Context-Aware Transaction Trust Computation in E-Commerce Environments

by

Haibin Zhang

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy
in the
Department of Computing
Faculty of Science
Macquarie University

Supervisor: Supervisor: A/Prof. Yan Wang
Associate Supervisor: Prof. Mehmet A. Orgun

2014

Statement of Candidate

I certify that the work in this thesis entitled **"Context-Aware Transaction Trust Computation in E-Commerce Environments"** has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree to any other university or institution other than Macquarie University.

I also certify that the thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged.

In addition, I certify that all information sources and literature used are indicated in the thesis.

Haibin Zhang

1 August 2014

To my parents,

Shuyun Ma and Fa Zhang,

who dedicated all their life to me

Abstract

In e-commerce environments, the trustworthiness of a seller is very important to potential buyers, especially when the seller is not known to them. Most existing trust evaluation models compute a single value to reflect the general trustworthiness of a seller without taking any transaction context information into account. With such a result as the indication of reputation, a buyer may be easily deceived by a malicious seller in a transaction where the notorious *value imbalance* problem is involved, namely, a malicious seller accumulates a high level reputation by selling cheap products then deceives buyers by inducing them to purchase more expensive products.

This thesis aims to systematically investigate some key and open challenging research problems in context-aware transaction trust computation in e-commerce environments. In general, it includes our work from the following two aspects.

The first aspect is the trust vector based approach to context-aware transaction trust evaluation. In contrast to most existing trust management models that compute a single trust value, a trust vector is first presented consisting of three major values for Contextual Transaction Trust (CTT). In the computation of CTT values, three identified important *context dimensions*, including Product Category, Transaction Amount and Transaction Time, are taken into account. In the meantime, the computation of each CTT value is based on both past transactions and the forthcoming transaction. In particular, with different parameters specified by a buyer regarding context dimensions, different sets of CTT values can be calculated. As a result, all these trust values can outline the reputation profile of a seller that indicates his/her dynamic trustworthiness in different products, product categories, price ranges, time periods, and any necessary combination of them. We term this new model as *ReputationPro*. Nevertheless, in *ReputationPro*, the computation of reputation profile requires new data structures for

appropriately indexing the pre-computation of aggregates over large-scale ratings and transaction data in three context dimensions, as well as novel algorithms for promptly answering buyers' CTT requests. In addition, storing pre-computed aggregation results consumes a large volume of space, particularly for a system with millions of sellers. Therefore, reducing storage space for aggregation results is also a great demand.

The second aspect is efficient computation of a seller's reputation profile. Towards efficient computation of CTT values aiming at outlining a seller's reputation profile, four index schemes have been proposed. We first extend the approaches to the two-dimensional (2D) Range Aggregate (RA) problem as the preliminary solutions for CTT computation. They are effective approaches, but have low efficiency in computing CTT values in some cases. Then, to overcome the problems in the preliminary solutions, a new disk-based index scheme and a new query algorithm are further proposed. Compared with the preliminary solutions, when answering a buyer's CTT queries for each brand-based product category, the new index scheme has almost linear query performance. This is a significant advantage in answering queries on CTT values especially when a large number of buyers are accessing a seller's reputation data simultaneously. In addition, several strategies are proposed for storage space reduction in CTT computation. These strategies include aggregating ratings and transaction data at different time granularity as well as deleting the index records that are generated based on the ratings and transaction data from remote history. The experiments conducted on synthetic datasets generated from eBay datasets have demonstrated that the proposed *ReputationPro* model can be more effectively applied to large-scale e-commerce websites in terms of efficiency and storage space consumption.

Acknowledgments

First of all, I would like to express my sincere thanks to my supervisor, A/Prof. Yan Wang, who has offered me this opportunity to pursue my doctoral study at the Macquarie University. Also, I would like to express my deepest appreciation to him for his kindness and patience. He has led me in the correct direction over the past few years not only with his knowledge and experience, but also with attitude for high quality research. Literally, without his continuous support and endless guidance, this work would not have been possible. It is my great fortune to have him as my supervisor at Macquarie University.

I also would like to thank Senior Lecturer Dr. Xiuzhen (Jenny) Zhang, co-supervisor Prof. Mehmet Orgun and Prof. Ee-Peng Lim for their help in various stages of my work. This academic journey would not have been so rewarding without their kindness and wisdom.

In addition, my colleagues have helped me to develop this work. I wish to express my thanks to Dr. Lei Li, Dr. Guanfeng Liu, Joe Zou, Lie Qu and Xiaoming Zheng for providing a friendly and enjoyable environment during these years.

Many thanks to the staff in the Department of Computing for their administrative help. I would like to thank Sylvian Chow, Melina Chan, Donna Hua, Camille Hoffman and Jackie Walsh for their warm support and help.

Most important of all, I would like to thank my family. My parents, Shuyun Ma and Fa Zhang, have always been there for me. Their love, support and encouragement have been the foundation for my life. Without their love, unwavering support and inspiration, this work could never have been accomplished.

Publications

This thesis is based on the research work I have performed with the help of my supervisors and other colleagues during my PhD program at the Department of Computing, Macquarie University between 2010 and 2014. Some parts of my research have been published/submitted in the following papers:

[1] **Haibin Zhang**, Yan Wang, Xiuzhen Zhang and Ee-Peng Lim, ReputationPro: The Efficient Approaches to Contextual Transaction Trust Computation in E-Commerce Environments, ACM Transactions on the Web (TWEB), under second round review, 2014.

[2] **Haibin Zhang**, Yan Wang and Xiuzhen Zhang, The Approaches to Contextual Transaction Trust Computation in E-Commerce Environments, Security and Communication Networks Journal (SCN), doi:10.1002/sec.839, Preprint, available at http://onlinelibrary.wiley.com/doi/10.1002/sec.839/abstract.

[3] **Haibin Zhang** and Yan Wang, A Novel Model for Contextual Transaction Trust Computation with Fixed Storage Space in E-commerce and E-service Environments, IEEE International Conference on Services Computing (SCC 2013) (**CORE2013 rank A conference**), pages 667-674, 27 June - 2 July, 2013, Silicon Valley, California, USA.

[4] **Haibin Zhang**, Yan Wang and Xiuzhen Zhang, A Trust Vector Approach to Transaction Context-Aware Trust Evaluation in E-Commerce and E-Service Environments, IEEE International Conference on Service-Oriented Computing Applications (SOCA 2012), pages 1-8, 17-19 December, 2012, Taipei, Taiwan.

[5] **Haibin Zhang**, Yan Wang and Xiuzhen Zhang, Efficient Contextual Transaction Trust Computation in E-Commerce Environments, IEEE International Confer-

ence on Trust, Security and Privacy in Computing and Communications (TrustCom 2012) (**CORE2013 rank A conference**, **the Best Paper Award**), pages 318-325, 25-27 June, 2012, Liverpool, UK.

[6] **Haibin Zhang**, Yan Wang and Xiuzhen Zhang, Transaction Similarity-Based Contextual Trust Evaluation in E-Commerce and E-Service Environments, IEEE International Conference on Web Services (ICWS 2011) (**CORE2013 rank A conference**), pages 500-507, 4-9 July, 2011, Washington DC, USA.

[7] Xiaoming Zheng, Yan Wang, Mehmet A. Orgun, Guanfeng Liu and **Haibin Zhang**, Social Context-aware Trust Prediction in Social Networks, International Conference on Service Oriented Computing (ICSOC 2014) (**CORE2013 rank A conference**), accepted, 3-6 November, 2014, Paris, France.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"*Trust is the glue of life. It's the most essential ingredient in effective communication. It's the foundational principle that holds all relationships.*" – *Stephen Covey*

Trust, in society, plays an important role that, literally, is the basis for interpersonal relationships. For example, we trust our neighbours to leave the house keys with them, so they can look after our house while we are overseas. Similarly, our friends trust us, so will lend us money if we find ourselves in a debt crisis. In these cases, trust has been established, based on past experiences and direct interactions.

With the rapid development of Internet and Web technologies, people are more active in various large, open network systems, including social networks (e.g., Facebook[1] and Twitter[2]), Peer-to-Peer systems (e.g., uTorrent[3] and iMesh[4]) and e-commerce (e.g., Amazon[5] and eBay[6]). In particular, since the mid-1990s, many e-commerce applications have emerged and have been attracting a large number of customers, such as Amazon (founded in 1994) and eBay (founded in 1995), which have about 145 million [10] and 233 million [33] customers worldwide, respectively. According to statistics provided by comScore[7] (an American Internet analytics company), on 26 November 2013 [26], e-commerce spending in the United States during 2013 (from

[1] http://www.facebook.com/
[2] http://www.twitter.com/
[3] http://www.utorrent.com/
[4] http://www.imesh.com/
[5] http://www.amazon.com/
[6] http://www.ebay.com/
[7] http://www.comscore.com/

January to October) was $164.0 billion which is 14% more than for the same period in 2012. Apparently, e-commerce, or online trading, has greatly changed people's traditional behaviours, and the changes are not only limited to the spending patterns, but also extend to trust management.

Within e-commerce systems, which typically form a virtual community, people (sellers and buyers) do not meet or interact physically. Consequently, in contrast to direct interactions, trust establishment between sellers and buyers is usually through the *witness information* or *word-of-mouth* [121]. For example, at eBay, a centralised trust management system was developed to evaluate the reputation of sellers. After each transaction, a buyer has an opportunity to provide a rating (+1, 0, or -1) to the centralised trust management system based on the transaction quality. The ratings given to a seller are accumulated over a recent time period (e.g., one month or six months) and a single positive feedback rate is calculated as the indication of the seller's trustworthiness or reputation score. This single positive feedback rate can be used by buyers to select a trustworthy seller from a large pool of sellers, particularly when the sellers are not known to the buyer.

1.1 Motivation

1.1.1 Background and Problem Statement

In e-commerce environments, the trustworthiness of a seller is a crucial issue in the decision-making process prior to a purchase [90, 40, 121, 61, 68, 16]. The rating system used at eBay sets a good example for managing a seller's trustworthiness or reputation. However, such a simple trust management system is vulnerable to some frauds from malicious sellers [66, 113, 50, 58]. As shown in Fig. 1.1, within eBay community[8], we can see evidence of cheating. For example, a malicious seller can gain a good reputation by honestly selling good and low value (price) products. Once

[8] http://community.ebay.com/

having accumulated a good reputation, s/he may then deceive buyers by inducing them to buy more expensive products, but either not delivering the ordered product, or delivering a fake product. In the literature, this is referred to as the *value imbalance* problem [66, 29, 58, 67, 57] and such malicious sellers are named as *traitors* [92, 50]. Likewise, several real world cases are reported in [113]. For instance, an Australian traitor, who traded honestly by selling cheap products to gain a positive profile before selling expensive products, defrauded people of more than AU$10K in total. A Californian traitor, in the same way, managed to earn a high positive feedback rate before defrauding buyers of over US$300k.

In view of this problem, Zhang et al. [170, 172] identified the key issues related to *value imbalance* in transactions, as outlined below.

- **The lack of consideration of context in transaction trust evaluation**: In e-commerce environments, different transactions generally have different *natures* and *contexts*; even the same seller needs to be considered differently with regard to the trustworthiness in different forthcoming transactions [164, 158, 159, 121, 150, 149, 74, 111]. Actually, the *value imbalance* is only a type of the *context imbalance* [172] in transactions, where imbalance can also exist in product categories. For example, following a few cases of fraud at Alibaba[9], which is founded in 1999 and supports both B2B (Business to Business) and B2C (Business to Consumer) online trading, buyers are explicitly reminded to manually check if the products offered by a supplier fall into in the same categories as the products that the supplier usually sells [9]. This example also indicates that reputation-based transaction trust evaluation should be "transaction context-aware".

- **The static results of trust evaluation**: Most models compute a single trust value based on past transactions [120, 64, 159, 153, 154, 151, 88, 146]. However, such a single value basically only reflects a seller's general or global trust status, and is static with regard to any forthcoming transactions [149]. As illustrated

[9] http://www.alibaba.com/

	Feedback	From/price	Date/time
⛔	terrible was very unhappy when I recieved item Sexy Gorgeous Burlesque Corset & tutu/skirt Fancy dress outfit Halloween Costume (#310635697613)	Buyer: [redacted] (0) US $20.99	12-Jun-13 04:50 View Item
⛔	Item did not work, it smelt burned when i tried to turn it on H.264 HD 2.7" LCD car camera recorder vehicle rearview mirror DVR video dash cam (#111050422022)	Buyer: [redacted] (25 ★) US $57.50	12-Jun-13 01:48 View Item
⛔	I have got model GS1000 ??? This primitiv camera $20 Novatek GS8000L 1080P HD Car DVR Vehicle Camera Recorder Dash Cam G-sensor HDMI (#310653720801)	Buyer: [redacted] (0) US $57.50	11-Jun-13 03:14 View Item
⛔	article non reçu pas de retour de monnaie --	Buyer: [redacted] ★) --	10-Jun-13 02:44 Private
⛔	article non reçu pas de retour de monnaie --	Buyer: [redacted] ★) --	10-Jun-13 02:43 Private
⛔	paid extra for the 1080p camera and received the 720p unit	Buyer: [redacted] ★)	09-Jun-13 10:33

Figure 1.1: The real cheating behaviours happened at eBay

above, different transactions may have different contexts. The static trust evaluation of a seller can hardly predict the likelihood of a successful forthcoming transaction. Thus, trust evaluation should be associated with both past transactions and the forthcoming transaction.

1.1.2 New Challenges in Transaction Trust Computation

Now, let us consider a simple example. Suppose a malicious seller S_1 has completed 198 transactions with good quality selling "*AT&T SIM Card*" at the price of $1 and obtained 198 positive ratings. The seller S_1 has also completed another 2 transactions with poor quality selling "*Apple iPhone5s 16GB*" at the price of about $700 and obtained 2 negative ratings. Based on the trust evaluation model used at eBay, the trustworthiness of S_1 is as high as 0.99. Next, consider a scenario that a buyer B plans to buy an "*Apple iPhone5s 16GB*". In the meantime, the seller S_1 is selling this product, and the price $700 offered by S_1 is cheaper than other sellers. Clearly, the seller S_1 is very attractive and appears to be trustworthy as well. Thus, the buyer B tends to buy the "*Apple iPhone5s 16GB*" from S_1. In such a case, B is likely to suffer monetary loss. However, in addition to the general trust value 0.99 to buyers, if B knew that S_1 received negative ratings in occurred transactions selling "*Apple iPhone5s 16GB*", B would not purchase from S_1. In fact, from buyers' point of view, they are more concerned about the trustworthiness of a seller in a potential forthcoming transaction, rather than a general trust value resulting from all occurred transactions.

Suppose a seller S_2 has completed many more transactions, selling products in a variety of categories over a long period of time. When the buyer B plans to buy a "*Canon EOS 6D SLR Digital Camera*" at the price of around $1600 from S_2, in addition to the trustworthiness of S_2 in selling this product, B could also be concerned about the trustworthiness of S_2 in selling "*Canon DSLR camera*" with a price range of "$1000-$2000" (i.e. a query w.r.t. a higher layer in the hierarchical product category in a price range) in *the latest 3 months* or *the latest 6 months*. This is particularly the

case when the product in the forthcoming transaction is just available in market and the number of existing transactions selling this product is quite low or even zero. If S_2 is reputable in all these related transactions, there should be good reasons for B to trust S_2 in a new transaction for purchasing a "*Canon EOS 6D SLR Digital Camera*" at the price of around \$1600. Otherwise, if S_2 has problems in the transactions in a certain product category or a certain price range (e.g., S_2 received a lot of negative ratings in the transactions in selling "*Canon EOS 6D SLR Digital Camera*"), it is necessary for the trust evaluation to indicate the flaw of S_2 in reputation.

The above process follows the suggestion provided on the Alibaba website, which advises buyers to check if the product to be purchased from a seller is in the categories in which the seller usually sells and if the existing transactions in these categories are reputable. Similarly, as a buyer is very concerned about the possibility of monetary loss, trust evaluation needs to indicate trustworthiness over different price ranges, each of which takes the price of the product to be purchased as approximately the medium value. In addition to them, a further step is to evaluate trust over the combination of product category and price range, as well as time period, as different buyers buy products in different categories and with difference prices from the same seller. Such evaluation results can reveal potential risk if a seller has problems in reputation in the transactions in a product category, a price range and a time period, related to the potential new transaction that the buyer plans to complete with the seller.

Obviously, these identified needs cannot be satisfied by any single-value trust evaluation model. In the meantime, the new needs bring challenges to trust computation as a long-existing seller usually has large-scale transactions with a wide variety of product categories as well as a wide price range. These challenging problems will be addressed in this thesis.

Broadly speaking, the context of transactions can be modelled or described by transaction *dimension*. Let us suppose that the number of identified transaction context dimensions is n and the cardinality for each dimension is m (i.e. each dimension takes m values). Basically, in order to promptly answer a buyer's request on trust-

worthiness of a seller for a specific transaction context, the corresponding trust value can be pre-computed and kept in a multi-dimensional structure (e.g., a matrix). Obviously, a brute-force algorithm for computing this multi-dimensional structure incurs a computational complexity of $O(m^n)$. Therefore, the computation of a seller's trustworthiness in various transaction contexts incurs high computational complexity.

In summary, this thesis is not confined to the approaches designed to avoid some attacks from malicious sellers [67, 58], but systematically investigates some key and open challenging research problems in context-aware transaction trust computation in e-commerce environments. These problems include identifying important context dimensions, proposing a new trust evaluation model that takes identified transaction context dimensions into account, and giving technical solutions, i.e. designing new data structures and novel algorithms that can efficiently and comprehensively evaluate the contextual trust of large-scale online transactions, and prevent frauds and monetary loss. The detailed contributions of the thesis are given in the following section.

1.2 Contributions of the Work

This thesis makes a contribution in the following three major aspects.

1. The first contribution of this thesis lies in the proposed trust vector based approach to context-aware transaction trust evaluation.

   (a) We distinguish the definitions of context in different applications, and focus on discussing and defining transaction context in e-commence environments. In particular, we identify three important transaction context dimensions, i.e. *Product Category*, *Transaction Amount (Price)* and *Transaction Time* with influence on the trustworthiness of a forthcoming transaction.

   (b) In situations where there are no, or not enough, ratings from the past transactions with the same context as the forthcoming transaction, we provide a set of methods to calculate transaction context similarity, the results of

which are used in inferring the trustworthiness of a forthcoming transaction. Our solution, potentially, has wide applications in e-commerce systems.

(c) In contrast to most existing trust valuation models that compute a single trust value [120, 27, 64, 159, 163, 154, 88, 146], we present a trust vector based approach consisting of three major trust values, which are also termed as *CTT* (Contextual Transaction Trust) values. In the computation of CTT values, three identified context dimensions are taken into account. In the meantime, the computation of each CTT value is based on both past transactions and the forthcoming transaction. In particular, with different parameters specified by a buyer regarding context dimensions, different sets of CTT values can be calculated. As a result, all these trust values can outline the reputation profile of a seller that indicates his/her dynamic trustworthiness in different products, product categories, price ranges, time periods, and any necessary combination of them.

Finally, we have studied the effectiveness of our proposed trust vector based approach both analytically and empirically. In particular, it can reflect a seller's dynamic trustworthiness in different transaction contexts and identify risks potentially existing in a forthcoming transaction, thus outperforming single-value trust valuation methods [120, 154] and a prior trust vector based approach [149].

2. The second contribution of this thesis lies in the proposed efficient approaches to the computation of Contextual Transaction Trust (CTT) values.

At e-commerce websites, a popular seller usually sells a wide variety of products distributed in a number of product categories. In addition, a large number of buyers can be accessing one seller's reputation data simultaneously with regard to their potentially forthcoming transactions in various contexts. In order to promptly answer a buyer's requests on CTT values, it is necessary to pre-

compute aggregates over large-scale transaction data and ratings with necessary combinations of three context dimensions. Thus, our proposed trust vector based approach for outlining sellers' reputation profiles is a very challenging problem that requires new data structures and novel algorithms. Note that, as illustrated in Section 1.1.2, it may incur the total computational complexity of $O(m^n)$.

(a) As mentioned above, with all trust results computed by the trust vector, the reputation profile of a seller can be outlined, which greatly helps to identify the *value imbalance* problem that may exist in forthcoming transactions, and thus avoids monetary losses for buyers. We term this new model as *ReputationPro*. In addition, we term the query on CTT values as a *CTT query*, and term the computation of CTT values as *CTT computation*.

(b) In the literature, our targeted CTT computation problem is similar to data warehousing and OLAP (On-Line Analytical Processing) technology [25]. In particular, the traditional RA (Range Aggregate) [104] in spatial data warehouses is relatively close to CTT computation. Thus, to address CTT computation problem, we first provide three approaches, i.e. *eaR-tree*, *eaP-tree* and *eH-tree*, by extending the existing approaches to the two-dimensional (2D) RA problem. To the best of our knowledge, this thesis is the first work in the literature which evaluates the reputation profile of a seller taking advantage of related techniques in data warehousing. Experiments conducted on both eBay datasets and large-scale synthetic datasets illustrate the advantages and disadvantages of three approaches when answering a buyer's CTT queries.

(c) Although three disk-based approaches have been proposed that meet the requirements of CTT computation, they have low efficiency in computing CTT values in some cases. Towards efficient CTT computation, a new disk-based index scheme *CMK-tree* and a new query algorithm are proposed. We have proved that while answering a buyer's CTT queries for

each brand-based product category, the *CMK-tree* has almost linear query performance. This is a significant advantage in answering CTT queries when a large number of buyers are accessing a seller's reputation data simultaneously.

3. The third contribution of this thesis lies in the proposed strategies for storage space reduction in Contextual Transaction Trust (CTT) computation.

   Generally speaking, the approaches to CTT computation pre-compute the aggregates over historical large-scale ratings and transaction data of a seller. Then, the aggregation results are stored appropriately in the specialised index forming a tree structure. Nevertheless, with continuous growth in transaction time and significant increase of transaction data, the size of additional storage space for storing the aggregation results can become much larger.

   To address this problem, we propose several strategies to reduce that additional storage space consumption. With these strategies, we make the new trust evaluation model *ReputationPro* more effectively apply to large-scale e-commerce websites in terms of efficiency and storage space consumption.

   (a) In the literature, Zhang et al. have proposed a Hierarchical Temporal Aggregation model with fixed storage space (*HTA*^{FS}) to deal with temporal aggregation over data streams [165]. According to our analysis, we firstly point out the limitations of *HTA*^{FS} model in solving our targeted problem. Then, a new solution CTT^{FS} model for CTT computation is introduced. The CTT^{FS} model guarantees the fixed storage space for storing the aggregation results as well as good performance in responding to CTT queries. Also, the experiments are conducted on both an eBay dataset and a large-scale synthetic dataset to validate our proposed structure and approach.

   (b) As stated above, the *HTA*^{FS} can be applied to CTT computation to control the size of the storage space allocated to a seller. However, it is difficult

to select the size of the fixed storage space since the number of distinct products, as well as the number of product categories in the transactions traded by different sellers are different. Therefore, a new strategy is adopted, which aggregates ratings with different time granularities for different time periods. We apply this strategy to the *CMK-tree* and propose the *CMK-tree*^{RS} approach. According to experimental results, our proposed *CMK-tree*^{RS} can further reduce the storage space allocated to each seller as well as the time of computing the CTT values with a little loss in accuracy of CTT computation.

(c) In solving the large storage space consumption, a fundamental idea is to delete index records that are generated based on ratings and transaction data from "remote" history (e.g., "*the 12 months ago*"). As stated before, the index scheme *CMK-tree* is considered as the most efficient approach to CTT computation. However, in order to achieve nearly linear query performance, the deletion operations in the *CMK-tree* become quite complicated. Therefore, in this thesis, we propose three different deletion strategies for the *CMK-tree*. Also, we have conducted experiments to illustrate both advantages and disadvantages of our proposed deletion strategies.

1.3 Roadmap of the Thesis

This thesis is structured as follows.

Chapter 2 presents a comprehensive literature review of the basic concepts of trust management as well as trust evaluation methods, context-aware trust evaluation and related techniques in data warehousing.

