, when a new transaction has completed, the new leaf entry is inserted into the tree with new aggregates computed. Meanwhile, after all new transactions in the current day have completed, all transactions completed on the first day (e.g. 12 months ago) available in the tree should be removed. This aims to maintain the information of all transactions of a seller in a time period (e.g. 12 months).

(3) Considering our proposed trust vector that includes CTT with different granularities in section IV, our approach uses a hybrid structure of aP$^+$-tree and aB$^+$-tree for computing elements (a) and (b) in the trust vector and uses aB$^+$-tree for computing element (c) in the trust vector. Our approach leads to much shorter computation time than the methods based on aP-tree or aR-tree in the responses to CTT queries. This is also demonstrated by experiments, with details to be introduced in the following section.

VI. Experiments

In this section, we evaluate our proposed method in a scenario where a buyer plans to buy a "Cannon EOS T3i Digital Camera". The experimental setup is introduced in Section VI-A. The results and the analysis are introduced in Section VI-B.



Seller id	Transaction item	Price	Transaction time	Subcategory	Category	Rating
1001	Canon EOS Rebel T3i Digital Camera	693.75	2011-09-16T 15:10:45.000Z	Digital Cameras	Cameras & Photo	+1
1001	Canon Metal Neck Strap	3.99	2011-09-16T 16:00:05.000Z	Camera& Photo Accessories	Cameras & Photo	+1
1001	B+W 52mm Ultraviolet (UV) Filter	22.0	2011-09-16T 16:23:26.000Z	Lenses & Filters	Cameras & Photo	+1
….	…	…	…	…	…	…

Figure 6. The transaction information of a real seller at eBay

A. Experiment Setup

The experiments are implemented using C++ running on a Lenovo Y560 laptop with an Intel Core i5 2.20GHz CPU and 2GB RAM. eBay has released APIs[3], with which we could obtain detailed feedback and transaction information for a seller for up to 90 days. We have developed a program based on eBay APIs and obtained the *real dataset* including the information of transactions and ratings from all the 56 sellers who sell 'Cannon EOS T3i Digital Camera' on 6th October 2011 at eBay. Among all 56 sellers, we selected the most popular seller, who had totally 2600 transactions in 17 different subcategories and 3 different categories within 90 days (approx. 800-900 transactions per month see Fig. 6). In particular, this seller had a large amount of transactions in the subcategory 'Digital Camera' and this is more suitable for our experiments for answering CTT queries with different granularities. Our experiments are also carried out on a *synthetic dataset*, which contains the transactions of a seller in 12 months. Each transaction in our synthetic dataset is generated by randomly selecting a transaction from 2600 transactions in the real dataset. Meanwhile, we generate 10 times transactions as much as this popular seller on each day and the seller has 8000-9000 transactions in each of 12 months in our synthetic dataset, and the total number of transactions is around 100,000. The purpose of generating this synthetic dataset is to test the performance of our approach under the circumstance with a large volume of transactions in both each day and a long time period (e.g., 12 months).

With a CTT query on this seller, we only measure the computation time for the trust vector elements (b) and (c) proposed in section IV, as (a) and (b) have the same calculating procedure, i.e., the element (a) can be obtained by further searching each actual record (pointed by leaf entry) in data warehouse. We set the minimum unit of transaction time to a day (i.e., the transactions that occur in the same day have the same x-coordinate (see Fig 1)). In addition, eBay does not provide detailed ratings for r_g, r_{st}, r_c, r_{sc} and r_q; therefore, our experiments randomly generate integers in $[1, 5]$ for them. Here, we need to emphasize that their random values do not have any effect on node access and computation time.

B. The Comparison of Computation Time

In this subsection we compare the computation time of our approach with the aR-tree based approach and the aP-tree based approach for answering a CTT query. Each result of the computation time is the average based on 3 independent executions. All the trees have the same page size of 512 bytes.