Chapter 3 first presents the definition transaction context, then proposes a set of methods to compare similarity between transaction context dimensions, and finally, introduces our proposed trust vector based approach to context-aware transaction trust evaluation. This chapter includes our papers [170, 172] published at IEEE ICWS 2011

and IEEE SOCA 2012 (please refer to [6] and [4] in the publication list on page ix). This chapter is also the basis of whole thesis.

The remainder of this thesis covers the following two aspects.

The first aspect is the proposed efficient approaches to CTT computation, which is discussed in Chapters 4 and 5. In Chapter 4 we present the differences between Two-Dimensional Range Aggregate (RA) and our targeted CTT computation problem, and then, we propose three preliminary solutions, *eaR-tree*, *eaP-tree* and *eH-tree* for CTT computation. This chapter is based on the papers [171, 173] published in IEEE TrustCom 2012 and Security and Communication Networks Journal (SCN) (please refer to [5] and [2] in the publication list on page ix). In Chapter 5, we propose the *CMK-tree*, which is an efficient approach to CTT computation and a new query algorithm. The experimental results illustrate that the *CMK-tree* is superior in efficiency when computing CTT values to all three existing approaches in the literature. This chapter is based on the paper submitted to ACM Transactions on the Web (TWEB) [174] (please refer to [1] in the publication list on page ix).

The second aspect of the work concerns storage space reduction strategies in computing CTT values. Firstly, in Chapter 6, based on HTA^{FS} model, we propose a new solution CTT^{FS} model for CTT computation. This part includes our paper [169] published at IEEE SCC 2013 (please refer to [6] in the publication list on page ix). Secondly, in view of the drawbacks of CTT^{FS} model, we propose another strategy to reduce the storage space consumption and apply to the *CMK-tree*, i.e. $CMK\text{-}tree^{RS}$. This part includes the paper submitted to ACM Transactions on the Web (TWEB) [174] (please refer to [1] in the publication list on page ix). Thirdly, we present three deletion strategies for *CMK-tree*, which can eliminate the index records in the *CMK-tree* that are generated based on ratings and transaction data from "remote" history.

Finally, Chapter 7 concludes the work in this thesis and discusses some directions for future research opportunities.

Chapter 2

Literature Review

"*Learning to trust is one of life's most difficult tasks.*" *–Isaac Watts*

"*Trust is a complex and slippery notion,....*" *–Bart. Nooteboom*

"*...trust is a term with many meanings.*" *–Oliver. Williamson*

Trust is a complex subject relating to belief, truth, competence and reliability between a *trustor* (i.e. the subject that trusts a target entity) and a *trustee* (i.e. the entity that is trusted). After recognising its importance, scientists and researchers from different disciplines including sociology, psychology, economics and computer science, attempted to define the meaning of trust, but without consensus [40]. Moreover, from the perspective of computer science, the trust management systems have been extensively studied in different application environments such as Peer-to-Peer (P2P) networks [131], cloud environments [100], e-commerce environments [61], web services [155] and social networks [124].

In e-commerce environments, an effective trust management system can help both sellers and buyers obtain the maximum benefit [141]. In addition, from the perspective of economic benefits, large differences in selling prices have been observed between new sellers (no reputation) and sellers with a good reputation [14]. Furthermore, the sellers with excellent trustworthiness are more likely to sell items successfully than those with poor trustworthiness [109, 110]. In light of this, over recent years, trust has received much attention from researchers to build various trust management systems [120, 158, 88, 146, 173].

In this chapter, from the perspective of the general concept of trust to a specific perspective of context-aware trust evaluation, we present a comprehensive review. In this way, the contributions of the whole thesis can be more clearly understood. We organise this chapter as follows:

- Section 2.1 introduces different definitions of trust, the trust properties and basic processes of building a trust management system. This section is helpful to have a global picture of trust management.
- The trust evaluation approach is the most essential part for any trust management system, thus Section 2.2 focuses on classifying the existing trust evaluation approaches from different perspectives.
- In the literature, it has already been recognised by increasingly more researchers that "trust evaluation should be context dependent". In Section 2.3, we first discuss the definition of context in different application environments. Then, we introduce existing context-aware trust evaluation approaches and focus on highlighting the differences from our work and its contributions.
- Section 2.4 reviews related techniques in data warehousing and OLAP (On-Line Analytical Processing), since they have been improved to solve our targeted problem in this thesis.
- Finally, Section 2.5 summarises our work in this chapter.

2.1 Overview of Trust Management

The notions of "trust management" and "trust management system" first appear in [20], which are originally defined to evaluate security policies in traditional distributed systems. The concept of trust management then spread to many different application environments. In order to have a global picture of trust management, we first introduce the basic concept of trust.

2.1.1 Meanings of Trust in Different Disciplines

The Oxford Reference Dictionary[1] states that trust is the "firm belief in the reliability, truth, or ability of someone or something". From this definition, we can see the trustee should be highly reliable and have honest performance when interacting with trustor. The concept of trust is not new, but people can have different understandings of it. In the literature, in addition to these definitions, scientists have defined the meaning of trust in different disciplines [91].

2.1.1.1 Trust in Psychology

In psychology, Deutsch [30] states that trust is considered to be a psychological state (perception) of the trustor (the individual), where the trustor is willing to take a risk after balancing both costs and benefits. In addition, from a psychological perspective, Beatty et al. [16] point out that trust should include three aspects: cognitive, emotive and behavioural. The cognitive aspect of trust refers to a rational decision [11] based on the trustor's evaluation of the trustee; the emotive aspect of trust refers to nonrational decision but emotional drive [137]; the behavioural aspect of trust refers to the final actions that makes the trustor vulnerable to the trustee.

2.1.1.2 Trust in Sociology

In sociology, Sztompka [133] defines trust as "a bet about the future contingent actions of others". To some extent, the trust definitions for both Deutsch and Sztompka are similar, which are considered as the expectation of unknown.

In fact, the above similarity is due to the fact that sociology contains two parts: individual (micro level) and societal (macro level). At the individual level, the definition of trust is similar to the perspective from psychology [117, 124]. At the societal level, trust is considered to be a property of social groups. For instance, Luhmann [86] considers trust as "a means for reducing the complexity of society", as people in society

[1] http://www.oxfordreference.com/

generally obey certain rules and are, thus, predictable.

However, trust at the individual level and trust at the societal level is not the same thing. In order to explain these differences, we introduce two forms of trust: "relational trust" [117] and "generalised trust" [116]. Relational trust at the individual level refers to specific trust between the trustor and trustee. It is built up through repeated direct interactions between two parties and declines when trust is betrayed [117]. By contrast, generalised trust at societal level refers to a general belief of the trustor towards a group of members, such as "scientists are always stiff". In human society, the generalised trust allows an initial trust relationship between unfamiliar trustor and trustee, and then relational trust is established through further interactions between them. As pointed out by Marsh [91], studying the phenomenon at either level relational or generalised but ignoring the other leads to the inevitable loss of understanding of trust as a personal and a social concept.

2.1.1.3 Trust in Economics

In economics, the European Commission Joint Research Center [54] defines trust as "trust is the property of a business relationship, such that reliance can be placed on the business partners and the business transactions developed with them". This definition also implies that trust plays an important role in commercial relations.

To be precise, trust can affect transaction itself, in particular, it will mitigate information asymmetry in transactions [8, 14]. During transactions, information asymmetry may lead to opportunistic behaviour of one party that deceives the other to increase his/her profits, and this phenomenon is quite evident in online trading. Buyers who are participating in online trading cannot judge product quality in advance, as it is difficult to confirm that the product quality, as posted by a seller, is accurate. Even the information asymmetry may generate price premiums for the sellers who are more trustworthy [14]. In summary, some economists consider trust as a mechanism that restricts opportunistic behaviour, and it makes two parties establish a reciprocal relationship during transactions.

2.1.1.4 Trust in Computer Science

In computer science, Marsh is among the pioneers who introduced a computational model for trust [91]. Broadly speaking, the discussion of trust in computer science can be classified into two different aspects: "target" and "process".

The definition of trust, from the "target" aspect, is the same as that of psychology, sociology and economics [91]. More specifically, the work in this aspect focuses on the subjective property of trust. For example, Mui defines trust as "a subjective expectation an entity has about another's future behavior" [97]. Similarly, Jøsang et al. combine both psychology and economics, and state that "Trust is the subjective probability by which an individual expects that another performs a given action on which its welfare depends" [61]. From the "process" aspect, computer science researchers design various trust evaluation models to help users of trust management systems establish trust relationships. Usually, these models attempt to simulate the process of trust establishment amongst people in human society. In the following, we take eBay (one of most popular online shopping systems) as an example to explain the process of trust establishment amongst users.

Since most eBay users are not familiar with each other, eBay provides a rating system to assist buyers and sellers. After each transaction, a buyer can provide a rating (+1, 0, or -1) to a trust management system according to transaction quality. In this case, when a potential buyer is planning to buy a product from an unfamiliar eBay seller, the initial trust establishment of the buyer is based on the experiences (ratings) of others, i.e. "recommendation trust" [2]. Normally, if a seller has a high level of "recommendation trust", it implies that s/he is trustworthy. However, this initial level of trust may rise, or decline, after direct interactions (transactions) between the buyer and the seller, and that is "relational trust" [117]. In addition, eBay has identified "top-rated sellers"[2], who consistently deliver outstanding customer experiences, which helps an initial trust relationship to be established between an unfamiliar buyer and a

[2]http://pages.ebay.com.au/help/sell/top-rated.html

seller. This is an example of "generalised trust" [116]. Of course, the members in this group may then generate price premiums, as they have been identified as being more trustworthy [14].

Other popular trust evaluation models, such as EigenTrust [64], PeerTrust [159] and PowerTrust [180], adopt the idea of "recommendation trust" and "relational trust" for trust establishment. To sum up, the prototype of trust definition in computer science, regardless of "target" and "process", can be found or explained from perspectives of psychology, sociology and economics. More importantly, they give a solid theoretical foundation for the discussion of trust in computer science.

2.1.1.5 Trust Related Concepts

This subsection introduces two concepts related to trust: *risk* [62] and *reputation* [2].

Trust is always accompanied by risk. As pointed by Bohnet [53], the willingness to trust is closely associated with the willingness to take risk. Ruohomaa [118] even defines trust using the concept of risk as "Trust is the extent to which one party is willing to participate in a given action with a given partner, considering the risks". Risk has also been studied in many disciplines, including psychology, economics and computer science. In psychology, risk is the psychological state in emotion facet such as fear [73] and in cognitive facet such as uncertainty [117], while in economics, risk is tied to money or profit and thus affects trust decision [53]. In computer science, most researchers acknowledge that the terms of trust and risk are in an inverse relationship, with a high level of risk linked to a low level of trust and vice versa [106, 62, 156].

Reputation is another trust related concept, and its discussion depends on the past behaviours of an individual. For instance, Abdul-Rahman defines reputation as "an expectation about an individual's behaviour based on information or observations of its past behaviours" [2]. Likewise, Mui defines reputation as "a perception a party creates through past actions about its intentions and norms" [97]. Clearly, the reputation is the result of the accumulation of past behaviours, or it can be considered as the statistical result of "recommendation trust" (experiences of others) [2]. In [61], Jøsang et al.

present the following two statements to differentiate trust and reputation:

(a) *I trust you because of your good reputation.*

(b) *I trust you despite your bad reputation.*

The first statement reflects that reputation is essential for two unfamiliar parties to establish an initial trust relationship. That is because, in this case, experiences of others ("recommendation trust") is the only reliable information on which a party can make a judgement. Therefore, the higher the reputation, the more trustworthy the party is considered to be. The second statement reflects that reputation is redundant for "relational trust", where the trust is formed by direct interactions [117]. That is, if two parties have direct interaction, the experiences of others is no longer useful. These two statements also suggest that trust is a subjective phenomenon [97, 61].

2.1.2 Properties of Trust

Following on from the discussion of the meanings of trust, this section focuses on introducing some trust properties. These proposed properties are based on either experimental verification or long-term observations of human activities, which provide the theoretical foundation for researchers in computer science to design various trust evaluation approaches.

2.1.2.1 Trust is Subjective

It is well acknowledged that trust is subjective, regardless of disciplines. For example, in psychology, trust is defined as a personal psychological state [116], and it reflects the trustor's subjective attitude, or personal opinion, towards the trustee. For the same person, however, different people may have different opinions. For instance, Alice trusts Bob, since Bob always keeps promise during their interactions, but, due to a betrayal, Cathy does not think Bob is trustworthy.

In computer science, the subjective property of trust is one of the major aspects in trust discussion. Jøsang took advantage of subjective logic (logic that operates on subjective beliefs about the world), to explain trust [55, 56]. To be precise, Jøsang's model contains three parts: *belief (b)*, *disbelief (d)* and *uncertainty (u)*, which satisfy an equation as $b + d + u = 1$. Jøsang further explains that an opinion can be uniquely described as a point in the triangle where belief, disbelief and uncertainty are three vertices. Moreover, in e-commerce or service-oriented environments, the above theory is applied to evaluate the subjective trustworthiness of a consumer during sellers or services selection [75]. In particular, some mathematical theorems have been proposed to depict the changes of subjective trustworthiness. For example, with an increase in the total number of positive/negative experiences (ratings) from other consumers, the uncertainty decreases and belief/disbelief increases.

In addition, trust is a subjective phenomenon that is also treated as personalisation. For example, Richardson et al. [112] stress that trust is a personal trust, and the ratings given by users in a trust management system belong to personal ratings.

2.1.2.2 Trust is Dynamic

Having introduced the subjective property of trust, it is easy to understand that trust changes dynamically with time. Specifically, based on repeated direct interactions, trust can build up, or decline, with new experiences [117]. Meanwhile, the dynamics of trust also reflects a temporal characteristic of trust. In the literature, the concerns of trust and time have been discussed in three major ways: trust decay, trust time window and their hybrid.

The experiences will fade over time, thus trust decay refers to the decline of trust with the fading experiences. This characteristic is widely used in computer science, with a well-acknowledged method being to gradually reduce the influence of old interactions (experiences), or increase the weighting of more recent interactions (experiences) [120, 64, 153, 88, 147]. Of course, this method is reasonable, since new interactions are more important than old ones for evaluating the recent behaviours of a party.

In addition to the fading experiences, Spitz and Tüchelmann [128] point out a phenomenon called inactivity. In particular, this phenomenon is common for e-commerce websites. Inactive members are the sellers where there has been a large lapse of time since their last transaction. With the existing approaches, such as [120, 153], to deal with inactivity, an inactive member will never reduce its trustworthiness as long as no new experiences are made. Apparently, these inactive members are different from the other active members, but there are few studies to evaluate their trustworthiness.

Trust time window is another way of concern the temporal characteristic of trust. In some trust evaluation models, such as PeerTrust [158, 159], users are allowed to choose the recent time window. Similarly, at eBay, the above time window is set to three different time ranges: *"the latest 1 month"*, *"the latest 6 months"* and *"the latest 12 months"*.

At last, the hybrid of trust decay and time window is adopted to reflect the dynamic property of trust [154, 153, 147].

2.1.2.3 Trust is Propagative

Trust propagation is an important property that helps unfamiliar parties to establish a trust relationship. For example, Alice trusts Bob, who trusts Cathy, but Alice and Cathy are not familiar with each other. Then, Alice and Cathy may establish a trust relationship, however, how much Alice trusts Cathy should depend on how much Alice trusts Bob, and how much Bob trust Cathy [162, 59]. Due to trust propagation, the information of trust can pass from one to another, and finally form a trust path or a trust chain. Within a large social network, trust propagation becomes quite complex, since more parties are involved in a trust path.

In e-commerce environments, trust propagation can be considered as a simple form of "recommendation", without a complex trust path. Let us take the rating system used at eBay as an example. The buyer, Alice, gives an eBay seller a positive rating (+1) after a transaction, which implies that she "recommends" this seller to other buyers with regard to his/her trustworthiness. The buyer, Bob, who trusts Alice, may then

carry out a transaction with this eBay seller as well. Likewise, another buyer, Cathy, who trusts either Alice or Bob, may continue to trade with that seller. Clearly, this process is the same as *"word-of-mouth"* in human society.

2.1.2.4 Trust is Context Dependent

Across all the disciplines, Rousseasu et al. provide an extensive review of the concept of trust, and suggest that "research on trust requirements the attention to *context*" [117]. Conceptually, *context* is always bound to *situation*. For example, the Webster Dictionary states context as "the situation in which something happens". Similarly, a widely accepted definition presented by Dey [31] states "context is any information that can be used to characterize the situation of an entity". Rehak et al. [108] point out the difference between context and situation is that situation is the state of reality and context is a formal, simplified representation of the situation. Therefore, the statement, "trust is context dependent", implies different situations require different considerations with regard to trust.

More specifically, some social scientists propose "interpersonal and personal trust" as one of topological categories on trust [94], namely, that one person trusts another person in a specific situation. For example, Alice may trust Bob as a mechanic in the specific context of servicing her car but probably not in the context of babysitting her children [2]. Based on the above considerations, Grandison and Sloman [40] introduce context to define trust as "the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified *context*".

In computer science, Marsh proposes the concept of "situational trust", such as "Whilst I may trust my brother to drive me to the airport, I most certainly would not trust him to fly the plane!" [91]. Namely, even for the same person, different contexts (situations) may deliver different results for his/her trustworthiness. In addition, Mui [97], from the reputation point of view, expands the concept of situational trust and stresses that reputation also depends on the context. "Bill Clinton's reputation as a politician is likely to be very different from his reputation as a cook." It means that,

from past experiences (these experiences are mainly the comments from others), we could only infer that we trust Bill Clinton as a politician instead of a cook.

In order to provide an accurate evaluation on the trustworthiness of a trustee, the calculation of trust needs to consider the contextual information. In recent years, in recognition of the importance of context, more studies have introduced context in trust evaluation [159, 81, 108, 150, 142, 74, 170, 111, 83]. In Section 2.3, we will review some context-aware trust evaluation approaches separately, and context-aware transaction trust evaluation in e-commerce, which is the focus of this thesis.

2.1.3 Building Trust Management Systems

At the beginning of this section, we have pointed out that trust is a term with many meanings, and there is no consensus on its meaning across all the disciplines [72, 40]. Nevertheless, in the field of computer science, the researchers share the same goal of building an effective trust management system. The trust management system aims to model the process of trust establishment in human society, help users identify the trustworthy members and deliver reliable information.

As shown in Fig. 2.1, the trust management systems are built on a logic hierarchy with three levels: infrastructure level (security-oriented trust), service level (service-oriented trust) and community level (socially-oriented trust) [118, 43]. In this section, from the three levels, we give an extended introduction to trust management systems.

2.1.3.1 Logic Hierarchy of Trust Management Systems

In computer science, early trust management systems use security policies to establish trust. These systems are usually applied for the purposes of security and focus on the reliability of information (security-oriented trust [43]) thus traditional security techniques [175, 176], e.g., authentication, encryption and access control, can be adopted. Correspondingly, trust is the extent to which a party is willing to depend on something or somebody, in a given situation, with a feeling of relative security [94]. For example,

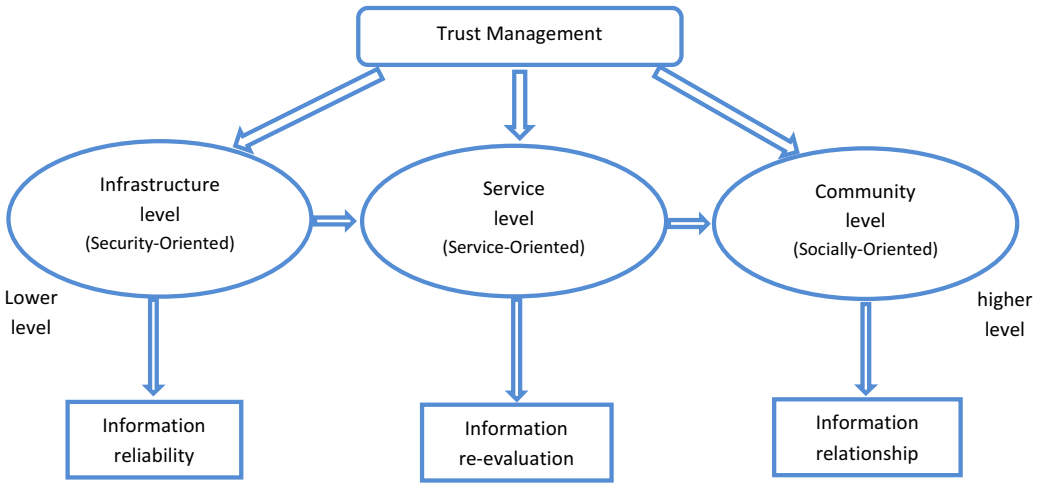


Figure 2.1: Trust management systems built on a logical hierarchy with three levels

the Kerberos protocol [71] takes advantage of a third party to facilitate the exchange of credentials (digital signatures) between a user and a computer. PolicyMaker [20] and KeyNote [19] use signatures authentication to form the trust relationship between a trustor and a trustee. The trust management systems at infrastructure level are more concerned about reliability than about the content of information.

The service-oriented trust management re-evaluates the content of information that refers to direct and indirect experiences or interactions between a trustor and a trustee. In an open environment, such as e-commerce or web services, where members (e.g., service providers and service consumers) can freely access (join and leave) and interact with each other, to evaluate the trustworthiness of a target, security-oriented trust evaluation leads to a large overhead for trust management systems [89]. In such a case, each member typically maintains its own trust information about others, possibly incorporating the recommendations of others, and, finally, trust is established based on a combination of all the information. Usually, the above process is also termed as reputation-based trust evaluation [118, 119, 121, 131, 155, 50]. As the focus of this thesis, we introduce them separately in the following subsection.

Finally, socially-oriented trust management has many intersections with service-oriented trust management in terms of components [43]. In the meantime, it brings in

a new problem regarding the relationship of trust information in social environments. Generally speaking, socially-oriented trust management is to model and reason about the relationships between members in social environments where the relationships may be influenced by social science, psychology or philosophy [43]. More specifically, within a large-scale social network, the trust path will be extended to involve more members. Therefore, the relationship of trust information, such as trust propagation [48], becomes quite complex, which may be affected by personal and social factors [13]. Moreover, the problems, including trust relationship mining [37] and trustworthy path finding [79] within a large social network, are all challenging issues at the community level.

2.1.3.2 Trust Management Systems at Service Level

The trust management systems at service level include three fundamental parts: trust information collection, trust value assessment and trust value dissemination [92, 50, 124]. First, we discuss trust information collection from the following three aspects: information sources, information expressions and information acquisitions.

- **Information sources:** The trust information mainly comes from two sources: direct interactions (first-hand) and indirect interactions or inference (second-hand). There is no doubt that first-hand, direct interactions between the trustor and trustee provide the most relevant and reliable information. However, in most cases, it is difficult to obtain first-hand information, and so second-hand, or indirect interactions, e.g., "*word-of-mouth*", or inference from first-hand interactions is also essential for trust establishment. For trust management system, however, it can be complex when using second-hand information since it is less reliable. In fact, the computed trust values based on second-hand information are the reputation score of trustees.

- **Information expressions:** In the literature, the computational expressions of trust information are usually via the form of user ratings. Normally, the user-

s of a trust management system need to convert qualitative information into quantitative information. For example, the ratings at eBay are in the set of $\{-1, 0, 1\}$; at Amazon, each rating is an integer in $\{1, 2, 3, 4, 5\}$; EigenTrust [64] and PeerTrust [159] use binary ratings, such as $\{-1, 1\}$ or $\{0, 1\}$. The corresponding ratings measure the level of satisfaction with a past interaction between the user and provider. Then, trust management systems will collect all ratings given by users. Finally, a well-designed trust evaluation model is adopted to deal with the collected ratings in order to compute the trust value of a provider. That trust value can be either binary, discrete or continuous [50]. In addition, apart from ratings, at eBay, the buyers are allowed to leave detailed comments. The comments can be a word or a sentence, which facilitate other buyers to judge the trustworthiness of a seller.

Other trust information expressions include human verbal statements (Very Trustworthy, Trustworthy, Untrustworthy, Very Untrustworthy) and linguistic variables (fuzzy values) [28]. In [126], information entropy is adopted to reflet the level of trustworthiness.

- **Information acquisitions:** Generally speaking, information acquisitions include centralised mechanism and decentralised mechanism. For a centralised mechanism, there is a trusted third party to take all the responsibilities of managing trust information, such as ratings, for the all the members. This mechanism is particularly common for web services [85, 151].

  By contrast, for a decentralised mechanism, there is no trusted third party, and the members in trust management systems have to share the responsibilities of managing trust information, e.g. the peer-to-peer networks [131, 92]. Obviously, it is more complex than centralised mechanism.

Secondly, we discuss trust value assessment and trust value dissemination. For trust value assessment, various trust evaluation approaches have been proposed, and it is an important part for the trust management systems. The existing trust evaluation

approaches can be classified from different perspectives. For example, from the point of view of applications, they are categorised into trust evaluation models applied in network and trust evaluation models applied in internet [124]. We close the review of trust evaluation approaches here, and more details will be given in the next section.

In addition, compared with socially-oriented trust, trust value dissemination for trust management systems at service level is relatively simple. More specifically, for a centralised mechanism, the computed trust values are stored and disseminated by a trusted third party. For instance, eBay uses a centralised trust management system, which posts the computed reputation information of each seller. Meanwhile, the potential buyers can freely access the reputation information of a seller. For a decentralized mechanism, each member in trust management system is responsible for disseminating part of the trust or reputation information. The corresponding responsibility may be symmetrical, e.g., distributed hash tables (DHTs) [129], or asymmetrical, e.g., power-law peer-to-peer network [32]. In [50], Hoffman et al. further present a brief review on the dissemination approaches used in decentralised mechanism. Furthermore, reputation-based trust management systems now take advantage of social networks to extend the range of dissemination. For instance, at eBay, a trustworthy seller can be recommended via Facebook or Twitter, and, consequently, friends and relations, as well as friends' friends, will know the seller.

2.2 Taxonomy of Trust Evaluation

In this section, we will give a detailed review on the existing trust evaluation approaches. In contrast to the works in [131, 121, 119, 61] which introduce typical trust evaluation models, we focus on categorising trust models from different perspectives. In addition, the approaches to be reviewed are not limited to trust evaluation in e-commerce environments, with the scope extended to include other areas, such as peer-to-peer networks, social networks and web services.

2.2.1 Application-Based Taxonomy

In the literature, some works categorise trust evaluation approaches according to their application environments [76, 124]. These are generally subdivided into trust evaluation models applied in network and those applied in Internet. Specifically, network applications include peer-to-peer (P2P) networks [131], multi-agent systems [121], social networks [124] and ad hoc networks [177]. Internet applications include e-commerce [61], web services [155] and cloud computing [100].

2.2.1.1 Network-Based Trust Evaluation

Peer-to-Peer (P2P) Networks: For network-based trust evaluation, peer-to-peer (P2P) networks are one of the major application areas. The typical models include P-Grid, XREP, EigenTrust, PeerTrust and PowerTrust.

The P-Grid [3] model defines a global trust value (measured on a continuous scale from 0 to 1) to determine whether a peer is trustworthy. XREP [27] adopts a binary rating system and provide a distributed polling protocol to evaluate the reputation of each peer. EigenTrust [64] also adopts a binary rating system that computes a global reputation for each peer in a network using an algorithm similar to Google's PageRank [102]. PeerTrust [159] defines some trust metric formulas that aggregate ratings into a general trust value. In PeerTrust, important factors including context factor and community factor, as well as the credibility of feedback are identified, each of which may influence the result of trust evaluation. In PowerTrust, Zhou and Hwang [180] find a power-law distribution in a peer's feedback ratings, and develop a reputation system PowerTrust that dynamically selects a small number of the most reputable power nodes. PowerTrust still focuses on computing a global trust value.

Ad-hoc Networks: Trust evaluation approaches are proposed and applied in ad-hoc networks (both vehicular (VANETs) and mobile (MANETs) networks).

Zhang [177] identifies some characteristics of ad-hoc networks, such as highly dynamic and distributed environments, limited information gathering ability of each peer

and collusion between peers formed easily, which lead to new requirements for designing trust models. The existing trust models in MANETs focus on how to model the trustworthiness of nodes (e.g., collecting trust information about them from other nodes in the network [81]) and how to deliver reliable (security and privacy) packets [23, 160, 22, 181]. In VANETs, trust management is not limited to reliable package delivery, i.e. building a security infrastructure [35, 45], but is also concerned with detecting false information provided by malicious peers [38].

Multi-Agent Systems: Trust evaluation approaches are also widely discussed in the fields of multi-agent systems.

Marsh [91] proposes a computational trust model in multi-agent systems, which is acknowledged as the earliest work on trust evaluation in computer science. In Marsh's model, the trust properties, such as non-transitive and propagative, are discussed. In multi-agent systems, two popular mathematical models for managing trust information are Bayesian systems [97, 60] and the subjective belief model [56]. In addition, Griffiths [42] provides a Multi-Dimensional Trust (MDT) model which allows agents to model the trust value of others according to their personal preference. In the REGRET system [120], an agent-based social-wide trust evaluation method has been proposed. In particular, it takes into account trust development between groups, i.e. the "social dimension". For example, when calculating the trust from agent A to agent B, we need to consider what the other members of A's group think about the agent B and B's group.

Social Networks: With the rapid development of social network research in recent years, the trust properties, such as non-transitive and propagative, are still hotspot issues. For example, in social networks, Golbeck et al. [37] propose trust propagation algorithms based on binary ratings. Guha et al. [44] develop a framework for both trust and distrust propagation within social networks. Hung et al. [48] propose an algebraic approach to trust propagation which includes a concatenation operator for trust aggregation between neighbours, an aggregation operator for combining evidence, and a selection operator for multiple paths selection. Recently, Sherchan et

al. [124] present a comprehensive review on the trust evaluation models in social networks and categorised them into network structure models, interaction-based models and hybrid models.

2.2.1.2 Internet-Based Trust Evaluation

E-Commerce: An e-commerce environment is like a virtual world that combines individuals [95] and society [69], economics [14] and culture [139].

In e-commerce environments, the issue of trust has received much attention from researchers. For instance, Grandison and Sloman [40] explain the importance of trust and present a survey on trust management systems in e-commerce environment. However, most trust models mentioned in that article are used for security-oriented trust. As described in Section 2.1.3.1, the earlier trust management systems are built on lower infrastructure level. However, due to the popularity of e-commerce websites, such as eBay and Amazon, there has been an increase in research studying reputation-based trust evaluation approaches [164, 97, 158, 150]. In [164], the Sporas and Histos systems are introduced. The Sporas system takes into account the temporal characteristics of trust, and the ratings of later transactions are given higher weights. Unlike Sporas, the Histos system adds direct experience, and the reputation value in Histos is subjective property that is assigned by each individual. In P2P e-commerce environments, Xiong and Liu [158] identify some factors, such as context factor and community factor, which affect a forthcoming transaction between buyer and seller. In addition, attacks and defence mechanisms [58] and the credibility of ratings [158] are all trust related topics in e-commerce environments.

Web Services: With the development of websites and applications, trust models are studied in a wider range of context, such as web services.

Basically, the quality of service (QoS) is the focus of current web service techniques, thus trust has been applied for high quality service selection. In [93, 88], the basic structures are built, where service providers can advertise their services and consumers can express their preferences and provide ratings. Vu et al. [145] propose a

trust model for QoS-based service selection, trust information is obtained by comparing the advertised service quality and the delivered service quality. Liu et al. [85] propose an algorithm for the purpose of combining different QoS metrics to get a fair general rating. The RATEWeb model [88] aggregates consumers' ratings which aims to facilitate the trust-oriented service provider selection. In [151], Wang et al. provide some trust evaluation metrics and a formula for trust computation, with which a final trust value is computed. Additionally, they propose a fuzzy logic based approach for determining reputation ranks that particularly differentiate the service periods of new and old service providers.

Similarly, web service takes advantage of policies for security-oriented trust as well. For example, Vimercati et al. [144] provide a solution for secure data access through web services by using an approach of credential-based access control and trust management. In addition, the credibility of users' ratings is also discussed in web service environments [88, 146]. An important survey has been provided by Jøsang et al. [61] in which they review a number of typical trust and reputation evaluation models used in online services.

Cloud Computing: As a new application of web services, cloud computing develops very quickly. Meanwhile, trust management in cloud service is becoming a challenging issue [12]. According to the classification given by Noor et al. [100], trust evaluation in cloud environments is based on four ways: policy, recommendation, reputation and prediction. For example, Hwang and Li [52] propose a security-aware cloud architecture, which uses policies to evaluate the credibility of cloud services. Based on feedback, Habib et al. [46] aggregate the reputation of a particular cloud services.

2.2.2 Technique-Based Taxonomy

Like the taxonomy proposed in [28, 124], we further attempt to categorise trust evaluation approaches according to the techniques that are adopted for trust establishment.

2.2.2.1 Heuristic-Based Approaches

The heuristic-based trust evaluation approaches have been extensively discussed and applied in trust management systems. As pointed by Sherchan et al. [124], these approaches aim to define a practical model that is easy to understand and construct. Meanwhile, they are robust to resist some attacks from malicious members. Therefore, in practice, heuristic-based trust evaluation approaches are suitable for the systems with a large number of users.

From the computational point of view, one type of the heuristic-based approaches is to aggregate and average quantitative feedback ratings. For example, the models in [164, 159, 42, 28, 151, 146] calculate the summation or weighted average of ratings. At eBay, the ratings given to a seller are accumulated over a recent period and a single positive feedback rate is calculated as an indication of the seller's trustworthiness or reputation score. Xiong and Liu [159] propose a PeerTrust model which aggregate ratings to measure the trust value of a seller. Wang et al. [146] propose a RLM model, taking into account malicious ratings before aggregation. The works in [28, 151] propose new aggregation methods taking advantage of fuzzy models, where membership functions are used to determine the trustworthiness of targets.

Additionally, the concept of "flow models" is proposed in [28, 61, 147], and "flow models" are widely used in network environments where a large number of members are involved. Essentially, they still belong to heuristic-based trust evaluation, which compute the trust of a target through some intermediate participants and the trust dependency between them. The typical ones are Google's PageRank [102] and EigenTrust [64]. The basic idea of PageRank [102] is to rank a web page according to how many other pages are pointing to it. To be precise, all the web pages initially have the same rank. The rank of a web page is divided evenly among its forward links, and then it will be recalculated based on its back links. Similarly, within P2P networks, EigenTrust [64] computes an agent trust value via multiple iterations along the trust chain until the trust values for all the agents become stable. Likewise, in social networks,

flow-based techniques are also used for trust management [44, 37, 48].

2.2.2.2 Information Theory-Based Approaches

In Section 2.1.1.3, we explained that, from perspective of economics, trust is related to two major issues: information asymmetry and reciprocity (cooperation).

As illustrated before, trust will mitigate information asymmetry [8]. In other words, the differences in information measured as information entropy, between trustors and trustees during interactions, can be used to measure the level of trustworthiness. Therefore, some researchers propose an information-based trust evaluation model [127, 126]. For example, in online trading, there is a gap between the committed information, such as product quality, and buyers' actual observations. From the point of view of information theory, Sierra and Debenham propose a set of formulas to define commitment and enactment (observation) as well as the concepts like reliability and reputation [126]. Similarly, in social networks, Adali et al. [4] use entropy to measure "balance in the conversation" between two users, and they define their model as behaviour-based trust model.

2.2.2.3 Statistical Theory and Machine Learning-Based Approaches

The statistical theory and machine learning-based approaches focus on proposing a reasonable mathematical model for managing or inferring trust information. However, these approaches have highly computational complexity, making them difficult to be applied to the environments with millions of users [124].

Typically, Bayesian systems [97, 60, 138] and subjective belief model [55, 56, 163, 152] are two major examples based on statistical theory. The former takes binary ratings as input and computes reputation scores by statistically updating beta probability density functions (PDF), while the latter uses subjective probability theory in trust evaluation. On the other hand, machine learning techniques, such as Artificial Neutral Networks (ANNs) and Hidden Markov Models (HMMs), are adopted for trust

evaluation. For example, Ham et al. [47] take advantage of RBF Neural Networks for reputation prediction in mobile ad hoc networks. In [83, 179], HMM is used for trust prediction before transactions in e-commerce environments, and ElSalamouny et al. [34], propose a discrete HMM-based trust evaluation model. In [148, 78], conditional probability model is used to infer the trust values between participants within online social networks.

2.3 Context-Aware Trust Evaluation

According to our review in Section 2.2, most trust evaluation approaches lack consideration of context information. In addition, they often compute one value to reflect a general or global trust status of a trustee [120, 27, 64, 163, 154, 22, 88, 146, 145, 28]. Nevertheless, as indicated in Section 2.1.2, a very important property of trust is context dependence. In recent years, increasingly more researchers turn their attention to the relationship between trust evaluation and contextual information.

This thesis targets context-aware transaction trust evaluation in e-commerce environment, thus we will first give a general review on context-aware trust evaluation from a broader perspective in this section. The review focuses on three aspects: context in different applications, the granularity of trust evaluation and the existing context-aware trust evaluation approaches.

2.3.1 Context in Different Applications

Context has been studied in many applications, which is not exactly the same as those in trust management. As defined by Palmisano et al. [103], context refers to the "conditions or circumstances which affect something". In different application fields, conditions or circumstances change, resulting in various definitions of context.

Context-Aware Pervasive Systems: Context-aware pervasive systems refer to a general class of mobile systems that can sense users physical environment in order to

automatically recognise their demands [115]. Such systems are a component of a ubiquitous computing or pervasive computing environment. In context-aware pervasive systems, context aims to model information from three aspects: where you are, who you are with, and what resources are nearby. For example, it is initially defined as the information about the location of a user, the identities of people near the user, and objects around the user [123]. Additional details, such as the date, season and temperature information, are then added [21].

Recommendation Systems (Recommender Systems): Recommendation systems refers to the systems, such as Douban[3] and MovieLens[4], that recommend products, books or films to potential users. A movie recommendation application, for example, can take into consideration contextual information [5], such as when, where, and with whom a movie is seen, resulting in a more accurate recommendation.

Here one may be confused with trust management systems and recommendation systems, because they are usually combined in practice (e.g., Amazon), and both collect ratings from members or users in a community. However, the two systems have fundamental differences and they are based on opposite assumptions. This difference has been explained by Jøsang et al. [61].

- Typically, for example, a recommendation system in e-commerce environments operates in the *Buyer* $\times$ *Item* space. Recommendations of a product are made to a potential buyer based on ratings from those raters (buyers) who share similar tastes with the potential buyer. This process is called "*Collaborative filtering*" [6]. The discussion of contextual information in these systems focuses on the *Buyer* space. For example, demographic attributes of buyers (e.g. age and income) [6] or the intent of buyers [103] mean different tastes, and all these information could affect the conclusion of a recommendation.

- By contrast, trust management systems operate in the *Seller* $\times$ *Transaction* space. The purpose for buyers to give ratings is to evaluate the performance of a

[3]http://www.douban.com
[4]http://movielens.org/

seller during transactions, so as to identify potential malicious sellers or services with poor quality. This process is called "*Collaborative sanction*" [98]. For instance, at eBay, buyers' ratings are mainly based on transaction quality, rather than significantly depending on subjective opinions compared to recommendation systems. Specifically, buyers may provide different ratings to the same type of *fridge* depending on their different preferences on its functionality or appearance, but they can all judge whether the seller provided a damaged *fridge*. Thus, the ratings biased to majority ratings in recommendation systems mean different tastes of buyers, but would be more likely to be unfair or incredible in trust management systems [88].

Context in Web Search: In web search environments, context is considered as the set of topics potentially related to the search term [103]. For instance, it is used to define a specialised search engine or narrow search fields, such as ResearchIndex (CiteSeer[5]) and DBLP[6], which are both specialised search engines for scientific literature. In addition, by providing user context, such as users' behaviours and interests (an ontological profile), better search results will be obtained that are most relevant to the user [125].

Context in Social Networks: In social networks, Liu et al. [80] define social contextual information including social relationships (e.g. a father and a son), social positions (e.g., a professor at macquarie university), and preferences (e.g., enjoys swimming). Then, by taking advantage of social contextual information, they will find a better quality of the extracted trust networks. Furthermore, within a social network, by combining social contextual information, more accurate recommendation results will be achieved, regardless of item recommendations [87] or people recommendations [148].

2.3.2 The Granularity of Trust Evaluation

Based on the above illustration, context has been applied in various applications for the purpose of describing things, or obtaining results, with more details and accura-

[5]http://citeseerx.ist.psu.edu/
[6]http://www.dblp.org/

cy. Likewise, after introducing contextual information, trust evaluation will be more accurate and comprehensive.

In the literature, some works differentiate reputation-based trust evaluation by granularities and categorise them into the single-context models and the multi-context models [121, 97]. More specifically, the single-context models refer to models that compute a single trust or reputation value, without taking into account the context information. This is the *coarse-granularity* trust evaluation. By contrast, the multi-context models refer to models that have the mechanism to deal with several contexts at a time, while comprising different trust or reputation values associated with them. This is the *fine-granularity* trust evaluation. Compared with single-context models, multi-context trust evaluation models can provide comprehensive and detailed trust information of a target, and so its results are more accurate. But multi-context trust evaluation is much more complex, particularly when considering the combinations of context dimensions. Thus, very limited work has been reported in the literature.

However, one may argue that it is necessary to introduce context information for trust evaluation with high computational complexity. Due to the diversification of a member's performance within certain application environment, the single-context model has its limitations. For example, in web service environments, an agent may be trustworthy in the context of providing high quality travel services (flights and accommodation); but in the context of car sales, may be untrustworthy [155]. Moreover, in e-commerce environments, as depicted by the motivation at the beginning of this thesis (see Section 1.1), *value imbalance* is a typical problem resulting from single-context trust evaluation. Similarly, Liu et al. [84] identify an 'unexpected' phenomenon of reputable sellers in e-commerce called *imprudence*, which refers to the situation where they behave inappropriately (possibly out of complacence to deliver poor products). All of these evidences suggest that trust evaluation with fine granularity (multi-context trust evaluation) is in great demand.

2.3.3 Trust Evaluation with Contextual Information

In this subsection, we review some existing studies considering the relationship between trust evaluation and context information. We broadly categorise them in following three aspects.

2.3.3.1 Multi-faceted Trust Evaluation

In [42], Griffiths proposes a Multi-Dimensional Trust (MDT) model based on Marsh's "general trust" [91]. However, MDT is distinct from both "situational trust" [91] and "general trust", which studies contextual trust from a multiple-faceted perspective. In particular, the trustworthiness of a particular task can be modelled in several dimensions (e.g., timeliness, quality and cost), and a trustor gives a trust value according to its direct experience in each dimension. Note that the trust value is updated after each interaction between a trustor and trustee. Then, a trustor can weigh personal preference of these predefined dimensions when the corresponding trust values are aggregated to compute a single general or global value. Thus, given the same trustee, the trust results computed for different trustors may vary.

Essentially, from the point of view of granularity, MDT belongs to the single-context model. As pointed out by Griffiths [42], MDT is complementary to "situational trust" rather than being a "situational trust" itself. In [155], Wang and Vassileva stress that trust evaluation should be both context dependent and multi-faceted. For instance, a user might evaluate a web service (single-context) from several QoS aspects, such as response time, throughput, and execution time. At eBay, a buyer provides a general rating after each transaction (single-context), which depends on the combination of the buyer's ratings from (a) item as described (i.e. whether the seller provides the quality of goods as s/he described), (b) shipping time (i.e. whether the seller delivers goods on time), (c) communication (i.e. whether the seller has prompt and friendly communication with buyers), and (d) shipping charges (i.e. whether the seller charges a reasonable price for shipment). Therefore, MDT can be termed as multi-faceted trust. Likewise,

in REGRET [120] and RATEweb systems [88], the same multi-dimensional structure is adopted when evaluating a seller or a service provider's reputation. However, as mentioned above, they overlook the changes of context (e.g., product/service category and transaction amount) in historical transactions, so the likelihood of a successful forthcoming transaction can hardly be predicted by them.

2.3.3.2 Similarity-Based Context-Aware Trust Evaluation

The context similarity calculation is regarded as an important means to deal with the context-aware trust evaluation problem. Generally speaking, these trust evaluation approaches will first model or describe context, and then trust value is computed from one context to another based on their context similarity.

Context description based on key-values for calculating similarity: Some context-aware trust models use key-values, such as keywords and task attributes, to model context. Uddin et al. [142] propose a CAT (Context-Aware Trust) model to compare the similarity of contexts by using key values that could describe, to some extent, certain context. 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, as two out of three keywords are different. Caballero et al. [24] define a formula using task key values to calculate the similarity between two tasks in order to evaluate the trust level of different tasks. Rehak et al. [108] propose a trust model to resolve contextual trust, which uses clustering to identify full context space to be several reference contexts based on the attributes of context. The trust evaluation of a new targeted problem is the weighted sum of the trust values in all reference contexts. The weight is based on distance function d; a smaller value of d means higher similarity between certain reference contexts. In [82], Liu and Datta take advantage of contextual trust to enhance the data availability in Peer-to-Peer (P2P) backup storage systems. Specifically, they describe context with different attributes/dimensions such as a peer's time zone, location, IP address, and similar context refers to the peers having the same

value in certain dimension, e.g., the same time zone or the same location.

It is true that key-values are easy to manage, but they lack capabilities for sophisticated structure [130], since contextual information is complex and ambiguous in most cases. For example, a potential buyer may not know whether a seller could sell *fridges* with high quality, but s/he knows this seller has a good reputation of selling *home appliances*. To a larger extent, these two contexts have similarity, conceptually, *fridge* is an instance of *home appliances*, and thus the potential buyer might choose to buy a *fridge* from this seller.

Context description based on ontologies for calculating similarity: Apart from key-values, Strang et al. [130] point out that "ontologies are a promising instrument to specify concepts and interrelations". Therefore, more studies use the ontological structure to analysis context-aware trust. Uschold et al. give an explanation of related conception of ontology [143]. Briefly, ontology is a conceptual model, but could express the relationship between different concepts. The relationship of these concepts can be "a part-of" (part and overall), "a kind-of" (Inheritance), or an "instance of".

Toivonen et al. [140] describe a trust determination process base on contextual information. They propose the ontology structure of a network, and some software components are downloaded via this network. Suppose that people need to download some software components from a new node in this network, the trust value of the node can be calculated from other trustworthy nodes in this ontological structure. The influence of other nodes on trust evaluation depends on their ontology-based intimacy to the new node. Tavakolifard et al. propose an enhanced trust model based on the above model [136]. In their work, a general comparator is defined to find similar and relevant nodes in ontology structure with few computational details. In addition, based on hierarchical structures, some researchers proposed a conception of *trust inherited* to resolve context-aware trust evaluation. Holtmanns et al. conceptually point out that context can often be structured hierarchically [51]. For example, if I trust my brother to drive me to the airport, I can most likely give him my car key, as giving him my car keys can be considered as a subset of the access rights of driving my car. Samek et

al. [122] provide a hierarchical model of trust in contexts (HMTC), which is used to find inclusion relations between contexts.

However, similarity-based context-aware trust evaluation models still do not belong to multi-context trust evaluation [121]. As pointed out in [97, 81, 82], the context similarity is applied to infer the trustworthiness of a target in a certain context where there is no or not enough trust or reputation information (e.g., ratings) from the same context. Therefore, these trust models focus on calculating a single trust value under the corresponding specific context. Note that the single trust value can also be general or global, associated with all the related contexts [81, 108, 24]. Furthermore, similarity-based context-aware trust evaluation often uses heuristic-based techniques for trust evaluation.