[3]developer.ebay.com/support/docs

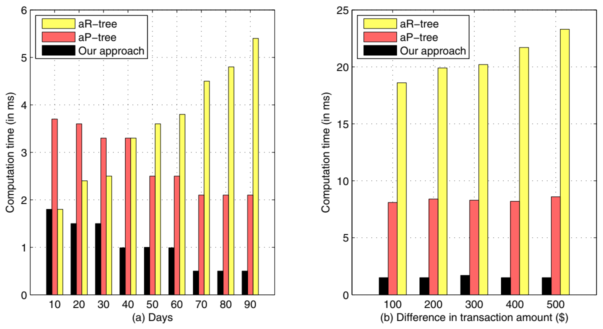


Figure 7. The computation time of CTT queries on a real dataset

The fields of transaction amount and transaction time in each entry have 4 bytes each, and the fields of $count$ and agg_r have 2 bytes and 10 bytes (due to five different aggregate ratings in agg_r), respectively. Therefore, for aP-tree, the capacity of the leaf node and the intermediate node is 19 and 16, respectively. For aR-tree, the corresponding capacity numbers are 19 and 14. Since our strategy introduces an extra field to record the maximum transaction amount in its subtree, the intermediate node's capacity of aP$^+$-tree is also 14. The parameter svo for both aP-tree and aP$^+$-tree is set to 5/6. For aB$^+$-tree, the capacity of both the leaf node and the intermediate node is 21.

Results on real dataset: The first set of experiments is conducted on the real dataset. We tested the computation time for the element (b) in trust vector (i.e., the trust value of a seller at a specific product subcategory ('Digital Camera'), in a price range and a time period). We choose a possible transaction amount range at [\$600,\$700]. The time range starts from 10 days ago to 90 days ago with a step of 10 days. Fig. 7(a) depicts the computation time of three different approaches. For the aR-tree based approach, when the time range in a query becomes larger, computation time increases linearly. By contrast, in the aP-tree based approach, a larger time range leads to a shorter computation time. As stated before, the right border VRA CTT query is fixed to "present" in CTT computation; thus, all the alive nodes need to be checked. When the query on time range increases it means that the number of nodes to be accessed in aP-tree for left the border VRA CTT query will decrease. That is because the alive nodes with its begin time bigger than the left-end point of time range will not be visited. As a result, the computation time decreases monotonously. At the same time, we observe the same feature of computation time in our approach. More impressively, our approach has a faster response time in CTT computation than the aP-tree based approach. That is because, on one hand we propose aP$^+$-tree to reduce the number of nodes to be accessed for left border VRA CTT query; on the other hand, the right border VRA CTT query only needs to search in fully ordered transaction amount space (i.e., aB$^+$-tree). On average, for the element (b) in trust vector, the computation time of our approach is 28% of that of the aR-tree based approach and 36% of that of the aP-tree based approach.

Fig. 7(b) depicts the computation time of CTT at a specific price range (i.e., the element (c) in trust vector). For the product 'Cannon EOS T3i Digital Camera' selling at price

\$694 (see Fig. 6), we test the computation time at five possible transaction amount ranges, i.e., [\$600,\$700] (difference = \$100), [\$500,\$700] (difference = \$200), [\$500,\$800] (difference = \$300), [\$400,\$800] (difference = \$400), [\$400,\$900] (difference = \$500). Note that here the difference in price, instead of the price itself, may effect the computation time. The aR-tree based approach has the worst performance due to a large query region q. In this case, the query region in time dimension is from 90 days ago to present. By contrast, we can see that the computation time for aP-tree based approach is stable. That is because, in such a case, the aP-tree based approach only needs to compute for the right border VRA CTT query. Similarly, our approach also computes for the right border VRA CTT query. But differently, our approach searches in the aB$^+$-tree with a fully ordered transaction amount space. Therefore, our strategy is much faster in computation time. On average, for the element (c) in trust vector, the computation time of our approach is 7% of that of the aR-tree based approach and 18% of that of the aP-tree based approach.

Results on synthetic dataset: The same experiments are also conducted on the synthetic dataset, which contains approximately 100,000 transactions of the seller distributed in 12 months. From the results depicted in Fig. 8, we can draw the same conclusion as above. For element (b) in trust vector, on average, the computation time of our approach is 8% of that of the aR-tree based approach and 28% of that of the aP-tree based approach. For element (c) in trust vector, on average, the computation time of our approach is 4% of that of the aR-tree based approach and 13% of that of the aP-tree based approach.