2.3.3.3 Machine Learning-Based Context-Aware Trust Evaluation

Rettinger et al. [111] propose a context-sensitive trust evaluation model (IHRTM) taking advantage of statistical relational learning. In the IHRTM model, contextual information is discussed in the $Seller \times Item$ space. According to the learning results, all 47 selected sellers in their experiments are assigned to 4 groups based on the context attributes in the $Seller$ space, such as feedback score and positive feedback rate, and the 630 items sold by these sellers are assigned to 40 clusters based on the context attributes in the $Item$ space, such as product category and product condition (new or used). Finally, a 4×40 matrix is formed to indicate the trustworthiness of 4 seller clusters under 40 item clusters. Note that the IHRTM model belongs to multi-context trust evaluation. In order to improve the accuracy of predicting the trustworthiness of a forthcoming transaction, Liu and Datta [83] extract useful features from transaction context, such as product category and price, as observations to construct a Hidden Markov Model (HMM) for modelling the dynamic trust of a seller. In addition, to reduce computational complexity, information theories and Multiple Discriminant Analysis (MDA) are adopted in their model for feature space reduction.

Actually, a major disadvantage of all the machine learning-based trust evaluation

approaches is their high computational complexity, which makes them difficult to be applied to the environments with millions of users [124]. For example, when having a large number of sellers, there will be many clusters of sellers for IHRTM, leading to a high complexity in learning iterations. As new transactions happen every day, the cost of re-learning, which takes new transactions into account, is higher. Moreover, from the transaction context perspective, IHRTM does not support the analysis of reputation on the product categories along a path in the product category hierarchy (e.g. "*Apple iPhone*" and "*Smartphone*" in sequence). This analysis is particularly necessary when a new product or a product in a new category is just released. Also, it does not support the trust evaluation of a seller in the transactions in a given price range. This need comes from a buyer when s/he is concerned about the risk of monetary loss in a forthcoming transaction [159, 132].

By contrast, our proposed *ReputationPro* model in this thesis is typically a heuristic-based [124] multi-context [121] trust evaluation model. There are two important reasons leading to its outperformance over the existing context-aware trust evaluation models:

- *ReputationPro* is a multi-context model which has the mechanisms to deal with several contexts at a time comprising different trust or reputation values associated with them.

  Compared with single-context trust evaluation and similarity-based context-aware trust evaluation, multi-context trust evaluation can reflect a seller's dynamic trustworthiness in various transaction contexts, which provides comprehensive and detailed trust information of a seller. As multi-context trust evaluation is much more complex particularly when considering the combinations of context dimensions, very limited work reported in the literature. Thus, our proposed *ReputationPro* model makes great sense.

- *ReputationPro* is an efficient heuristic-based multi-context model that can be directly applied in large-scale e-commerce applications.

Like PeerTrust [159] and RATEWeb [88] trust evaluation models, *Reputation-Pro* adopts heuristic-based technique to aggregate and average trust ratings as the trustworthiness or reputation values of a seller. Compared with IHRTM model [111], which is the only multi-context model reported in the literature and adopts statistical and machine learning-based techniques, *ReputationPro* is much more efficient and thus more suitable to be applied to the dynamic environments of e-commerce applications with millions of users and transactions that are updated every day.

In Fig. 2.2, the *ReputationPro* trust evaluation model is compared with some existing trust evaluation approaches so as to highlight its characteristics and the contributions of our work from the perspective of trust evaluation.

2.4 Related Techniques in Data Warehousing

This section reviews the related techniques in data warehousing, which will be improved and adopted to resolve context-aware trust evaluation (multi-context trust model) in e-commerce environments. To the best of our knowledge, the work introduced in this thesis is the first one in the literature that computes the reputation profile of a seller taking advantage of related techniques in data warehousing.

2.4.1 OLAP (On-Line Analytical Processing) and Data warehouses

In the broader research literature, our targeted context-aware trust evaluation problem is somewhat similar to sales analysis from multiple perspectives in data warehousing and business intelligence. Typically, the sales data warehouse for a company contains three dimensions Product category, Location and Time. The OLAP operations refer to the queries on the aggregation of sales over each dimension or their combinations, such as the sum of sales per product category or the sum of sales per product category

Approaches	Application Fields	Logic Hierarchy	Technology Adoption	Contextual Information
Marsh [91]	MAS/SN	SerTE, SocTE	HET	CSI
PolicyMarker- Blaze et al. [20]	EC	SecTE	TST	SC
EigenTrust- Kamvar et al. [64]	P2P	SerTE	HET	SC
Jøsang [56]	MAS/SN	SerTE	ST/ML	SC
PeerTrust- Xiong and Liu [159]	P2P, EC	SerTE	HET	CSI
Vimercati et al. [144]	WS	SecTE	TST	SC
RATEWEB- Malik and Bouguettaya [88]	WS, EC	SerTE	HET	SC
Golbeck and Hendler [37]	MAS/SN	SocTE	HET	SC
Wang et al. [148]	MAS/SN	SocTE	ST/ML	CSI
Liu and Issarny [81]	AHN	SerTE	HET	CSI
Hwang and Li [52]	CC	SecTE	TST	SC
CONFIDANT- Buchegger and Boudec [23]	AHN	SecTE	TST	SC
TRSIM- Caballero et al. [24]	P2P	SerTE	HET	CSI
XRep- Damiani et al. [27]	P2P	SerTE	HET	SC
MDT- Griffiths [42]	MAS/SN	SerTE	HET	SC
Ham et al. [47]	AHN	SerTE	ST/ML	SC
Liu and Datta [83]	EC	SerTE	ST/ML	CSI
REGRET- Sabater and Sierra [129]	MAS/SN	SerTE, SocTE	HET	SC
Sierra and Debenham [120]	MAS/SN, EC	SerTE	IT	CSI
Wang and Singh [152]	MAS/SN	SerTE	ST/ML	SC
Yu and Singh [162]	MAS/SN	SerTE, SocTE	HET	SC
Wang et al. [151]	WS, EC	SerTE	HET	SC
IHRTM- Rettinger et al. [111]	EC	SerTE	ST/ML	MC
RepuationPro	EC	SerTE	HET	MC

ApplicationFields	Logic Hierarchy	Technology Adoption	ContextualInformation
P2P: **P**eer to **P**eer networks MAS/SN: **M**ulti-**A**gent **S**ystems and **S**ocial **N**etworks AHN: **A**d-**h**oc **N**etworks EC: **E**-**C**ommerce WS: **W**eb **S**ervices CC: **C**loud **C**omputing	SecTE: **Sec**urity-oriented **T**rust **E**valuation SerTE: **Ser**vice-oriented **T**rust **E**valuation SocTE: **Soc**ially-oriented **T**rust **E**valuation	TST: **T**raditional **S**ecurity **T**echniques HET:**He**uristic-Based **T**echniques ST/ML: **St**atistical **T**heory and **M**achine **L**earning-based Techniques IT: **I**nformation **T**heory-Based Techniques	SC:**S**ingle-**C**ontext model (without taking into account contextInformation or multi-faceted trustevaluation) CSI: **C**ontext information or context **S**imilarity as **I**mpact factor (no in-depth discussions on the impact of context and still not a multi-context trust evaluation) MC: **M**ulti-**C**ontext model (computing several contexts at a time and maintaining different trust or reputation values associated to them)

Figure 2.2: The comparison of existing trust evaluation approaches

and per month combination. Gray et al. [41] point out that there are $O(2^n)$ possible aggregations for a data warehouse with n dimensions composing a "data cube".

In order to accelerate query processing, some results can be pre-computed and stored as *materialised views* [49, 77]. However, these approaches only benefit the queries on dimensions with predefined hierarchies. In particular, the *Time* dimension in sales analysis refers to static calender months, e.g., January, February, etc. By contrast, as will be discussed in relation to transaction context dimensions in Section 3.1, while the product category hierarchy is predefined and static, *Price* and *Transaction Time* are dynamic dimensions. Specifically, the dynamicity of the *Price* dimension refers to the reality that the price of a product may change over time. Even on a given day, multiple transactions selling the same product may have different prices. The dynamicity of the *Transaction Time* dimension refers to new transactions are to be added to the database continuously over time, modifying the set of "most recent transactions". In addition to *materialised views*, some other works improve the performance of queries in data based on specifically designed column-oriented database systems [1]. Different from these works, our proposed approaches in this thesis are based on popular relational database management systems that are being widely used by e-commerce websites, so that the designed models can be directly applicable.

As outlined in the analysis given in Section 4.1, the research on point aggregate problem based on spatial or spatio-temporal data warehouses is relatively close to our targeted context-aware trust evaluation problem. In the literature, there are many related approaches to point aggregation. More specifically, in order to accelerate query processing, they still pre-compute some results, and then appropriately store the results in specialised indexes forming various tree structures [105, 135, 134, 168]. We review them separately in the following subsections. Moreover, since memory-based approaches [70, 96] are inappropriate for large-scale data processing, we will restrict our review to some well-known disk-based approaches.

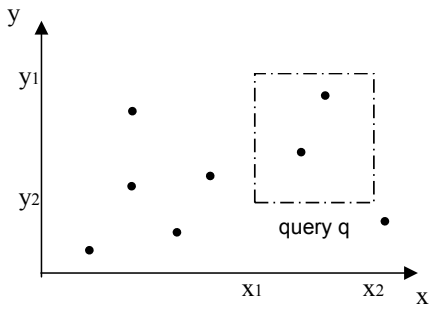


Figure 2.3: An example of an RA query

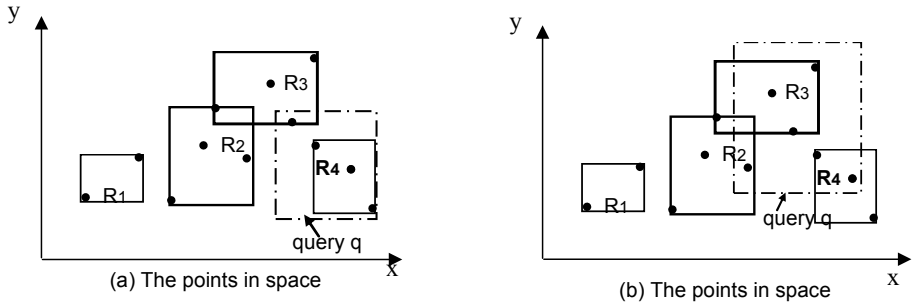


Figure 2.4: An aR-tree

2.4.2 Range Aggregate (RA)

Fig. 2.3 shows the traditional RA query [104] in a two-dimensional space which is in regards to computing the total number of points falling into a query region q surrounded by $[x_1, x_2]$ and $[y_1, y_2]$, e.g., answering a query in traffic supervision systems, such as: "What is the total number of cars inside a certain district?". Usually, a query region can be any area within the two-dimensional space. Since our targeted multi-context trust evaluation is modelled as an extend RA problem (see Section 4.1) in two-dimensional space, we review the approaches to two-dimensional RA problem in more detail.

2.4.2.1 The aR-tree

The *aR-tree* [63, 104] maintains the x-y coordinates for each minimum bounding rectangle (MBR) (e.g., R_1, R_2, R_3, R_4 in Fig. 2.4(a) are all MBRs). In the meantime, each MBR records the total number as an aggregate of the objects that fall into an MBR. To

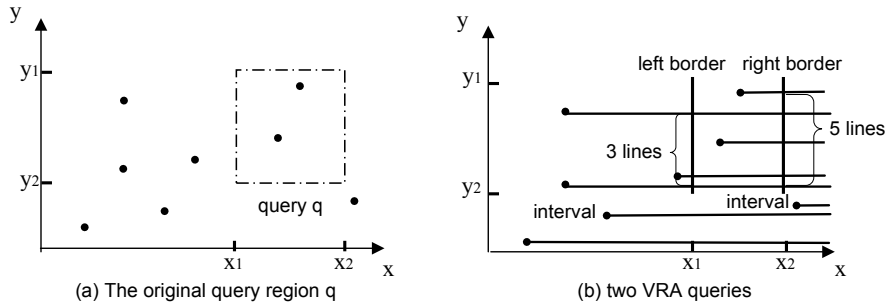


Figure 2.5: An RA query transformed to two Vertical Range Aggregate (VRA) queries

compute the number of objects in a query region q, in Fig. 2.4(a), the MBR R_4 within q will not be accessed. Rather, R_4's pre-computed aggregate (i.e. 3) is directly used. But R_3 needs to be visited, as it partially overlaps with q. The total number of points in q equals their sum $3 + 1 = 4$. A serious problem of the *aR-tree* is that its performance significantly degrades when answering a large query region, since in such a case there are more MBRs overlapping with the query region (see Fig. 2.4(b)).

2.4.2.2 The aP-tree

Tao et al. [135] propose the *aP-tree* to improve the *aR-tree*, based on the following transformation on a query region q. They first convert each spatial point to an *interval* (i.e. a horizontal line) (see Fig. 2.5(b)). When q is surrounded by $[x_1, x_2]$ and $[y_1, y_2]$ is transformed to two *borders* (i.e. two vertical lines): $x_1 : [y_1, y_2]$ and $x_2 : [y_1, y_2]$, an RA query is converted to retrieving the number of intervals that intersects the two borders. For instance, in Fig. 2.5, the number of intervals intersecting the left border $x_1 : [y_1, y_2]$ is 3 while the number of intervals intersecting the right border $x_2 : [y_1, y_2]$ is 5. The total number of points in q equals their difference $5 - 3 = 2$. Tao et al. [135] define the number of intervals intersecting a border as a Vertical Range Aggregate (VRA). In order to compute each VRA value, the *aP-tree* is then proposed, extending the original *multiversion B-tree (MVBT)* [17], that contains an additional field *agg* in each record to store the aggregation result.

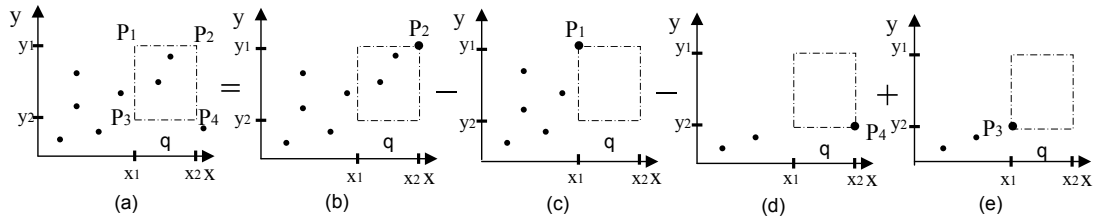


Figure 2.6: An RA query transformed to four dominance-sum queries

2.4.2.3 The MVSB-tree and The BA-tree

Zhang et al. [168] address the RA problem in a two-dimensional space by converting an RA query to four dominance-sum queries. Given two two-dimensional points $x = (x_1, x_2)$ and $y = (y_1, y_2)$, x dominates y if $x_1 \geq y_1$ and $x_2 \geq y_2$. The corresponding dominance-sum of the point P is the aggregation of all the points that are dominated by P. Therefore, in Fig. 2.6(a), the total number of points in the query region $P_1P_2P_3P_4$ equals $7 - 5 - 2 + 2 = 2$, namely, the dominance-sum of the point P_2 (see Fig. 2.6(b)) subtracts the dominance-sum of the point P_1 (see Fig. 2.6(c)) and the dominance-sum of the point P_4 (see Fig. 2.6(d)). As the dominance-sum of the point P_3 (see Fig. 2.6(e)) has been subtracted twice, their sum must be added again. In order to compute each dominance-sum query, Zhang et al. further propose the *MVSB-tree* [166, 168] and the *BA-tree* [167], respectively.

The *MVSB-tree* results from augmenting the *SB-tree* [161]. It logically divides the two-dimensional space into multiple nonintersecting rectangles. When inserting an object with the coordinate (x_i, y_i), the aggregation operations perform in all the rectangles within the area $[x_i, max_x) \times [y_i, max_y)$. Here max_x and max_y collectively form the upper-right corner of the complete space.

The *BA-tree* is another index scheme for answering RA queries that extends the *K-D-B-tree* [114]. Fig. 2.7 depicts a general structure of *BA-tree*, as in the *K-D-B-tree*, each node corresponds to a rectangular space, such as the area A. The node at a higher level corresponds to a larger rectangular space formed by several adjacent areas, such as the area formed by A, B and C. The root node corresponds to the complete space.

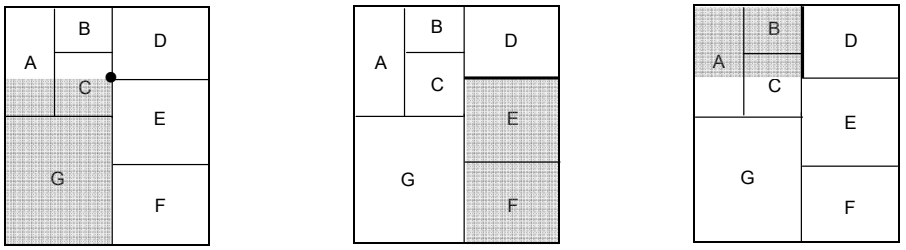


Figure 2.7: The structure of *BA-tree*

The augmentation of the *BA-tree* over the *K-D-B-tree* is that each node (e.g., the area D) also stores three aspects of information: the subtotal of points to the lower left of D (see Fig. 2.7(a)), the x-coordinates of the points below D (see Fig. 2.7(b)) and the y-coordinates of the points to the left of D (see Fig. 2.7(c)). The *BA-tree* achieves linear performance when answering each dominance-sum query.

Apart from the above reviewed approaches, the *CRB-tree* has been proposed for solving RA problem [39, 7]. The general structure of a *CRB-tree* contains two parts: 1) one normal B^+*-tree* [15] constructed on the y-coordinates of points in two-dimensional space; and 2) another B^+*-tree* constructed on the x-coordinates of points, but with each internal node storing weights as a secondary structure. The *CRB-tree* has good performance for answering RA queries as well as further compresses the space consumption. However, as pointed out in [135, 168], it is based on a stringent assumption that it runs on top of *bit-wise* machines. Specifically, the integers in secondary structure are stored by bits. Namely, an integer with value v is represented by exactly $\log_2 v$ bits so that multiple integers may be compressed into a single word. By contrast, a typical *word-wise* machine model uses four bytes to store a single integer. Therefore, as pointed out in [135, 168], the *CRB-tree* "is mainly of theoretical interest", and does not apply to the prevalent commercial word-wise computers. Note that our proposed approaches in the thesis are all based on common word-wise machine models and thus can directly apply to commercial servers.

2.5 Summary

This chapter provides a general overview of the research studies on trust, trust management systems, trust evaluation, context-aware trust evaluation and related techniques in data warehousing. Conceptually, we first present the definition of trust in different disciplines as well as its properties. Second, we introduce three logic levels of trust management systems. Third, the typical trust evaluation methods are categorised and reviewed based on their application environments and adopted technologies. Fourth, we present a comprehensive review on context-aware trust evaluation approaches and highlighted the contributions of this thesis. Finally, we introduce some related techniques in data warehousing to be improved and adopted for solving our targeted context-aware trust evaluation problem.

Chapter 3

A Trust Vector Approach to Context-Aware Transaction Trust Evaluation

The trustworthiness of sellers is an essential issue for buyers before placing an order. With simple trust management mechanisms at some e-commerce websites, such as e-Bay, a single trust value of a seller is computed based on the ratings of past transactions given by buyers. Likewise, in the literature, single value (e.g. a value in the range of [0,1]) based trust models [120, 27, 64, 163, 154, 22, 88, 146] still play an important role in measuring the trustworthiness of trustees. Such a single value, however, can only reflect the general or global trustworthiness, without any contextual transaction information taken into account. With such a result as the indication of reputation, a buyer may be easily deceived by a malicious seller in a transaction where the notorious *value imbalance* problem [66, 29, 58, 67, 57] is involved, i.e. a malicious seller accumulates a high level reputation by selling cheap products then deceives buyers by inducing them to purchase more expensive products. In addition, the single value computed by these trust evaluation models is based on past transactions, and it is static with regard to any forthcoming transaction of selling different products. As a result, they can hardly predict the likelihood of a successful forthcoming transaction [150, 149, 170].

Thus, a good trust management system in e-commerce environments should be transaction context-aware. In the meantime, the computed trust values need to be

based on both past transactions and the forthcoming transaction. Therefore, in contrast to most existing trust evaluation models that compute a single trust value, we propose a trust vector approach to context-aware transaction trust evaluation.

This chapter is organised as follows. In Section 3.1, we distinguish the definitions of context in different application environments, and focus on discussing and defining transaction context in e-commence environments. In particular, we model transaction context with several dimensions, and first propose the concept of the transaction context imbalance problem and several types of it in e-commerce environments. In situations where there are no, or not enough, ratings from the transactions with the same context as the forthcoming transaction, we propose a set of methods to calculate transaction context similarity in Section 3.2, the results of which are used in inferring the trust level of a forthcoming transaction. Our solution has potentially wide applications in e-commerce environments. Section 3.3 introduces the trust vector approach to outline a seller's reputation profile. In Section 3.4, we present empirical studies on the proposed trust vector. Section 3.5 summarises our work in this chapter.

3.1 Transaction Context

This section first discusses the meaning of transaction context, differentiates some related concepts, and then models transaction context for evaluating the trustworthiness of sellers. Finally, the concept of transaction context imbalance problem in e-commerce environments is proposed.

3.1.1 What is Transaction Context?

As defined by Palmisano et al. [103], context refers to "conditions or circumstances which affect something". But different application fields have many definitions of context. For instance, in Section 2.3.1, we have illustrated that, in context-aware pervasive systems, contextual information is defined as the information about the location of a

user, the identities of people near the user, objects around the user, the date, season and temperature information [123, 21].

Differently, in recommendation (recommender) systems, which recommend products to potential buyers, the discussion of contextual information focuses on the *Buyer* space. For example, demographic attributes of buyers (e.g. age and income) [6] or the intent of buyers [103] mean different tastes, and all these information could affect the conclusion of a recommendation. By contrast, trust management systems operate in the *Seller* $\times$ *Transaction*$(Item)$ space [61]. As sellers' information is relatively simple and static, in this thesis, the discussion of context focuses on the transaction space w.r.t transaction trust evaluation.

Definition 1: *transaction context* refers to *the factors that can determine, imply or affect the trustworthiness of a forthcoming transaction*.

3.1.2 Transaction Context Modeling

Based on Definition 1, our work considers the factors with influence on the trustworthiness of a forthcoming transaction.

3.1.2.1 Transaction Context Dimensions

• **Product Category (a static but hierarchical dimension):** Product Category refers to the category of item traded in a transaction. The product category determines the nature of the transaction which greatly influences transaction trust [158, 159].

The category of transaction items has a hierarchical structure. There are some *Products and Services Categorization Standards (PSCS)* that aim at constructing the product category hierarchy, such as UNSPSC[1] and eCl@ss[2], each of which groups similar products and provide an industry-neutral hierarchical structure of product categories with up to four layers. eBay has a different product category[3] schema with

[1] http://www.unspsc.org/
[2] http://www.eclass.de/
[3] http://pages.ebay.com/sellerinformation/ebaycatalog/categories.html

simply two layers, and it groups products by considering factors such as marketing and common use.

• **Transaction Amount (a dynamic linear dimension):** Transaction amount refers to the sum of prices of all products in a transaction. A transaction of about US$\$10$ is different in nature to the one of about US$\$10K$. The larger the transaction amount, the more likely a fraud will occur, since the benefits of cheating are greater [14]. For the sake of simplicity, like eBay, *each item in a transaction is considered separately* in our work. Hence, the transaction amount equals the price of the item in a transaction. In this thesis, we use "transaction amount" and "price" interchangeably.

The Price dimension is dynamic as the price of a product may vary from time to time. Owing to product condition (new and used) and product value changes over time, the prices of transactions selling the same product may be different.

• **Transaction Time (a dynamic linear dimension):** Transaction time is the time when a transaction happens. As mentioned in Section 2.1.2, transaction trust evaluation is time-sensitive, because transaction quality may change with time [128].

The Transaction Time dimension is also dynamic because the time point "*now*" changes every day, and new transactions added to the database over time change the set of "most recent transactions".

3.1.2.2 Transaction is Multi-Faceted

In [155], Wang and Vassileva stress that trust evaluation should be both context dependent and multi-faceted. Even in the same context, there is a need to develop differentiated trust in different aspects. For instance, in the process of transactions, the quality of services affects the transaction trustworthiness as well. With regard to the quality of these services, buyers can provide corresponding ratings in some e-commerce websites. At eBay, buyers' ratings also evaluate (a) item as described (i.e. whether the seller provides the quality of goods as s/he described), (b) shipping time (i.e. whether the seller delivers goods on time), (c) communication (i.e. whether the seller has prompt and friendly communication with buyers), and (d) shipping charges (i.e. whether the

seller charges a reasonable price for shipment).

Imagine that both sellers, S_a and S_b, sell the same high quality product. In past transactions, S_a always provided good quality services, such as delivering goods on time and having prompt and friendly communication with buyers, while S_b delayed delivery now and then and had poor communication with buyers. Surely, in such situations, seller S_a is more trustworthy and preferable.

3.1.3 Transaction Context Imbalance

As stated before, malicious sellers and fraudulent transactions could take advantage of transaction trust result without considering any context. Consequently, it may lead to some *transaction context imbalance* problems, which is first proposed in this thesis and can include the following types.

- **Transaction Amount Imbalance:** There are two different cases in the *transaction amount imbalance*.

  (a) A seller accumulates a high level of trust by offering cheap and attractive products, and then s/he may deceive buyers with expensive products. In the literature, this issue is also termed as *value imbalance* [66, 29, 58, 67, 57].

  (b) Buyers usually believe that if a seller has successfully finished many transactions selling expensive products, s/he may not cheat in forthcoming transactions selling cheaper products. In fact, such a "reputable" seller may not be as prudent as in expensive transactions to serve each buyer well due to limited profit.

- **Product Category Imbalance:** A seller has accumulated a high trust level by selling certain products, and then s/he can utilise this high trust value to sell products in different categories for more profit. According to the suggestion from Alibaba (see Section 1.1.2), such a seller should have different levels

of trustworthiness with respect to different products or different product categories [159]. For instance, a seller, who sold *watches* before and now starts to sell a certain type of *notebook computers*, should not have the same level of trust as before due to the lack of sufficient experience and reputation in selling the new products with a completely different nature.

The focus of our work is to identify and prevent potentially malicious transactions with respect to different types of *transaction context imbalance* problems. Based on the modelled transaction context, we will first propose methods for transaction context similarity comparison in Section 3.2. According to the comparison result, we will propose a trust vector consisting of a set of trust values to reflect the trust level of a seller from different transaction context in Section 3.3. In particular, this trust vector is also bound to a forthcoming transaction, rather than past transactions only.

3.2 Similarity Comparison Between Transaction Context Dimensions

In order to obtain the trustworthiness of a seller in a specific transaction context, a buyer needs to take other buyers' ratings on this seller in the same transaction context into account. But when there are no, or not enough, ratings for the same transaction context, it is a good practice to derive the trustworthiness of the seller from all the ratings in any related transaction context. In this case, the similarity of the context in the forthcoming transaction, and the different context in a past transaction, should be compared to weight the rating for the past transaction [97]. According to our modelled transaction context, we propose methods for computing transaction context similarity.

3.2.1 Similarity Comparison of Product Category

Next, let us consider the dimension of Product Category, for example, a buyer plans to buy a *'Cannon EOS 6D SLR (single-lens reflex) Digital Camera'* from a seller.

The related items that s/he could be also concerned about the trustworthiness of this seller in selling various *'Cannon SLR digital cameras'* or *'SLR Digital Camera'*. In subsection 3.2.1.1, we propose a hierarchical structure of product category. With this structure, all the related products can be located by comparing their similarity, and the method for similarity measurement is given in subsection 3.2.1.2.

3.2.1.1 Hierarchical Structure of Product Category

As stated before, there are three existing popular product classification schemas which aim at grouping similar products, UNSPSC, eCl@ss and eBay. They all adopt hierarchical structures. In [18], Beneventano et al. briefly review these three product classification schemas.

Similarly, we establish the product category hierarchy for the analysis of dynamic reputation of a seller. We extend eCl@ss due to its reasonable classification in practice, i.e. products in eCl@ss are more functionally grouped and are subdivided for specific usage. For example, in eCl@ss, "*Digital Camera*" can be further classified as "*DSLR (Digital single-lens reflex cameras)*", "*Compact Digital Camera*" and "*Mirrorless Digital Camera*". But it is not subdivided in both UNSPSC and eBay. In addition, we sort out the logical relations between product categories in eCl@ss. Then, add the attribute, "*Brand*", to the product category hierarchy to support finer-grained analysis on transaction trust with "*drill-down*" and "*roll-up*" operations in the hierarchy. Under each brand, there are corresponding products that belong to this brand.

Fig. 3.1 presents a small part of our extended product category hierarchy. For instance, if the product is "*Apple iPod nano 16GB (mc696ll/a)*", then its ancestors in the product category hierarchy, in sequence, are "*Apple MP3 player (iPod)*" and "*MP3 player*". If the product is "*Apple iPhone5 16GB*", then its ancestors in the product category hierarchy, in sequence, are "*Apple iPhone*" and "*Smartphone*". The category hierarchy for the product "*Canon EOS 6D SLR Digital Camera*" is complex that has sever layers, and its ancestors are "*Canon DSLR camera*", "*DSLR camera*" and "*Digital camera*" in sequence. In our extend hierarchy, each product category has

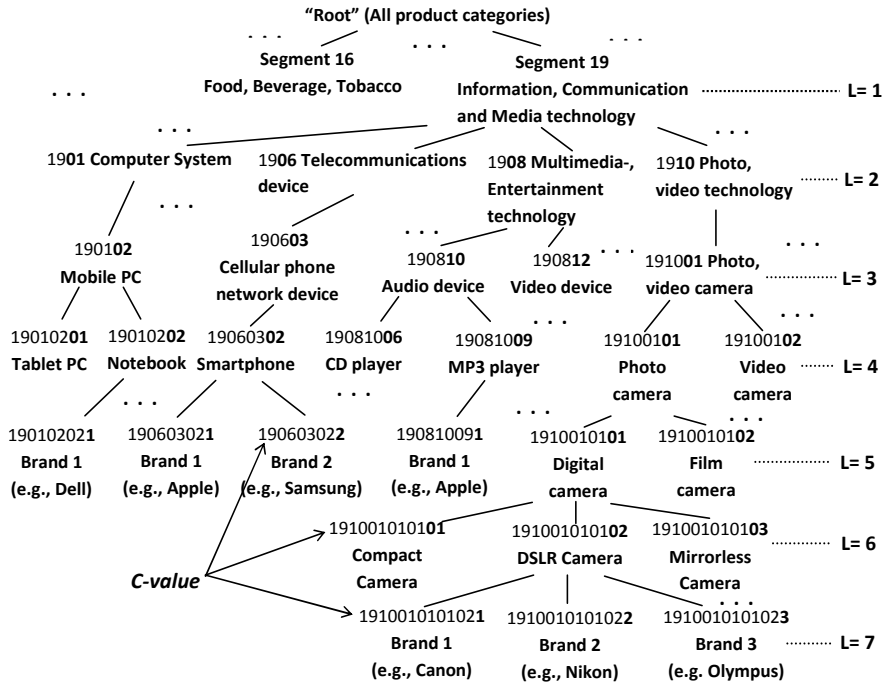


Figure 3.1: Part of product category hierarchy for the segment *"Information, communication and media"*

a unique id termed as *C-value* (see Fig. 3.1), with which a layer, the layer's parent and children can be located. Note that *C-value* is also available in eCl@ss.

3.2.1.2 Similarity Measurement within Product Category

To measure the similarity of two nodes within a hierarchical structure, a crucial factor is the depth d of the deepest common ancestor of the two nodes. For instance, the deepest common ancestor of *Notebook* and *Tablet PC* is *Computer System* (see Fig. 3.1), and the depth d of *Computer System* within the product hierarchy is 2. If the deepest common ancestor is located high in the hierarchy, it indicates that the two products have general classification without much similarity between them. If the deepest common ancestor is located in the low hierarchy, it indicates that the two products have a common classification with stronger similarity. Hence, the transaction item similarity S_{TI} between two products p and p' should be a monotone function with respect to the depth d of the deepest common ancestor of them. We use a hyperbolic tangent func-

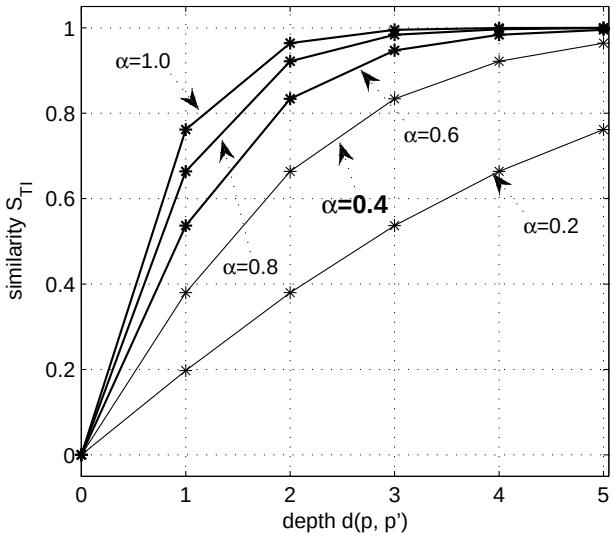


Figure 3.2: The influence of different α on similarity values

tion to follow this principle. Thus, the similarity of two products p and p' can then be defined as the function of d as follows.

$$S_{TI}(p, p') = \frac{e^{\alpha d(p,p')} - e^{-\alpha d(p,p')}}{e^{\alpha d(p,p')} + e^{-\alpha d(p,p')}} \tag{3.1}$$

where $\alpha > 0$ is a constant, and $d(p, p')$ denotes the the depth d of the deepest common ancestor between two products. Moreover, we draw the following properties from this product hierarchy (i.e. Fig. 3.1):

Property 1: When $d \geq 3$, the two products have strong similarity. For example, for *Tablet PC* and *Notebook*, their common ancestor is *Mobile PC* with $d = 3$.

Property 2: When $d \leq 2$, the two products have weak similarity. For example, for *Camera (digital)* and *Lens*, their common ancestor is *Photo technology, video technology* with $d = 2$.

Property 3: When the two products belong to different segments, their similarity is 0. For example, *Tablet PC* is in segment 19 and *Food* is in segment 16, as plotted in Fig. 3.1.

According to the above properties, we take the parameter $\alpha = 0.4$ to illustrate the

Table 3.1: Examples of transaction item similarity

Product Pair (p, p')	S_{TI}	d
Canon EOS T3i Rebel , Canon A2200 Digital Camera	0.96	5
Camera (digital), Video camera	0.83	3
Camera (digital), Lens	0.66	2
Laptop, Camera (digital)	0.38	1
Tablet PC, Food	0	0

comparison process of our approach, leading to the curve plotted in Fig. 3.2. Algorithm 1 presents how to find the deepest common ancestor of two products p and p' in a category hierarchy, and then return the depth d of their ancestor. Some examples of similarity measure are listed in Table 4.1.

Algorithm 1 Similarity Measurement within Product Category

Input: c_{code} and c'_{code} are the *C-value* of two products or items within category hierarchy, p, m, d.
Output: S_{TI}

for all numbers in c_{code} and c'_{code} **do**
  if c_{code} and c'_{code} contain different number **then**
    record position p.
    $m = p \; mod \; 2$
    if m=1 **then**
      set $d = floor(p/2)$
    else
      set $d = (p/2) - 1$
    end if
  else
    c_{code} and c'_{code} are the same; return $S = 1$
  end if
end for
return $S_{TI}(c_{code}, c'_{code}) = \frac{e^{\alpha d} - e^{-\alpha d}}{e^{\alpha d} + e^{-\alpha d}}$

3.2.2 Similarity Comparison of Transaction Amount

As pointed out in Section 3.1.3, when the transaction amount of a forthcoming transaction is much smaller or much larger than those of past transactions, the significant difference may lead to the *transaction amount imbalance*. Hence, the similarity com-

parison of transaction amount is important for trust evaluation.

For calculating the transaction amount similarity S_{TA} between the amount ta_p of a past successful transaction and the amount ta_f of a forthcoming one, we first consider two typical examples below:

(1) $ta_p = \$50$ while $ta_f = \$550$;

(2) $ta_p = \$5000$ while $ta_f = \$5,500$.

In the existing studies, the *difference* of transaction amounts (denoted by $D_{ta} = |ta_f - ta_p|$) has been used in trust evaluation [149, 157], but it cannot distinguish the above two examples. Although the values of D_{ta} in these two examples are both $\$500$, $\$500$ is only 10% increase in example (2) while it is 10-times increase in example (1). To this end, our model also introduces the *relative value* between ta_p and ta_f, denoted by R_{ta} (i.e. $R_{ta} = \frac{ta_f}{ta_p}$) for calculating S_{TA}. Both D_{ta} and R_{ta} are used to determine S_{TA} according to the following principles:

Principle 1: When R_{ta} is fixed, S_{ta} increases, if D_{ta} decreases;

Principle 2: When D_{ta} is fixed, S_{ta} becomes larger, if R_{ta} approaches 1.

Based on both principles, some definitions for calculating S_{TA} are proposed below.

Definition 2: If $ta_p \leq ta_f$, then the transaction amount similarity of ta_f and ta_p is

$$S_{TA}(ta_f, ta_p) = \varepsilon * f_D(D_{ta}(ta_f, ta_p)) + (1 - \varepsilon) * f_R(R_{ta}(ta_f, ta_p)) \tag{3.2}$$

The definition of function f_D in Eq. (3.2) can be based on the transaction amount category partitions in e-commerce environments that is proposed in [157].

$$f_D(D_{ta}(ta_f, ta_p)) = \frac{2}{e^{C(D_{ta}(ta_f, ta_p))*\beta} + e^{-C(D_{ta}(ta_f, ta_p))*\beta}} \tag{3.3}$$

where

$$C(D_{ta}(ta_f, ta_p)) = \begin{cases} 0, \ if \ D_{ta}(ta_f, ta_p) \in (0, 1] \\ 1, \ if \ D_{ta}(ta_f, ta_p) \in (1, 10] \\ 2, \ if \ D_{ta}(ta_f, ta_p) \in (10, 50] \\ 3, \ if \ D_{ta}(ta_f, ta_p) \in (50, 100] \\ 4, \ if \ D_{ta}(ta_f, ta_p) \in (100, 500] \\ 5, \ if \ D_{ta}(ta_f, ta_p) \in (500, 10^3] \\ 6, \ if \ D_{ta}(ta_f, ta_p) \in (10^3, 5 \times 10^3] \\ 7, \ if \ D_{ta}(ta_f, ta_p) \in (5 \times 10^3, 10^4] \\ 8, \ if \ D_{ta}(ta_f, ta_p) \in (10^4, 3 \times 10^4] \\ 9, \ if \ D_{ta}(ta_f, ta_p) \in (3 \times 10^4, 10^5] \\ 10, \ if \ D_{ta}(ta_f, ta_p) > 10^5 \end{cases}$$

$\varepsilon, \beta \in (0, 1]$ and we choose $\varepsilon = 0.5, \beta = 0.2$, as an example in our approach. As a result, if $ta_f = 550$ and $ta_p = 50$, $f_D(D_{ta}(ta_f, ta_p)) = 0.75$.

In addition, the definition of function f_R used in Eq. (3.2) is based on a threshold λ_R of *relative value* ($\lambda_R > 1$), which can be specified as a default value by the trust management authority (e.g. $\lambda_R = 20$). With λ_R, the function f_R can be defined below.

$$f_R(R_{ta}(ta_f, ta_p)) = \begin{cases} 0, & if \ R_{ta}(ta_f, ta_p) \geq \lambda_R \\ \frac{\lambda_R - R_{ta}(ta_f, ta_p)}{\lambda_R - 1}, & if \ R_{ta}(ta_f, ta_p) < \lambda_R \end{cases} \tag{3.4}$$

In Eq. (3.4), when $R_{ta} \geq \lambda_R$, $f_R(R_{ta}(ta_f, ta_p))$ is set to 0, which indicates a large *relative value* in transaction amounts between the forthcoming transaction and the past one. When $R_{ta} < \lambda_R$, $f_R(R_{ta}(ta_f, ta_p))$ is a projection from 0 to 1 of R_{ta}.

Definition 3: If $ta_p \geq ta_f$, then the transaction amount similarity of ta_f and ta_p can be calculated as:

$$S_{TA}(ta_f, ta_p) = \varepsilon * f_D(D_{ta}(ta_f, ta_p)) + (1 - \varepsilon) * f_R(R_{ta}(ta_f, ta_p)^{-1}) \quad (3.5)$$

where f_D is defined in Eq. (3.3) and f_R is defined in Eq. (3.4).

According to Definition 2 and Definition 3, when the past transaction amount ta_p gets closer to the forthcoming transaction ta_f, their transaction amount similarity S_{TA} has a higher value, which follows Principles 1 and 2.

3.2.3 Similarity Comparison of Transaction Time

As suggested by some studies on trust evaluation, ratings of recent transactions should be assigned higher weights in trust evaluation [120, 88, 147]. We use S_{TT} to denote the transaction time similarity between a forthcoming transaction and a past transaction, and use S_{TT} as the weight for the rating of a past transaction. The higher the transaction time similarity, the higher the weight for the rating of the past transaction.

Definition 4: During a time period $[t_1, t_n]$, where $t_k < t_{k+1}$ $(1 \leq k < n)$, t_n is the most recent transaction time, and the time for a forthcoming transaction is $t_n + 1$ ($t_n + 1 = t_{n+1}$). The transaction time similarity S_{TT} can be calculated by the exponential moving average:

$$S_{TT}(t_k, t_{n+1}) = \gamma^{t_n - t_k}, 0 < \gamma < 1, 0 < k \leq n \quad (3.6)$$

The exponential moving average has also been widely used in financial markets [99], as the weight for each older data decreases exponentially. Moreover, in Eq. (3.6), the value of γ should be high enough (e.g., $\gamma = 0.9$) so as to avoid that the weight decreases too rapidly.

3.3 A Trust Vector Approach to Outlining Reputation Profile

In the literature, there also exist some approaches using trust vectors, but most of them have no focus on context-aware trust evaluation. In [107], Ray et al. propose a trust vector that consists of experience, knowledge and recommendation. The focus is how to address these three independent aspects of trust. Zhao et al. [178] propose a method using a trust vector to represent the directed link with a trust value between two peers. In [147], Wang et al. use a trust vector to describe the trustworthiness and trust trend of sellers. Different from these works, in this section, we introduce a trust vector approach to context-aware transaction trust evaluation.

3.3.1 Trust Data Representation and Trust Metrics

This subsection first defines the data that are needed for computing trust values in e-commerce environments.

$$TR^{(t)} =< S; B; p; \textit{C-hrchy}; ta; t; r > \tag{3.7}$$

- $TR^{(t)}$ is a transaction between a seller S and a buyer B happening at time t;
- p is the product (i.e. transaction item) traded in the transaction $TR^{(t)}$;
- *C-hrchy* represents the path in product category hierarchy to which p belongs;
- ta is the transaction amount in transaction $TR^{(t)}$ for p;
- r is a rating (an integer in a range, e.g., $\{-1, 0, 1\}$ or $\{1, 2, 3, 4, 5\}$) that the buyer B gives to the seller S for $TR^{(t)}$ to reflect a seller's performance during the whole transaction;
- A set of n past transactions are denoted as $Trans = \{TR^{(t_1)}, TR^{(t_2)}, ..., TR^{(t_n)}\}$.

In addition, the approaches proposed in this thesis are considered to be directly applied in large-scale e-commerce applications. Thus, for trust metrics, in line with the studies [120, 64, 158, 159, 153, 147, 88], our work adopts heuristic-based [124] (see Section 2.2.2) techniques to aggregate and average trust ratings as the trustworthiness or reputation values of a seller.

3.3.2 A Trust Vector

Our proposed trust vector approach consists of three major elements, which are called *Contextual Transaction Trust (CTT)* values. For each element in the trust vector, the higher the value, the more trustworthy the seller will be.

(a) **Transaction Item Specific Trust (TIST):** TIST is the average of all the ratings $\{r_i\}$ in the past transactions $Trans$ for trading the same transaction item p as in a forthcoming transaction.

(b) **Product Category based Trust (PCT):** PCT is the average of all the ratings $\{r_i\}$ of the past transactions $Trans$ for selling the products in a product category (e.g., "*Canon DSLR cameras*" or "*DSLR cameras*") of p (e.g., "*Canon 6D DSLR camera*") in the product category hierarchy (see Fig. 3.1). When computing PCT, a price range covering the price ta and a time range can be specified as the parameters. The variables p and ta come from the context of the forthcoming transaction.

(c) **Similar Transaction Amount based Trust (STAT):** STAT is the trust value of a seller in a specific price range covering price ta and a time range. STAT is important for analysing the trustworthiness of a seller in different price ranges.

In the above trust vector, all three CTT values are associated with both past transactions and the forthcoming transaction. With the same seller, but a different forthcoming transaction, the computed trust values may be different. Even with the same forthcoming transaction, the trust values can vary. This is because a buyer can specify

and change the layer in the product category hierarchy, the price range and the time period for computing the last two CTT values: PCT and STAT. With the combinations of the parameters specified in all three context dimensions, different sets of CTT values can be computed, all of which can outline the reputation profile of the seller indicating the trustworthiness in various types of transactions.

3.3.3 Similarity Used in Trust Vector Computation

This subsection presents theoretical solutions for computing our proposed trust vector by taking advantage of similarity comparison introduced in Section 3.2.

3.3.3.1 Transaction Item Specific Trust (TIST)

Transaction Item Specific Trust (TIST) in the trust vector takes into account past transactions, which sell the same transaction item as the forthcoming transaction. However, three cases are identified below in the calculation of TIST.

Case 1: If there is a sufficient number of ratings (named as "direct reference" ratings) from past transactions selling the same item as a forthcoming one, TIST can be determined from these ratings directly;

Case 2: If the transaction item in a forthcoming transaction has never been sold by a seller, as stated in Section 3.2, the value of TIST can to be inferred from the ratings on the transactions selling different items (named as "indirect reference" ratings). Transaction context similarity should be compared to discount these ratings [97];

Case 3: If both cases are not true, we need to use both "direct reference" ratings and "indirect reference" ratings to compute TIST.

Transaction Content Similarity: Before giving formulas for calculating TIST, we first introduce *transaction content similarity* S_{TC}, which includes transaction item similarity S_{TI} and transaction amount similarity S_{TA}.

Definition 5: With a set of n past transactions $Trans = \{TR^{(t_1)}, TR^{(t_2)}, ..., TR^{(t_n)}\}$ between buyers and seller S in the time period $[t_1, t_n]$ and $TR^{(t_f)}$ denoting the forthcoming transaction at time t_{n+1}, we define *transaction content similarity* S_{TC} as:

$$S_{TC}(TR^{(t_i)}, TR^{(t_f)}) = \frac{S_{TI}(TR^{(t_i)}, TR^{(t_f)}) + S_{TA}(TR^{(t_i)}, TR^{(t_f)})}{2} \tag{3.8}$$

where $S_{TC} \in [0, 1]$. For convenience of description, in the following, we denote $S_{TC}(i, f) = S_{TC}(TR^{(t_i)}, TR^{(t_f)})$, $S_{TI}(i, f) = S_{TI}(TR^{(t_i)}, TR^{(t_f)})$, $S_{TA}(i, f) = S_{TA}(TR^{(t_i)}, TR^{(t_f)})$ and $S_{TT}(i, f) = S_{TT}(TR^{(t_i)}, TR^{(t_f)})$ (defined in Eq. (3.6)).

The Calculation of TIST: Assume a buyer B is planning to buy a product p in a forthcoming transaction $TR^{(t_f)}$ from seller S.

In Case 1, we use θ to denote the threshold of sufficient number of "direct reference" ratings[4]. Thus, all the "direct reference" ratings are used in the calculation of TIST, where the transaction time similarity S_{TT} is used to weight each rating [120, 88, 147, 153].

$$T^{[t_1,t_n]}_{TIST_{s(p)}} = \frac{\sum_{i=1}^{m_1}(r(TR^{(t_i)}) * S_{TT}(i, f))}{\sum_{i=1}^{m_1} S_{TT}(i, f)} \tag{3.9}$$

In Eq. (3.9), m_1 is the number of "direct reference" ratings from past transaction set $Trans$, $\theta_1 \leq m_1 \leq n$.