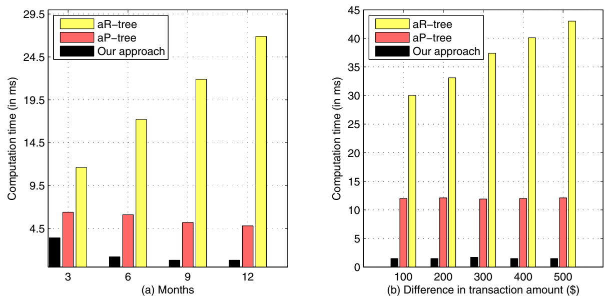


Figure 8. The computation time of CTT queries on a synthetic dataset

VII. Conclusion

In the literature, a lot of researchers have pointed out that it is necessary to introduce context factors in trust evaluation in e-commerce environments [18, 15]. To this end, in this paper, we proposed an approach to contextual transaction trust computation in e-commerce environments based on data warehouse technology. To the best of our knowledge, this is the first solution in the literature to the computation of CTT. In our solution, we first model CTT computation as a range aggregate (RA) problem. Then, we designed a new data structure to support CTT computation based on some existing data structures for the RA problem. In addition, after identifying the limitations of existing methods for solving our problem, we proposed an approach to further reduce the computation time of CTT queries. From the result of our experiments, we can see that our proposed approach is much faster than the traditional approaches for RA query in responding to CTT queries. In addition, there is no significant increase in data space for our strategy.

In our future work, we plan to design a new data structure to further reduce the data space with the good performance in computation time.

References

[1] Protect yourself from fraudsters pretending to be gold suppliers, http://resources.alibaba.com/article/232530/protect_yourself _from_fraudsters_pretending_to_be_gold_suppliers.htm.

[2] B. Becker, S. Gschwind, T. Ohler, B. Seeger, and P. Widmayer. An asymptotically optimal multiversion b-tree. *The International Journal on Very Large Data Bases*, 5(4):264–275, 1996.

[3] N. Griffiths. Task delegation using experience-based multi-dimensional trust. In *IEEE International Conference on Autonomous Agents and Multiagent Systems*, pages 489–496, Utrecht, Netherlands, 2005. IEEE.

[4] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996. ACM.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *International World Wide Web Conference*, Budapest, Hungary, 2003.

[6] R. Kerr and R. Cohen. Modelling trust using transactional, numerical units. In *ACM International Conference on Privacy, Security and Trust*, Oshawa, Canada, 2006. ACM.

[7] Z. Malik and A. Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. *The International Journal on Very Large Data Bases*, 18(4):885–911, 2009.

[8] D. Papadias, P. Kalnis, J. Zhang, and Y. f. Tao. Efficient olap operations in spatial data warehouses. In *International Symposium on Spatial and Temporal Databases*, pages 443–459, California, USA, 2001.

[9] B. Rietjens. Trust and reputation on ebay: Towards a legal framework for feedback intermediaries. *Information and Communications Technology Law.*, 15(1):55–78, 2006.

[10] J. Sabater and C. Sierra. Regret: Reputation in gregarious societies. In *ACM AGENTS*, pages 194–195, Montreal, Canada, 2001. ACM.

[11] S. Spitz and Y. Tuchelmann. A trust model considering the aspects of time. In *International Conference on Computer and Electrical Engineering*, pages 550–554, Dubai, UAE, 2009. IEEE.

[12] Y. Tao, J. Zhang, D. Papadias, and N. Mamoulis. Range aggregate processing in spatial databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1555–1570, 2004.

[13] S. Toivonen, G. Lenzini, and I. Uusitalo. Context-aware trust evaluation functions for dynamic reconfigurable systems. In *Models of Trust for the Web Workshop*, Edinburgh, Scotland, 2006.

[14] M. Uddin, M. Zulkernine, and S. Ahamed. Cat: A context-aware trust model for open and dynamic systems. In *ACM Symposium on Applied Computing*, pages 2024–2029, Fortaleza, Brazil, 2008.

[15] Y. Wang and K.-J. Lin. Reputation-oriented trustworthy computing in e-commerce environments. *IEEE Internet Computing*, 12(4):55–59, 2008.

[16] Y. Wang and V. Varadharajan. Trust2: Developing trust in peer-to-peer environments. In *International Conference on Services Computing*, pages 24–31, Orlando, USA, 2005. IEEE.

[17] Y. Wang, H. B. Zhang, and X. Z. Zhang. A trust vector approach to contextual transaction trust evaluation in e-commerce environments. *Technical Report, Macquarie University*, 2012.

[18] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities. In *IEEE International Conference on E-Commerce*, pages 275–284, San Diego, California, USA, 2003. IEEE.

[19] H. B. Zhang, Y. Wang, and X. Z. Zhang. Transaction similarity-based contextual trust evaluation in e-commerce and e-service environments. In *IEEE International Conference on Web Services*, pages 500–507, Washington DC, USA, 2011. IEEE.