In Case 2, when evaluating TIST, *transaction content similarity* S_{TC} is regarded as the weight to discount these "indirect reference" ratings, and thus TIST can be computed as:

$$T^{[t_1,t_n]}_{TIST_{s(p)}} = \frac{\sum_{i=1}^{n}(r(TR^{(t_i)}) * S_{TC}(i, f) * S_{TT}(i, f))}{\sum_{i=1}^{n} S_{TT}(i, f)} \tag{3.10}$$

In Case 3, there are not enough "direct reference" ratings from past transaction set $Trans$. It is necessary to combine both "direct reference" and "indirect reference"

[4]The parameters θ can be specified by buyers or by the trust management authority.

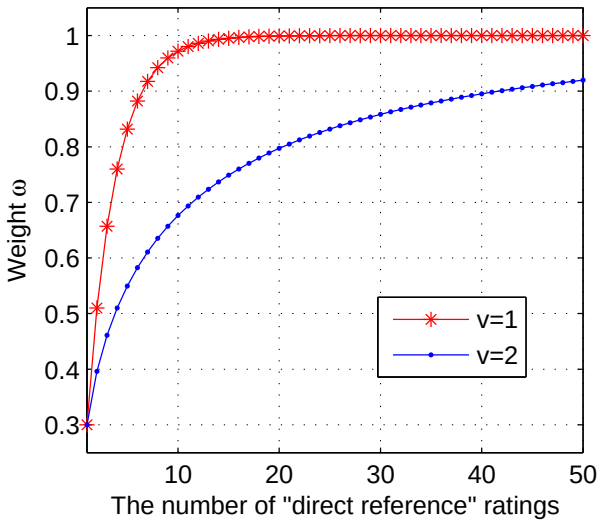


Figure 3.3: The variation of ω ($u = 0.7$)

ratings, and then give "direct reference" ratings higher weight ω. Moreover, the determination of weight ω should follow some principles.

Principle 3: When the number of "direct reference" ratings increases, the weight for these ratings increases as well.

Principle 4: The initial value of weight ω should be low to avoid that trust level of seller S tears down very fast after a few misbehaviours at the beginning.

Following these two principles, our model uses a function in Eq. (3.11) with two parameters u and v to control the changes of ω.

Definition 6: Given parameters u and v, the weight ω can be calculated as follows:

$$\omega(m_1) = 1 - u^{m_1^{\frac{1}{v}}}, u \in (0, 1) \tag{3.11}$$

where u determines the initial value of weight ω and $v \in \{1, 2, 3, ...\}$. According to Principle 4, the value of u should be more than 0.5 (i.e. $0.5 < u < 1$). The variations of ω are plotted in Fig. 3.3. With two fixed parameters u and v, the larger m_1, the larger is $\omega(m_1)$, which confirms Principle 3.

The value of v depends on the parameter θ_1 (i.e., the threshold of sufficient number of "direct reference" ratings). For example, if θ_1 is 10, $v = 1$ is suitable according to Fig. 3.3. If θ_1 is 50, $v = 2$ is suitable.

TIST in Case 3 is the weighted summation of $T^{[t_1,t_n]}_{TIST_{s(p)}}$ defined in both Eq. (3.9) and Eq. (3.10):

$$T^{[t_1,t_n]}_{TIST_{s(p)}} = \omega(m_1) * \frac{\sum_{i=1}^{m_1}(r(TR^{(t_i)}) * S_{TT}(i,f))}{\sum_{i=1}^{m_1} S_{TT}(i,f)} +$$

$$(1 - \omega(m_1)) * \frac{\sum_{j=1}^{m-m_1}(r(TR^{(t_j)}) * S_{TC}(i,f) * S_{TT}(i,f))}{\sum_{j=1}^{n-m_1} S_{TT}(i,f)} \tag{3.12}$$

3.3.3.2 Product Category based Trust (PCT) and Similar Transaction Amount based Trust (STAT)

In addition to TIST, the buyer may also be concerned about whether the seller obtained a high level of trust in selling various products similar to p. *Product Category based Trust* (PCT) is computed based on the ratings of transactions containing products similar to that in the forthcoming transaction. Its value is defined in Eq. (3.13):

$$T^{[t_1,t_n]}_{PCT_{s(p)}} = \frac{\sum_{k=1}^{m_3}(r(TR^{(t_k)}) * S_{TT}(k,f))}{\sum_{i=1}^{m_3} S_{TT}(k,f)} \tag{3.13}$$

where $m_3 = \mid \{TR^{(t_k)} | TR^{(t_k)} \in Trans, S_{TI}(k,f) \geq \theta_{TI}\} \mid$ and $m_1 \leq m_3 \leq m$; θ_{TI} is the threshold for transaction item similarity S_{TI}, i.e. PCT considers the ratings from the transactions selling products with a similarity larger than θ_{TI}. Essentially, the process of computing PCT is equivalent to performing "*roll-up*" operations in the hierarchy, and the number of "*roll-up*" operations determine the θ_{TI}.

Similar Transaction Amount based Trust (STAT) is to outline the trustworthiness of the forthcoming transaction in terms of transaction amount. The STAT is different from *PCT*, because a seller may have lots of past transactions with the amounts similar

to the forthcoming transaction but their corresponding transaction items may be in different product categories.

$$T_{STAT_{s(p)}}^{[t_1,t_n]} = \frac{\sum_{l=1}^{m_4}(r(TR^{(t_l)}) * S_{TT}(l,f))}{\sum_{l=1}^{m_4} S_{TT}(l,f)} \tag{3.14}$$

where $m_4 = \mid \{TR^{(t_l)} | TR^{(t_l)} \in Trans, S_{TA}(l,f) \geq \theta_{TA}\} \mid$ and $m_1 \leq m_4 \leq m$; θ_{TA} is the threshold for transaction amount similarity S_{TA}, i.e. STAT considers the ratings with the transaction amount similarity higher than θ_{TA}.

3.4 Empirical Studies

In this section, empirical studies are presented to illustrate the effectiveness our proposed trust vector approach for context-aware transaction trust evaluation in e-commerce environments.

3.4.1 Study 1 - Transaction Item Specific Trust (TIST)

In this study, an example is used to study the changes of TIST when a seller provides new products.

Example: Four sellers S_1, S_2, S_3 and S_4 provide a latest popular model of *Apple MacBooK Pro laptop (e.g., MC700LL/A)* with an attractive price of around $\$900$ at time t ($t > t_{50}$). Assume that the four sellers sold different products respectively before, but since t_{51} they start to sell this popular laptop. The products that they have sold before are listed in Table 3.2 below.

Furthermore, the example adopts the rating model where each rating $r(TR^{(t_i)})$ is an integer in $\{1,2,3,4,5\}$[5]. We normalize ratings to $[0,1]$, namely $\{0, 0.25, 0.5, 0.75, 1\}$. A good quality transaction means the value of $r(TR^{(t_i)})$ lies in the range $[0.75, 1]$. By contrast, the $r(TR^{(t_i)})$ for a poor quality transaction lies in the range $[0, 0.25]$. We

[5]This rating model provides more accurate information than the tripple-rating model with 1 for positive, 0 for neutral and -1 for negative as eBay [88]

Table 3.2: The products sold by four sellers

	transaction context			
	time period $[t_1, t_{50}]$		time period t_{51} and after	
Sellers	Product	Price	Product	Price
S_1	Apple iPad Air	\$600	MacBook Pro Laptop	\$900
S_2	Canon A2200 Digital Camera	\$150	MacBook Pro Laptop	\$900
S_3	Luxury Watch	\$2500	MacBook Pro Laptop	\$900
S_4	Food	\$10	MacBook Pro Laptop	\$900

adopt other parameters $\gamma = 0.9$ in Eq. (3.6), $v = 2$ and $u = 0.7$ in Eq. (3.11), $\lambda_R = 20$ in Eq. (3.4), and $\theta = 20$. We also assume that the four sellers all had good quality transactions during time period $[t_1, t_{50}]$ and obtained similar high rating values. After time t_{50}, S_1 and S_3 still have good quality transactions but S_2 and S_4 start to provide poor quality transactions (e.g., they sell refurbished laptops).

Note that this example uses the above four typical sellers which aims to fully demonstrate the changes of TIST value under different situations. More specifically, compared with their past transactions, seller S_1 has similar transaction item and transaction amount in the new transaction; seller S_2 has similar transaction item but different transaction amount; seller S_3 has similar transaction amount but different transaction item; seller S_4 has totally different transaction item and transaction amount. In addition, these four sellers are set to perform differently so as to reflect and observe the fluctuation of TIST value. The corresponding performance includes continuously having good quality transactions and having poor quality transactions after a series of good quality transactions.

Evaluation: In order to highlight the changes of TIST value, we compare it with the *general transaction trust* (GTT) value which is calculated by a trust evaluation approach without context consideration proposed in [154]:

$$T_{GTT_s}^{[t_1,t_n]} = \frac{\sum_{i=1}^{n}(r(TR^{(t_i)}) * S_{TT}(i,f))}{\sum_{i=1}^{n} S_{TT}(i,f)} \tag{3.15}$$

where $TR^{(t_i)} \in Trans$, $TR^{(t_f)}$ is the forthcoming transaction and S_{TT} is transaction

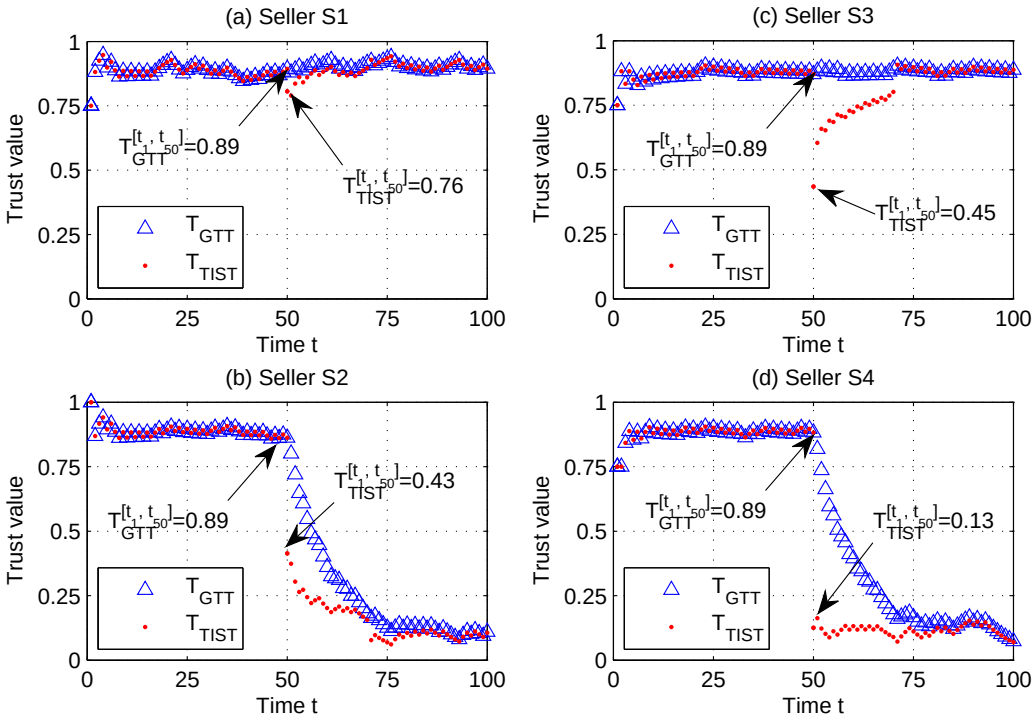


Figure 3.4: The changes of Transaction Item Specific Trust (TIST) for four different sellers

time similarity (defined in Eq. (3.6)), used to weight the rating of $TR^{(t_i)}$.

Results: The comparative results of the TIST value and the GTT value are shown in Fig. 3.4.

(a) As shown from Fig. 3.4(a) to 3.4(d), the TIST values and the GTT values of four sellers are very close to each other in time period $[t_1, t_{50}]$ (i.e., $T_{GTT}^{[t_1,t_{50}]} \approx 0.89$), when the products they sell do not change.

(b) At time t_{51}, when the four sellers start to provide new products, their TIST values decrease differently. The extent of decrease depends on the similarity between past transactions and the new transaction at time t_{51}.

(1) In Fig. 3.4(a), S_1, who has approximately the same transaction items (*iPad Air vs Laptop*) and transaction amounts ($600 *vs* $900) as those in the past transactions, the TIST value decreases slightly only.

(2) In Fig. 3.4(b) and 3.4(c), both S_2 and S_3, with decreased transaction context similarity (i.e., *Digital Camera vs Laptop* and $\$150$ *vs* $\$900$ for S_2, *Luxury Watch vs Laptop* and $\$2500$ *vs* $\$900$ for S_3), the TIST value drops remarkably. However, it also can be observed in Fig. 3.4(c) that $T_{TIST_{S_3}}$ quickly reaches a high value after accumulating for a while (i.e., above 0.8 after t_{75}) when S_3 keeps selling high quality laptops.

(3) In Fig. 3.4(d), S_4, who starts to have transactions at t_{51} with completely different transaction context (*Food vs Laptop* and $\$10$ *vs* $\$900$), his TIST values are quite low due to the potential *transaction context imbalance*.

(c) After time t_{51}, as depicted from Fig. 3.4(a) to 3.4(d), when the four sellers have completed a series of transactions, the TIST values and the GTT values tend to be close to each other again due to the increased number of "direct reference" ratings. The TIST values can be determined from them directly (Case 1 in Eq. (3.9)), which are the same as the GTT values (calculated in Eq. (3.15)).

Summary: (1) Obviously, it is unreasonable to use GTT to indicate the trust level of sellers, because it cannot reflect the changes of trustworthiness when a seller starts to provide new products. (2) By contrast, our proposed TIST takes into account the transaction context similarity. Only for those sellers who provide transaction items and transaction amounts similar to a forthcoming transaction, the trust value will be as high as before (such as S_1). (3) Therefore, TIST can be used to identify some potentially malicious transactions with the *transaction amount imbalance* problem, such as S_4 who obtains a quite low TIST value with our model.

3.4.2 Study 2 - Comparison with Existing Trust Evaluation Approaches

In this study, we compare our approach with a single-context but multi-faceted trust valuation model REGRET [120] and a prior trust vector based approach proposed by

Wang and Lim [149].

3.4.2.1 Comparison with REGRET

As depicted in Section 3.1.2.2, the rating $r(TR^{(t_i)})$ is usually regarded as a "general" rating in e-commerce, which combines with trust information from different aspects, such as the quality of product p, sellers' services (e.g., whether seller S processed buyer B's order on time and whether seller S had prompt and friendly communication with buyer B), delivery or shipping services, etc. In REGRET [120], the above multi-faceted trust is considered where a buyer B can specify the weight in each aspect based on personal preference, and then an aggregated value is computed to reflect the trustworthiness of seller S.

However, in the subsection, an example is given to explain that, compared with our proposed trust vector based approach, trust evaluation based on the REGRET model may lead to an unreasonable result for seller selection. To be precise, let us suppose that two other sellers, S_5 and S_6, have completed a series of transactions in time period $[t_1, t_{10}]$. The products that they have sold and the corresponding ratings that they have obtained are listed in Table 3.3. Note that, to facilitate discussion, we also assume that: 1) trust value computed based on the REGRET model derives from three aspects: the quality of product, the seller's service and delivery service. Correspondingly, as shown in Table 3.3, $r(TR^{(t_i)}) =< r_q(TR^{(t_i)}), r_s(TR^{(t_i)}), r_d(TR^{(t_i)}) >$ is a rating vector[6] that buyer B gives to seller S for $TR^{(t_i)}$, which consists of three ratings. $r_q(TR^{(t_i)})$ is the rating for the quality of product p; $r_s(TR^{(t_i)})$ is the rating for the seller's service; and $r_d(TR^{(t_i)})$ is the rating for delivery service; and 2) the "general" rating $r(TR^{(t_i)})$ equals to $r_q(TR^{(t_i)})$, since the quality of product determines transaction quality to a large extent in practice.

Then, we compare the trust value of two sellers computed based on the REGRET

[6]As mentioned in Section 3.1.2.2, at eBay, a buyer have to provide a rating vector after each transaction. Usually, there are five elements in that rating vector. For sake of simplicity, we only use three elements $r_q(TR^{(t_i)}), r_s(TR^{(t_i)})$ and $r_d(TR^{(t_i)})$ as the example which are also available at eBay. In addition, as introduced in Study 1, all the ratings will be normalized.

model and our approach as shown in Table 3.4, and we set the same parameters as the example introduced in Section 3.4.1.

Table 3.3: Rating vectors of sellers S_5 and S_6

transaction time	S_5			transaction item
	$r_q(TR^{(t_i)})$	$r_s(TR^{(t_i)})$	$r_d(TR^{(t_i)})$	
t_1	0.75	1	0.75	iPad Air
t_2	0.75	0.75	1	MacBook Pro Laptop
t_3	0.5	1	1	iPad Air
t_4	1	0.75	0.75	iPad Air
t_5	0.75	0.75	0.75	MacBook Pro Laptop
t_6	0.5	1	0.75	iPad Air
t_7	0.75	0.75	1	iPad Air
t_8	1	1	0.75	MacBook Pro Laptop
t_9	0.75	0.75	1	iPad Air
t_{10}	0.5	0.75	1	iPad Air
transaction time	S_6			transaction item
	$r_q(TR^{(t_i)})$	$r_s(TR^{(t_i)})$	$r_d(TR^{(t_i)})$	
t_1	1	0.75	1	Canon A2200 Digital Camera
t_2	1	1	1	Canon A2200 Digital Camera
t_3	0.75	0.75	0.75	Canon A2200 Digital Camera
t_4	0.5	1	1	MacBook Pro Laptop
t_5	0.75	1	0.75	Canon A2200 Digital Camera
t_6	1	1	0.75	Canon A2200 Digital Camera
t_7	0.5	0.75	1	MacBook Pro Laptop
t_8	1	1	0.75	Canon A2200 Digital Camera
t_9	0.25	0.75	1	MacBook Pro Laptop
t_{10}	1	1	1	Canon A2200 Digital Camera

Analysis and Summary: In REGRET, as both sellers sell this laptop with an attractive price of $\$900$, they have the same level of trustworthiness on price. Based on the ratings $r_q(TR^{(t_i)})$ in time period $[t_1, t_{10}]$, we can know that the trustworthiness of a new transaction selling this laptop is 0.71 for S_5 and the trustworthiness of service quality is 0.89 for S_5 based on ratings $r_d(TR^{(t_i)})$. Similarly, we know that the trustworthiness of a new transaction selling this laptop is 0.74 for S_6 and the trustworthiness of service quality is 0.90 for S_6 (see Table 3.4). Thus, with any weights specified by a buyer

that are used for both sellers simultaneously, the aggregated trust value of S_6 is always greater than S_5, i.e. S_6 is more trustworthy, denoted as $T_{S_5}^{[t_1,t_{10}]} < T_{S_6}^{[t_1,t_{10}]}$. Note that the higher the value, the more trustworthy the seller will be.

Table 3.4: The computed trust values of sellers S_5 and S_6 based on REGRET and the proposed trust vector approach

	REGRET		Trust Vector		
Seller	transaction trustworthiness	service trustworthiness	T_{TIST}	T_{PCT}	T_{STAT}
S_5	0.71	0.89	0.75	0.72	0.72
S_6	0.74	0.90	0.40	0.76	0.40

In Table 3.3, the ratings $r_q(TR^{(t_i)})$ of S_5 for selling *MacBook Pro laptop* are higher than S_6. Compared to the single value result $T_{S_5}^{[t_1,t_{10}]} < T_{S_6}^{[t_1,t_{10}]}$ from REGRET system, our approach is more reasonable when selecting a seller for a specific product, since the computed trust vector is particularly bound to each forthcoming transaction. As we can see in Table 3.4, S_6 has a lower T_{TIST} value (0.40 *vs* 0.75) and T_{STAT} value (0.40 *vs* 0.72) than S_5.

3.4.2.2 Comparison with Prior Trust Vector based Approach

In [149], Wang and Lim propose a preliminary trust vector approach for evaluating the context-aware transaction trust, which includes 1) *product specific trust (ST)*, calculated from the ratings of all past transactions selling the same product as the forthcoming transaction; 2) *product category specific trust (SCT)*, based on the ratings of past transactions having the same product category as that in the forthcoming transaction; 3) *transaction amount specific trust (TAST)*, based on the ratings of past transactions having the same transaction amount category as that in the forthcoming transaction; and 4) *global weighted trust (GWT)*. The GWT value is the weighted average of ratings from all past transactions in a recent time period, in which the transaction amount difference between the forthcoming transaction, and each past transaction, is used to calculate the weight for the rating of the past transaction.

Table 3.5: The computed two trust vectors of sellers S_1 and S_3 at time t_{51}

	Trust vector in [149]				Our proposed Trust vector		
Sellers	T_{ST}	T_{SCT}	T_{TAST}	T_{GWT}	T_{TIST}	T_{PCT}	T_{STAT}
S_1	0	0	0.89	0.67	0.76	0.89	0.89
S_3	0	0	0.89	0.86	0.45	0	0.89

Analysis and Summary: However, their model does not consider transaction item similarity, i.e. hierarchical structure of product category. As a result, the selection of the most trustworthy seller may be unreasonable when a seller just starts to sell a new product in the forthcoming transaction. As introduced in Study 1, for two sellers S_1 and S_3, we assume that S_3 sells a Luxury Watch for $\$1,000$. When S_1 and S_3 start to sell a model of *MacBook Pro Laptop* at time t_{51}, the calculated trust vectors of S_1 and S_3 using the approach in [149] and our approach (with $\theta_{TI} = 0.8$ and $\theta_{TA} = 0.8$) are both listed in Table 3.5, respectively.

Following the model in [149], with any specified weights, considering the aggregated trust values of two sellers, it always has $T_{S_3}^{[t_{41},t_{50}]} > T_{S_1}^{[t_1,t_{50}]}$ at t_{51}. That is because both S_1 and S_3 have the same T_{ST}, T_{SCT} and T_{TAST} values, but T_{GWT} of S_3 is higher than that of S_1 (0.86 *vs* 0.67). By contrast, in our model, transaction item similarity is introduced. Thus, S_1, who has sold more transaction items similar to the one in the forthcoming transaction, obtains a higher transaction specific trust value. Compared with S_3, S_1 has higher T_{TIST} value (0.76 *vs* 0.45) and T_{PCT} (0.89 *vs* 0) value.

3.5 Summary

In this chapter, we first have differentiated the definitions of context in different applications, and then discussed and defined transaction context in e-commence environments. We have identified three important transaction context dimensions, i.e. *Product Category*, *Transaction Amount (Price)* and *Transaction Time* with influence on the trustworthiness of a forthcoming transaction. Second, in situations where there are

no or not enough ratings from the transactions with the same context as the forthcoming transaction, we propose a set of methods to calculate transaction context similarity. Third, a trust vector approach has been proposed to outline a seller's reputation profile. Finally, we have studied trust vector based approach both analytically and empirically to illustrate its effectiveness.

This chapter is the basis of this thesis. As mentioned in Section 2.3, our proposed trust vector provides more detailed and comprehensive trust information of a seller. In particular, it reflects a seller's dynamic trustworthiness in various transaction contexts and identifies risks potentially existing in a forthcoming transaction, thus outperforming single-value trust valuation methods. On the other hand, multi-context transaction trust computation is also complex, in order to clearly outline a seller's reputation profile, if a user can specify or adjust layers in the product category, price range as well as transaction time range, accordingly, different ratings from different transaction contexts need to be taken into account for computation. If this trust vector is applied in e-commerce environments with millions of transactions, all these factors incur a high computational complexity. Thus, in the following chapters, we will focus on designing the new data structures to store the trust data and computation results. In addition, efficient algorithms are in high demand to facilitate buyers' context-aware trust enquiries on each element of the trust vector.

Chapter 4

Two-Dimensional Range Aggregate (RA) and CTT Computation

Over the past few years, in e-commerce and e-service environments, it has been receiving much attention from researchers to build various trust evaluation models [120, 158, 88, 146]. In brief, the basic idea of most existing trust evaluation models is to rate sellers (or service providers), and then use the aggregated ratings as the indication of their trustworthiness or reputation score. However, a single value only reflects the general trustworthiness of a seller without taking any transaction context information into account. With such trust evaluation models, buyers (or consumers) are vulnerable to some frauds from malicious sellers [66, 67, 58, 57].

In contrast to most existing trust management models that compute a single trust value, in Chapter 3, we have proposed to compute a trust vector for a seller. The computation of trust values in the trust vector takes transaction context into account and is associated with a forthcoming transaction.

The trust vector consists of three major elements, which are called Contextual Transaction Trust (CTT) values. They are

(a) ***Transaction Item Specific Trust (TIST):*** *the trustworthiness of a seller in selling a specific product to be traded in a forthcoming transaction;*

(b) ***Product Category based Trust (PCT):*** *the trustworthiness of the seller in a layer in the product category hierarchy that is higher than the specific product to be*

traded in the forthcoming transaction, within a price range and a time period;

(c) ***Similar Transaction Amount based Trust (STAT):*** *the trustworthiness of the seller in a valid price range and a time period.*

For each element in the trust vector, the higher the value, the more trustworthy the seller will be. When computing the last two elements, the parameters, such as product category, price range and time range, can be specified and adjusted by the buyer. For example, if "*Apple iPhone5s 16GB*" is the product in the forthcoming transaction, the buyer can specify and adjust *"product category"* along a path in the product category hierarchy, such as, "*Apple iPhone*" and "*Smartphone*", in sequence. Meanwhile, the buyer may also specify and adjust the price range and the time range. Each price range takes the price of a product as approximately the medium value. Each time range takes the recent time period. Note that, in Section 2.1.2, we have pointed out the temporal characteristic of trust. More specifically, the concerns of trust and time in the literature are discussed in three major ways: trust decay, trust time window and hybrid. In Section 3.3.3, like the studies in [120, 88, 147, 153], we adopt trust decay and weight more to recent ratings for computing the trust vector. However, trust decay is mainly of theoretical interest, since a seller may have hundreds of transactions within a short time period in practice. Therefore, to facilitate application in large-scale e-commerce websites, like PeerTrust [158, 159] or trust evaluation model used at eBay, a more reasonable approach is to allow buyers to choose the recent time window, such as "*the latest 1 month*", "*the latest 6 months*" or "*the latest 12 months*".

We use *granularity* to represent the differences in transaction context determined by a layer in the product category hierarchy, a price range and a time period. In addition, we term the query on CTT values as a *CTT query*, and term the computation of CTT values as *CTT computation*. Hence, with all computed trust results, the *reputation profile* of a seller can be outlined, which can indicate the dynamic trustworthiness of a seller in different products and product categories, price ranges, time periods and necessary combination of them. We term this new trust model as *ReputationPro*, which

greatly helps identify the *transaction context imbalance* problem potentially existing in forthcoming transactions, and thus avoid monetary losses of buyers.

However, at e-commerce websites, a popular seller usually sells a wide variety of products distributed in a number of product categories. In addition, a large number of buyers can be accessing one seller's reputation data simultaneously with regard to their potentially forthcoming transactions. In order to promptly answer a buyer's CTT requests, it is necessary to pre-compute aggregates over large-scale transaction data and ratings with necessary combinations of three context dimensions, i.e. Product Category, Price and Transaction Time. Therefore, the *CTT computation* for outlining sellers' reputation profiles is a very challenging problem that requires new data structures and novel algorithms.

This chapter is organized as follows. In Section 4.1, we introduce how to extend two-dimensional (2D) Range Aggregate (RA) for CTT Computation. Three preliminary solutions, namely, *eaR-tree*, *eaP-tree* and *eH-tree* are proposed for efficient CTT computation in Section 4.1. These approaches particularly fit the CTT computation with large-scale transaction data and ratings over a long time period. Section 4.3 presents the experimental results to illustrate both advantages and disadvantages of these approaches. Section 4.4 summarises our work in this chapter.

4.1 Extending Two-Dimensional RA for CTT Computation

As illustrated in Section 3.3.1, like the existing studies in [120, 64, 158, 159, 153, 147, 88], our proposed *ReputationPro* trust model adopts heuristic-based techniques that average the ratings for calculating the trust value. To this end, two aggregates are pre-computed and stored separately. They are $count\_r$, the number of ratings of the corresponding transactions, and sum_r, the sum of ratings in a specific layer of product category hierarchy within a specific transaction price range and a specific time

period. With a pair of $count\_r$ and sum_r, accordingly, the trust value can be computed as $T = \frac{sum\_r}{count\_r}$. In addition, based on the parameters of a CTT query, a set of $\{count\_r_i, sum\_r_i\}$ can be returned. Accordingly, the trust value is $T = \frac{\sum sum\_r_i}{\sum count\_r_i}$.

4.1.1 Relationship Between Two-Dimensional RA and CTT Computation

In this subsection, we discuss the relationship between the two-dimensional RA problem and our targeted CTT computation problem. In Section 3.1.2, we have introduced that transaction context includes a static and hierarchical dimension, i.e. Product Category, and two dynamic linear dimensions, i.e. Transaction Amount (Price) and Transaction Time. When computing PCT and STAT values, a CTT query covers both the Transaction Amount dimension and the Transaction Time dimension. Similar to the case depicted in Fig. 2.3 in Section 2.4.2, a CTT query can first be regarded as an RA problem in a two-dimensional space, where the x-axis represents the Transaction Time dimension in days and the y-axis represents the Transaction Amount dimension. Consequently, a CTT query on a seller in a time range $[t_1, t_2]$ and a transaction amount range $[ta_1, ta_2]$ can be converted by computing the number of the ratings $count\_r$ and the sum of the ratings sum_r of the transactions that fall into the query range formed by $[t_1, t_2]$ and $[ta_1, ta_2]$. Then, we further extend RA in a two-dimensional space and take Product Category as the third dimension.

4.1.2 The Extension Process

This subsection introduces how to extend the two-dimensional RA problem to CTT computation after taking into account the Product Category as the third dimension:

Step 1: Each transaction has a numeric string *C-hrchy* (see the definitions in Section 3.3.1) to represent the path in the product category hierarchy to which the product traded in the transaction belongs;

Following eCl@ss introduced in Section 3.2.1.1, a two-digit number is added to each layer of the product category hierarchy. Thus a unique *C-value* is assigned to each product category. For example, in Fig. 3.1, the node "*MP3 player*" at layer 4 has the *C-value* of "19081009" representing the path from the "product category root" to it. As the products traded in the transactions are at the bottom of product category hierarchy, the value of *C-hrchy* equals the *C-value* in the corresponding brand-based product category;

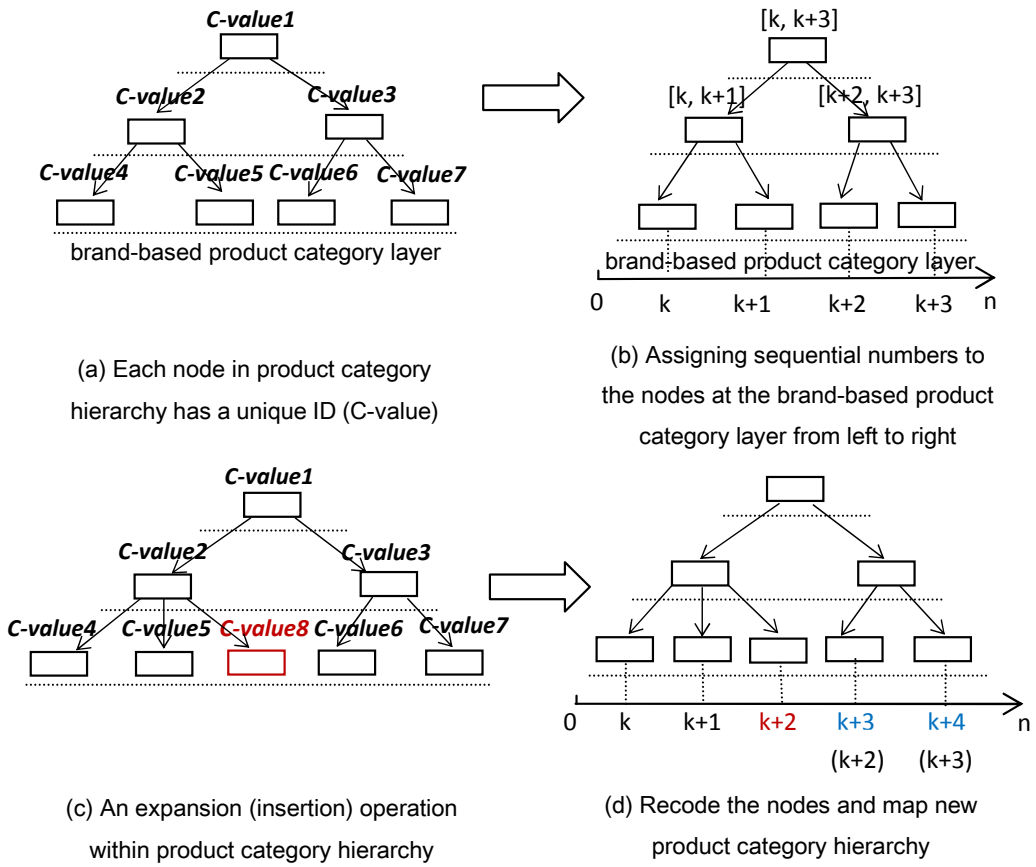
Step 2: Each product category in the product category hierarchy maintains the aggregates $count\_r$ and sum_r that are obtained from the past transactions selling the products in this product category as well as the corresponding transaction amount range and transaction time range;

Step 3: Each product category is an intermediate node in the product category hierarchy. In the meantime, for each brand-based product category (e.g., "*Canon SLR Digital Camera*"), it is the root of a subtree that is external to the hierarchy.

This subtree can be regarded as a tree for solving the RA problem in a two-dimensional space, which records the pairs of $count\_r$ and sum_r in the Transaction Amount dimension and the Transaction Time dimension. Accordingly, as we take the points depicted in Fig. 2.3 as transactions, all these transactions should belong to the same brand-based product category. Also, the subtree can be of multiple layers, depending on the number of transactions and the distributions in transaction amount and transaction time in the corresponding brand-based product category.

4.1.3 Why Not a Box Aggregate Problem?

Our targeted CTT computation can also be transformed into a three-dimensional box aggregate problem where the point aggregation is performed in three linear dimensions [167, 134]. More specifically, as shown in Fig. 4.1(a) and (b), instead of assigning a



(a) Each node in product category hierarchy has a unique ID (C-value)

(b) Assigning sequential numbers to the nodes at the brand-based product category layer from left to right

(c) An expansion (insertion) operation within product category hierarchy

(d) Recode the nodes and map new product category hierarchy

Figure 4.1: Convert Product Category dimension to a linear dimension

C-value for each node in product category hierarchy, only the nodes at the brand-based product category layer (i.e. the bottom of product category hierarchy) are assigned the numbers. However, the numbers assigned to the nodes at the brand-based product category layer should be sequential from left to right, such as $1, 2, ...k, k+1, k+2, ..., n$ (see Fig. 4.1(b)). In this way, all brand-based product categories will be mapped to a linear dimension[1]. Then, with the other two dimensions Transaction Amount and Transaction Time, CTT computation is transformed to a three-dimensional box aggregate problem.

However, our work does not convert product category hierarchy to a linear dimen-

[1]Based on the above transformation in Fig.3.1, let us assume that "*Canon SLR Digital Camera*" is assigned integer 10, "*Nikon SLR Digital Camera*" assigned 11 and "*Olympus SLR Digital Camera*" assigned 12, etc. In such a case, when performing a *"roll-up"* operation, i.e. the CTT query on the category of "SLR Digital Camera", it is equivalent to computing the aggregation result within range $[10, 12]$.

sion; rather, we regard CTT computation as an extended two-dimensional RA problem. That is because a problem arises when the product hierarchy updates.

Generally speaking, for each expansion (insertion) operation, the hierarchical structure is flexible. For example, as shown in Fig. 4.1(c), when a new product category is inserted into the product category hierarchy, a path is first searched from top (the product category root) to bottom (the brand-based product category) to locate the node where it is to be inserted. Then, the *C-value* of the new product category simply adds a two-digit number forming the *C-value8*. By contrast, in a mapped linear dimension, after each update or expansion, we have to recode and map new product category hierarchy. For example, the product categories coded with $k+2, k+3, ..., etc.$ as depicted in Fig. 4.1(c) will change to $k+3, k+4, ..., etc.$ as depicted in Fig. 4.1(b). More importantly, after the above mapping operations, the corresponding index for computing each seller's reputation profile also has to be regenerated. In practice, the product category hierarchy, such as UNSPSC and eCl@ss standards, is regularly updated. In such a case, the index for computing each seller's reputation profile has to be continuously regenerated. Considering a system with millions of sellers, this will lead to unnecessary overhead.

4.2 Preliminary Solutions for CTT Computation

This section proposes the preliminary solutions *eaR-tree*, *eaP-tree* and *eH-tree* for CTT computation which extend the existing approaches to a two-dimensional (2D) RA problem as introduced in Section 2.4.2. Note that the solutions proposed in this chapter are mainly based on extending the *aR-tree* [63, 104] and the *aP-tree* [135], since the others do not fully meet the requirements of CTT computation. The detailed summary of the limitations of them for CTT computation are given in the next chapter. Moreover, we introduce an additional new element, i.e. *element (d)*, into trust vector, which is termed as Transaction Proportion based Trust (TPT). TPT computes the trust value of a seller with regard to the proportion of transactions that are similar to the new

transaction amongst all past transactions. The introduction of TPT aims to illustrate the possibility of transaction context imbalance (see Section 3.1.3) in a forthcoming transaction. In the following, we first explain the meaning of TPT.

4.2.1 Transaction Proportion based Trust (TPT)

According to the definitions presented in Section 3.3.1, Transaction Proportion based Trust (TPT) is specifically designed to indicate the risk of context imbalance in a forthcoming transaction. The intent is to compute the trust level of the seller with regard to the proportion of past transactions in $Trans$ that are similar to the forthcoming transaction. As mentioned in Section 3.4.2.2, Wang and Lim propose a similar concept of *Global Weighted Trust* (GWT), which also takes into account the influence of the proportion of similar transactions [149]. The nature of the GWT is depicted as: 1) The GWT value is high if the transactions similar to $TR^{(t)}$ have a large proportion in $Trans$ and their ratings are high; 2) The GWT value is low if that proportion is small even if each corresponding transaction rating $r^{(t)}$ ($r^{(t)}$ is a short form of $r(TR^{(t)})$) is high. Then, in order to calculate transaction similarity, only the difference in transaction amounts is considered [149]. By contrast, TPT is a new element introduced in the trust vector which improves GWT. Being distinct to TIST, PCT and STAT, the value of TPT is *global* in the sense that it is based on all past transactions.

With a past transaction $TR^{(i)}$ in $Trans$ and a forthcoming transaction $TR^{(f)}$, in Section 3.3.3.1, we have defined the *transaction content similarity* S_{TC} between $TR^{(i)}$ and $TR^{(f)}$. More specifically, S_{TC} combines S_{TI} (the transaction item similarity) and S_{TA} (the transaction amount similarity). Table 4.1 lists some examples of transaction content similarity measures based on the product category hierarchy in Fig. 3.1.

Following the definition of S_{TC}, the TPT can be calculated as:

$$T_{TPT} = \frac{1}{m}\sum_{i=1}^{m}(r^{(i)} * S_{TC}(TR_i, TR_f)), \tag{4.1}$$

where m is the number of transactions in the past transaction set $Trans$. In the fol-

Table 4.1: Examples of transaction content similarity

Product Pair	depth d	S_{TI}	S_{TA}	S_{TC}
Canon EOS T3i Rebel ($700), Canon EOS T2i Rebel ($600)	7	0.99	0.92	0.96
Canon EOS T3i Rebel ($700), Canon A2200 Digital Camera ($150)	5	0.96	0.73	0.85
Canon EOS T3i Rebel ($700), Olympus Zuiko Macro Lens ($450)	2	0.66	0.86	0.76
Canon EOS T3i Rebel ($700), MacBook Pro Laptop ($900)	1	0.38	0.87	0.63
Canon EOS T3i Rebel ($700), AT&T SIM Card ($1)	1	0.38	0.32	0.35

lowing subsections, based on our proposed tree structures for computing CTT values, we also introduce how to compute TPT values in Equation (4.1).

4.2.2 The eaR-tree

The *eaR-tree* extends the *aR-tree* [63, 104], and it has three types of nodes: *R-node* (Rn), *I-node* (In) and *L-node* (Ln), each of which can have multiple records depending on the node capacity. As shown in Fig. 4.2, one *eaR-tree* consists of a *C-tree* and multiple *aR-trees* that are external to the *C-tree*. The *C-tree* consists of R-nodes, and an *aR-tree* consists of I-nodes and L-nodes. Here we need to emphasise that Fig. 4.2 essentially shows the general structure of our proposed approaches for CTT computation. Basically, the idea of extending *C-tree* with multiple subtrees for CTT computation is applied to three approaches proposed in this chapter as well as the index scheme *CMK-tree* (an optimal solution for efficient CTT computation), which will be proposed in the next chapter.

4.2.2.1 The C-tree (Product Category Tree)

Following Step 2 in the extension process described in Section 4.1.2, each record in an R-node Rn_i has the form

$$< \textit{C-value}, [ta_{min}, ta_{max}], [t_{min}, t_{max}], count\_r, sum\_r, pointer >,$$

where the *C-value* denotes the unique id of the product category within the product category hierarchy; $[ta_{min}, ta_{max}]$ and $[t_{min}, t_{max}]$ are the transaction amount range and the transaction time range of all the transactions belonging to the current product

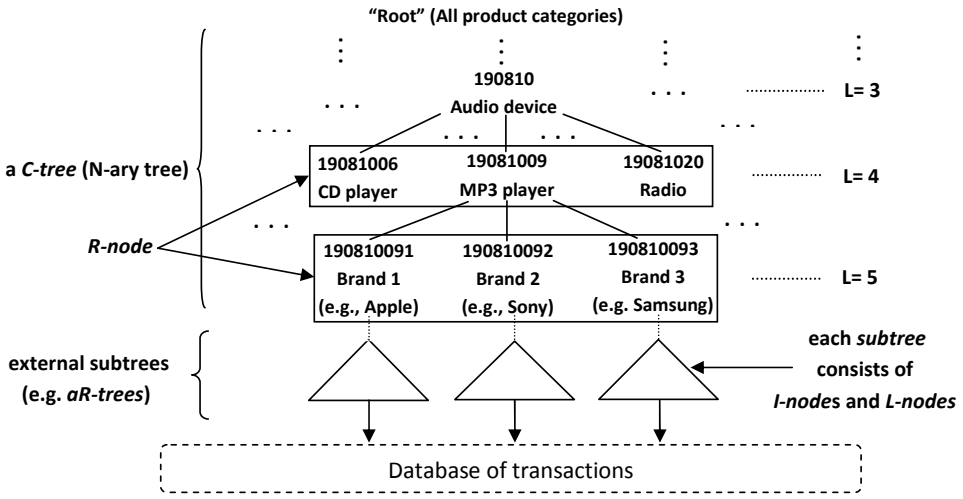


Figure 4.2: The general structure of our proposed approaches for CTT computation

category; $count\_r$ and sum_r denote the aggregates over these transactions; $pointer$ points to its child, which is an R-node or an I-node. Therefore, an R-node contains multiple product categories represented by corresponding records, and these product categories are on the same layer within the product category hierarchy. All R-nodes form an *N-ary tree*, and we term it as a *C-tree (product **C**ategory-tree)*.

4.2.2.2 The Other Node Structures

Apart from R-nodes, each record I_i in an I-node contains $< [ta_{min}, ta_{max}], [t_{min}, t_{max}]$, $count\_r$, sum_r, $pointer >$, where $pointer$ points to its child node, which is another I-node or an L-node. Each record L_i in an L-node contains $< price$, $time$, $count\_r$, sum_r, $pointer >$, where $pointer$ points to a record of a transaction in the database.

Essentially, each I-node is equivalent to an internal node and each L-node is equivalent to a leaf node in an *aR-tree* [104], respectively. However, in an *eaR-tree*, the structure of each record L_i in an L-node is different from the leaf record in the original *aR-tree*. That is because, for the two-dimensional RA problem, one point in space represents one object only (e.g., a car). Therefore, each leaf record in an *aR-tree* does not need to maintain the aggregate of objects since they are always 1. By contrast, in the *eaR-tree*, we introduce $count\_r$ and sum_r in each record of L-node to aggregate the

ratings derived from the repeated transactions on a given day selling the same product with the same price. On one hand, the structure of *eaR-tree* guarantees that each specific product can be indexed so as to compute its trust value (i.e., TIST). On the other hand, the added data fields $count\_r$ and sum_r allow operations on the index of product space rather than the whole transaction space on each day, which can reduce space consumption and the number of accessed nodes for answering a CTT query.

4.2.2.3 The Construction of an eaR-tree

Before inserting the data of a newly occurred transaction into an *eaR-tree*, a path is first searched from the product category root to the brand-based layer in the product category hierarchy based on the *C-hrchy* of the transaction. If the transaction belongs to a new product category on which the seller has no prior transactions, the new record in an R-node is generated for this product category as well as its corresponding sub-categories. Otherwise, the set of ranges and aggregates (i.e., $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$, sum_r) in each record along the path are updated accordingly. After that, the insertion operations should be performed in a subtree pointed by the corresponding record in an R-node in the brand-based product category layer.

Fig. 4.3 depicts the construction of an *eaR-tree* after inserting the data for three transactions trading different products, but these products all belong to the same product category with the same *C-hrchy*. In addition, the transaction information $< ta_i$ (transaction amount), t_i (transaction time), r_i (rating) $>$ of these three transactions is $< 5, 1, 1 >$, $< 15, 1, 1 >$ and $< 10, 2, 1 >$. The node I denotes an I-node, and $count\_r$ and sum_r in one record of I ($< [5, 15], [1, 1], 3, 3 >$) maintain the sum of all the aggregates $count\_r$ and sum_r respectively in the L-node A. Each record in the L-node A has the fields $count\_r$ and sum_r for the inserted multiple transactions that happened on a given day selling the same product.

For the split of an R-node, the corresponding new record in an R-node is generated, which contains a *C-value* to represent the parent of current R-node. Meanwhile, $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$ and sum_r in the new record also need to be up-

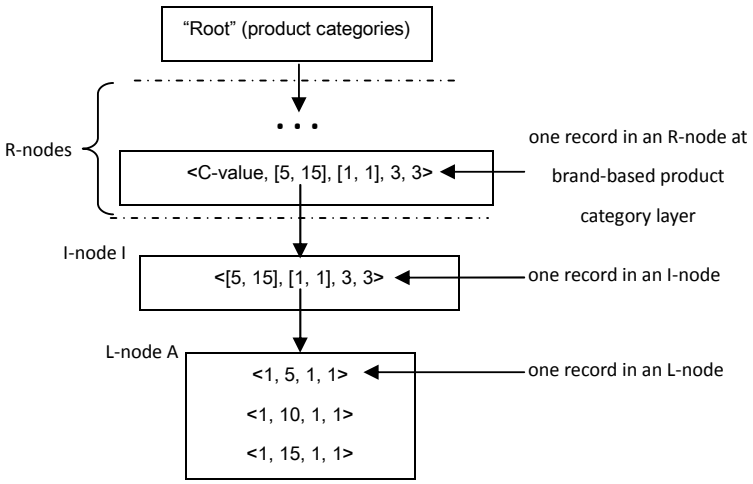


Figure 4.3: The construction of an *eaR-tree*

dated to reflect the ranges and aggregates of its child node. Then, the generated new record is added to the corresponding layer in the product category hierarchy. The insertion and split of either an L-node or an I-node in the *eaR-tree* is the same as the *aR-tree*, and more details can be found in [104].

4.2.2.4 Performance Analysis

To compute each of TIST, TPT, PCT and STAT to answer a CTT query, the search is first performed in a *C-tree*.

The queries on TIST, PCT and STAT: To answer a query on PCT: <product-category: "*DSLR Camera*"; price-range: "$500-$900"; time-range: from "*the latest 6 months*">, all sub-categories under "*DSLR Camera*" (see Fig. 3.1) are first considered. As each record in R-node contains a transaction amount range ($[ta_{min}, ta_{max}]$) and a transaction time range ($[t_{min}, t_{max}]$), three different cases arise, as follows:

> ***Case 1:*** *If a CTT query on the transaction amount range and the transaction time range falls into the region surrounded by* $[ta_{min}, ta_{max}]$ *and* $[t_{min}, t_{max}]$*, then it is not necessary to search its child node (sub-categories). Instead,* $count\_r$ *and* sum_r *in that record can be used directly.*

Case 2: *If a CTT query on the transaction amount range and the transaction time range are out of the region surrounded by* $[ta_{min}, ta_{max}]$ *and* $[t_{min}, t_{max}]$*, then it is not necessary to search its child node (sub-categories) either.*

Case 3: *If a CTT query on the transaction amount range and the transaction time range overlaps with the region surrounded by* $[ta_{min}, ta_{max}]$ *and* $[t_{min}, t_{max}]$*, then the search iteratively executes from Case 1 to Case 3 in its descendants until reaching the layer of I-nodes. Taking each reached I-node as a root, its child I-nodes and L-nodes will be searched.*

For Case 3, when searching a subtree containing I-nodes and L-nodes, as in the *aR-tree*, the query cost in the *eaR-tree* depends on the size of the query region formed by the transaction amount range and transaction time range. The larger the query region, the more overlapped MBRs (see Fig. 2.4(b)).

Compared with the computation of PCT, the search process for computing TIST and STAT is almost the same. The only difference is that in the computation of TIST, there is a need to further search the actual records in the database. In the computation of STAT, Case 1 to Case 3 need to be executed in all the categories of the products sold by the seller. For the sake of simplicity, we use the computation of PCT as the example to analyse these approaches.

The queries on TPT: Next, based on *C-tree* (Product Category Tree), we explain the search process for answering (computing) the queries on TPT (see subsection 4.2.1). Assume that all the m transactions in the past transaction set $Trans$ in Eq. (4.1) belongs to k brand-based product categories (see Fig. 3.1). As illustrated above, each record in R-node maintains the transaction amount range $[ta_{min}, ta_{max}]$, the number of ratings $count\_r$ and the sum of ratings sum_r over all these transactions in the corresponding product category. Therefore, the number m also equals $\sum_{i=1}^{k} count\_r_i$. Furthermore, given a forthcoming transaction $TR^{(f)}$, the following transformation is proposed so as to compute $\sum_{i=1}^{m}(r^{(i)} * S_{TC}(TR^{(i)}, TR^{(f)}))$ in Eq. (4.1).

(1) To calculate the transaction amount similarity $S_{TA}(TR^{(i)}, TR^{(f)})$ (denoted as

S^i_{TA}), we introduce an *approximate calculation method*. Instead of using the transaction amount ta in each transaction $TR^{(i)}$, we set the mean value $\frac{ta_{min}+ta_{max}}{2}$ of $[ta_{min}, ta_{max}]$ as the input for computing S^i_{TA}. $[ta_{min}, ta_{max}]$ represents the transaction amount range of a brand-based product category, to which the transaction $TR^{(i)}$ belongs.

(2) The way to calculate the transaction item similarity $S_{TI}(TR^{(i)}, TR^{(f)})$ (denoted as S^i_{TI}) is first to locate the deepest common ancestor between the transaction item in $TR^{(f)}$ and the brand-based product category, to which $TR^{(i)}$ belongs. Then, the formulas proposed in Eq. (3.1) can be used to calculate S_{TI}. In such a case, the original Eq. (4.1) is converted as:

$$T_{TPT} = \frac{\sum_{i=1}^{k}(sum\_r_i * (\frac{S^i_{TI}+S^i_{TA}}{2}))}{\sum_{i=1}^{k} count\_r_i}, \tag{4.2}$$

where sum_r_i is the sum of ratings for certain brand-based product category.

Obviously, the search needs to be performed from top to bottom in the whole product category hierarchy, i.e., *C-tree*, for computing T_{TPT}. Its time complexity is $O(k * d)$, where k is the number of brand-based product categories and d is the depth from the product category root (top) to a brand-based product category (bottom). As the *eaR-tree*, *eaP-tree*, *eH-tree* and *CMK-tree* (to be introduced in next chapter) have the same *C-tree* structure, they have the same time complexity in answering a TPT query. Therefore, in the following, the method for computing TPT values will not be described again; rather, we will focus on introducing their structures.

4.2.3 The eaP-tree

4.2.3.1 The Structure of an eaP-tree

The *eaP-tree* adopts an *aP-tree* [135] as each subtree to extend *C-tree*, thus it has the same structure as the *eaR-tree* for each record R_i in R-nodes. But each record I_i in an

I-node has a different structure as $< ta_{min}$, $[t_{start}, t_{end})$, $count\_r$, sum_r, $pointer >$, where $count\_r$ and sum_r denote the number of ratings and the corresponding rating sum in its subtree that is alive in $[t_{start}, t_{end})$. The term ta_{min} indicates all transactions in its children with a transaction amount no less than ta_{min}. The term $[t_{start}, t_{end})$ records the time period within which the corresponding transactions occurred. If t_{end} is "*", it means the records are alive; otherwise they have died before t_{end} [135]. Each record L_i in an L-node also has the structure $< price$, $time$, $count\_r$, sum_r, $pointer >$. In the following subsection, we further explain that, for an ***eaP-tree***, the additional data fields $count\_r$ and sum_r in each record of L-node cannot essentially aggregate repeated transactions that occurred on a day.

4.2.3.2 The Construction of an eaP-tree

The main difference between an *eaP-tree* and an *eaR-tree* lies in the construction of the subtrees containing I-nodes and L-nodes that are external to a *C-tree*. Thus we briefly introduce the construction of a subtree in the *eaP-tree* in this subsection. To some extent, it is similar to constructing an *aP-tree* [135].

In Fig. 4.4(a), three more transactions with x-y coordinates $(1, 25), (1, 35), (1, 45)$ (day represented by the x-axis and price represented by the y-axis) are inserted into the L-node A, and we assume the node capacity is 6. Also, these transactions represent different products, but they all belong to the same product category with the same *C-hrchy*. In an *eaP-tree* (see Fig. 4.4(a)), the record $< 5, [1, *), 6, sum\_r, A >$ in I-node I is generated to indicate that there are 6 records in the L-node A, which are alive from $x_{start} = 1$ and their transaction amounts are at least 5. The term sum_r in each record of an I-node is the sum of sum_r for all the records in the L-node A. Note that the fields $count\_r$ and sum_r in each record of L-node are ignored, and we use $count\_r$ in each record of I-node as the example to illustrate the aggregation process.

Fig. 4.4(b) illustrates when a new record $(2, 55)$ with a different transaction day (i.e. x-coordinate) is inserted into A, the record $< 5, [1, *), 6, sum\_r, A >$ dies with its x_{end} modified to 2. Since the L-node A is overflown after inserting the new record, it

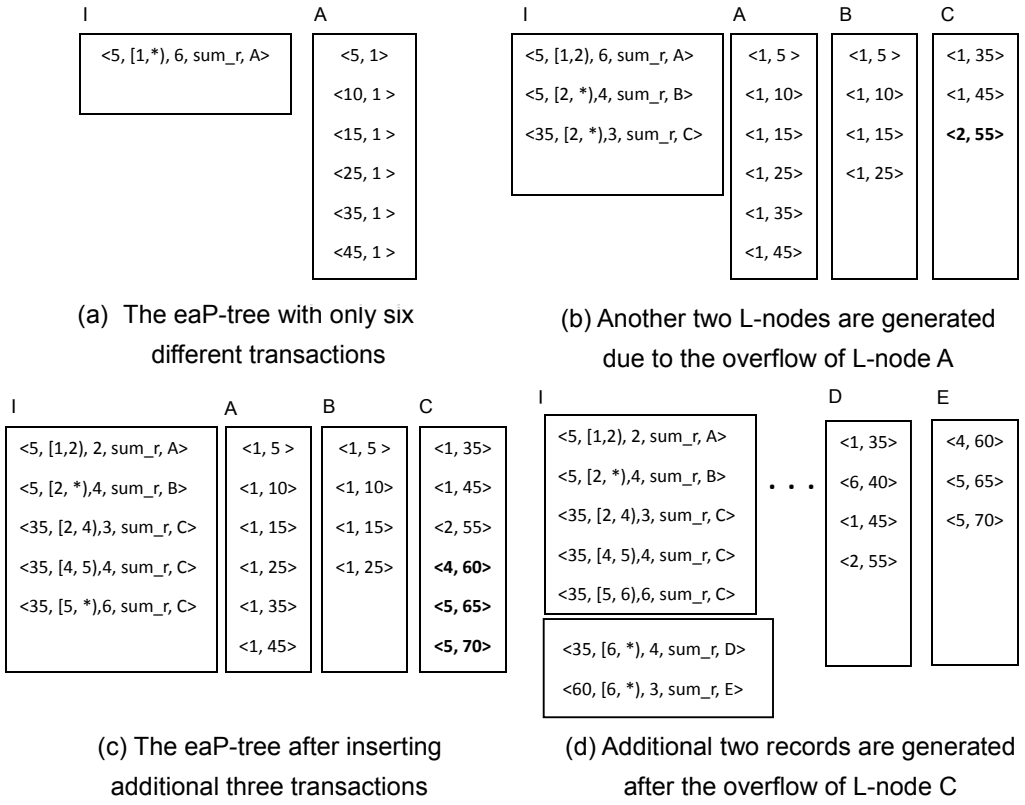


Figure 4.4: The construction of an *eaP-tree*

splits into two L-nodes B and C as shown in Fig. 4.4(b). The additional two records $< 5, [2, *), 4, sum\_r, B >$ and $< 35, [2, *), 3, sum\_r, C >$ are generated with $x_{start} = 2$, which points to the L-nodes B and C, respectively. Meanwhile, the whole process is also accompanied by the split of transaction amount (i.e. the y-coordinates).

When inserting a newly occurred transaction into a subtree that is external to the *C-tree*, the operation is to *iteratively search the live record* $[x_{start}, *)$ *whose y-coordinate (i.e., transaction amount) is the largest among all the records in I-nodes*. This process is the same as the *aP-tree* [135]. For example, Fig. 4.4(c) illustrates the *eaP-tree* structure after inserting three more records $(4, 60), (5, 65), (5, 70)$. When the record $(4, 60)$ with a different transaction time is inserted, the record $< 35, [2, *), 3, sum\_r, C >$ in an I-node dies, and a new record is duplicated from $< 35, [2, *), 3, sum\_r, C >$ with its x_{start} set to 4 and its $count\_r$ incremented by 1 (see the I-node I in Fig. 4.4(c)). Fig. 4.4(d) illustrates that two L-nodes D and E are generated after inserting the record

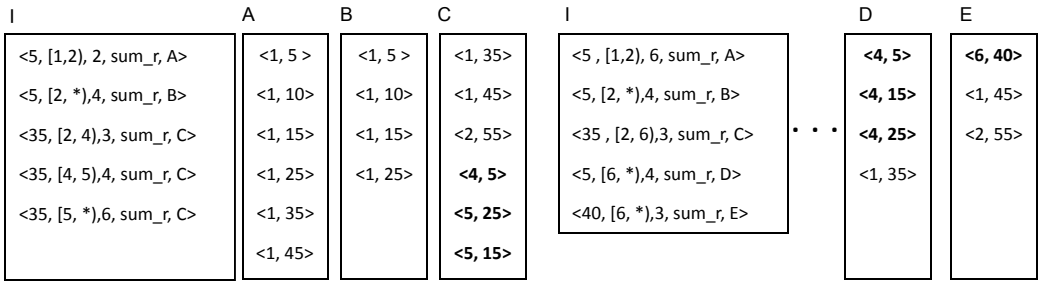
$(6, 40)$. Thus, the insertion of two additional records leads to the overflown of the I-node I overflown, and thus two new I-nodes I_1 and I_2 (I_2 contains all the alive records) are generated from the spilt of the original I-node I. In addition, like the *aP-tree*, the *eaP-tree* has only a single parameter svo that denotes the threshold of *strong version overflow*. The svo occurs when the number of alive records in a new node exceeds $b * svo$ (b is node capacity). More details about svo can be found in [135].

4.2.3.3 Performance Analysis

When answering a CTT query on PCT, the search in an *eaP-tree* still starts from locating the corresponding product category in the product category hierarchy according to three different cases (see subsection 4.2.2.4). For Case 3, unlike the *eaR-tree*, two VRAs are computed in the specific transaction amount range and transaction time range as in an *aP-tree* [135]. However, the *eaP-tree* has two obvious drawbacks when dealing with CTT computation in e-commerce environments.

(a) **Fixed right border** leads to a large number of nodes to be accessed in the *eaP-tree*. One of the important characteristics for a CTT query is that the transaction time range should start from a previous point and end at the time point "*now*". Hence, the right border VRA query is fixed to the "*now*" (i.e. the end time) in the CTT computation. In such a case, all the alive records (i.e. the records with x_{end} ="*") need to be checked regarding whether their transaction amount ranges intersect with the query range $[ta_1, ta_2]$.

(b) **Insertion algorithm** leads to a large number of nodes to be accessed in an *eaP-tree*. In subsection 4.2.3.2, we have mentioned that the way to insert a newly occurred transaction in an *eaP-tree* is to iteratively find the live record whose y-coordinate (i.e., Transaction Amount) is the largest among all the records of I-nodes in a specific brand-based product category. An important observation is that the efficiency of the *eaP-tree* depends on whether the transaction amounts (i.e., the y-coordinates of all records) are inserted in order. As the example



(a) Three different transactions inserted to eaP-tree

(b) Two different records are generated in I-node after inserting four transactions

Figure 4.5: Another example for the construction of an *eaP-tree*

shown in Fig. 4.4, the transaction amounts are also in ascending order. In such a case, it is easy to infer the range of transaction amounts in the subtree pointed by a record of I-node. For example, in Fig. 4.4(d), based on the records $< 5, [2, *), 4, sum\_r, B >$ and $< 35, [6, *), 4, sum\_r, D >$, the transaction amount range $[5, 35)$ in L-node B can be inferred.

Regarding our targeted CTT computation problem, it is difficult to satisfy the requirement that transactions are inserted in the order of transaction amounts. In e-commerce environments, it is quite common that a seller has multiple transactions representing the sale of the same product on a given day or at different transaction time. The range of transaction amounts in the subtree pointed by a record of I-node is difficult to infer as indicated above. As a result, there are a large number of nodes to be accessed in an *eaP-tree*. For instance, in Fig. 4.5(a), if the inserted records are $(4, 5), (5, 25), (5, 15), (6, 40)$ instead of $(4, 60), (5, 65), (5, 70), (6, 40)$ as in Fig. 4.4(c), the additional two records are $< 5, [6, *), 4, sum\_r, D >$ and $< 40, [6, *), 3, sum\_r, E >$ (see Fig. 4.5(b)), rather than $< 35, [6, *), 4, sum\_r, D >$ and $< 60, [6, *), 3, sum\_r, E >$ as shown in Fig. 4.4(d). In such a case, the transaction amount range in the L-node B cannot be inferred based on the records $< 5, [2, *), 4, sum\_r, B >$ and $< 5, [6, *), 4, sum\_r, D >$. Assume that a VRA query is $6 : [20, 50]$, then all the

L-nodes pointed by three alive records in an I-node should be visited.

As pointed out in [168], the insertion algorithm leads to a problem of *aP-tree* that operations on the index are performed on all the objects in order to solve RA problem in two-dimensional space. Likewise, the *eaP-tree* indexes each transaction separately and cannot essentially aggregate repeated transactions that occurred on a day. Therefore, the data field $count\_r$ in each record in L-nodes of the *eaP-tree* is always 1 and the corresponding sum_r is a single rating. In fact, the operation on the index of the whole transaction space is an important reason, leading to inferior performance for an *eaP-tree* when compared to an *eaR-tree*. This is also illustrated in our experiments to be introduced in Section 4.3.

4.2.4 The eH-tree

To overcome the disadvantages of the *eaP-tree*, we propose another structure, *eH-tree* which is an extended hybrid structure.

4.2.4.1 The Structure of an eH-tree

Basically, the *eH-tree* also extends a *C-tree*, where a subtree is generated and pointed by each brand-based product category. Therefore, it still has the same structure for each record in R-nodes. But the *eH-tree* has a different structure in each subtree. In particular, instead of using only a key (the field ta_{min}) as each record in I-nodes of the *eaP-tree*, the field $[ta_{min}, ta_{max}]$ is used to record the minimum and the maximum of transaction amounts respectively in its subtree. In order to differentiate from the original *aP-tree*, we denote the new structure as *aP$^+$-tree*. Although each record introduces one additional data field, it can reduce the number of nodes to be accessed in answering a CTT query. This advantage is further explained in the following subsection.

In addition to the *aP$^+$-tree*, another *B$^+$-tree* [15] is constructed in the separate transaction amount space. Meanwhile, the *B$^+$-tree* contains additional fields to store the aggregates $count\_r$ and sum_r, forming an *aB$^+$-tree*. Each record in an *aB$^+$-tree*

has the structure $< ta_{min}$, $count\_r$, sum_r, $pointer >$ where $pointer$ points to its child. But for a record in the leaf node of the *aB$^+$-tree*, $pointer$ points to the record of a transaction in the database. Notice that, unlike a *B$^+$-tree*, the size of an *aB$^+$-tree* depends on the number of distinct products in a brand-based product category rather than the number of distinct transaction amounts, as some different products can be sold with the same price. Also, as an extended hybrid structure, the records of R-nodes in the brand-based product category layer are modified with two pointers.

4.2.4.2 The Construction of an eH-tree

When inserting the data of a newly occurred transaction into an *eH-tree*, as illustrated in subsection 4.2.2.3, the set of ranges and aggregates (i.e., $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$, sum_r) in each record along the path from the product category root to the brand-based product category layer in the *C-tree* will be updated first. After that, both the *aP$^+$-tree* and the *aB$^+$-tree* in the corresponding subtree that is external to the *C-tree* are also updated.

The process of the *aP$^+$-tree* construction is similar to that of the *aP-tree* [135] except for recording the minimal and the maximal of the transaction amount in its subtree during insertion and split operations. While the process of an *aB$^+$-tree* construction is similar to that of a *B$^+$-tree* [15], different products are inserted as different records into an *aB$^+$-tree*, even if they have the same price. Fig. 4.6 depicts the construction of an *eH-tree*, after inserting three transactions as introduced in subsection 4.2.2.3. Given a CTT query, in the *eH-tree*, there is a need to search the *aP$^+$-tree* for the left border VRA and the *aB$^+$-tree* for the right border VRA, respectively.

4.2.4.3 Performance Analysis

When searching in a subtree that is external to the product category hierarchy, the *eH-tree* improves *eaP-tree* in two aspects for CTT queries on PCT.

(a) **Fewer nodes will be accessed for the left border VRA.**

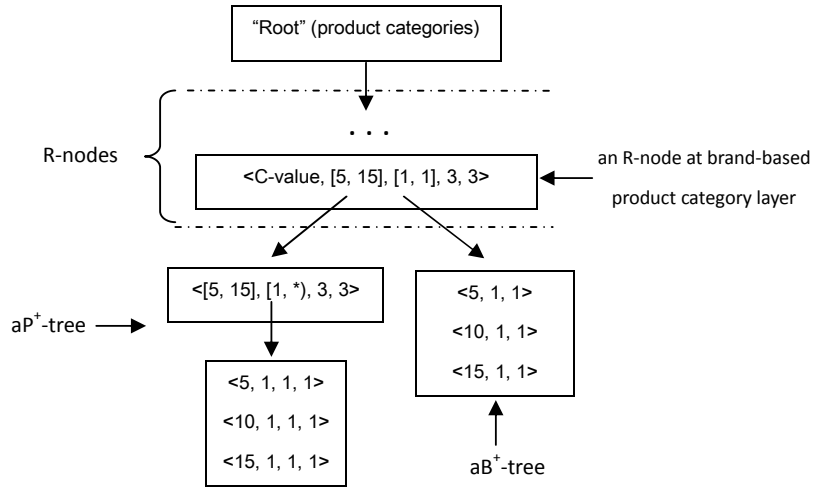


Figure 4.6: The contraction of an *eH-tree*

In an *eH-tree*, the number of accessed nodes can be reduced in some cases when computing the left border VRA. For example, in Fig. 4.5(a), if the inserted records are $(5, 4), (25, 5), (15, 5), (40, 6)$, the additional two records in an I-node will be $< 5, [6, *), 4, sum\_r, D >$ and $< 40, [6, *), 3, sum\_r, E >$ (see Fig. 4.5(b)) in *eaP-tree*. By contrast, in the *eH-tree*, the generated two records are $< [5, 35], [6, *), 4, sum\_r, D >$ and $< [40, 55], [6, *), 3, sum\_r, E >$. As a result, for the same VRA query $6 : [5, 30]$, the L-node pointed by $< [5, 25], [2, *), 4, sum\_r, B >$ is not accessed as it is covered by $[5, 30]$, while it will be visited in an *eaP-tree*.

(b) **Search is performed in the fully ordered transaction amount space for the right border VRA query.**

As stated before, each CTT query on the transaction time range refers to the latest time period, which starts from a previous point and end at the point "*now*" (i.e., the end time). If we take the points depicted in Fig. 2.5(b) as transactions, an important observation is that all the generated intervals by the transactions intersect with right border regardless of their transaction time. In order to further reduce the number of accessed nodes, in *eH-tree*, we construct another *aB*$^+$*-tree* in the separate transaction amount space. The same as the *B*$^+$*-tree* [15], all the

transaction information in an *aB$^+$-tree* is kept sorted based on their transaction amount. Note that each generated *aB$^+$-tree* is based on the transactions in a brand-based product category. In such a way, the computation of right border VRA can be converted to searching in the fully ordered transaction amount space and thus efficiency is improved.

4.3 Experiments on Preliminary Solutions

This section describes the experiments conducted on two eBay datasets and three large-scale synthetic datasets, which aim to evaluate the efficiency difference of three proposed approaches *eaR-tree*, *eaP-tree* and *eH-tree*. Here, we need to emphasize that the advantages of our proposed trust vector based approach have already been studied both analytically and empirically in Chapter 3.

4.3.1 Datasets

4.3.1.1 eBay Datasets

With eBay APIs[2], we have obtained detailed feedback and transaction data for up to 90 days of selected sellers. In seller selection, we first chose a number of popular products, and the sellers selling them with the largest number of reviews. With them, we finally selected two sellers S_1 and S_2 who had totally around 12,000 transactions (approx. 133 transactions per day) and 4,000 transactions (approx. 44 transactions per day) respectively within 90 days.

While the products sold by S_1 and S_2 exist in multiple product categories, most products are in the category '*Information, Communication and Media technology*' (see Fig. 3.1). Specifically, the products sold by S_1 include *MP3 players*, *Notebooks*, *Digital Cameras*, *CD & DVD players*, *LCD monitors* etc, and the products sold by S_2 include *Digital Cameras*, *Video Cameras*, *Camera & Photo Accessories*, *Printers*, *S-*

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

martphones etc. The selection of S_1 and S_2 allows performing both *"drill-down"* and *"roll-up"* operations in the product category hierarchy (see Fig. 3.1) when doing finer-grained analysis on a seller's transaction reputation.

4.3.1.2 Synthetic Datasets

Our experiments are also conducted on three synthetic datasets based on S_1. The three synthetic datasets can be categorized into two types:

1) In the first type of synthetic datasets (denoted as $\boldsymbol{SD_1}$), we generated the transaction data for a given day at 10 times as much as that of S_1, and then duplicated these transactions for 90 days to form the transaction data for a 12 month period. Thus, there are about $480,000$ transactions in total in $\boldsymbol{SD_1}$. The first type of synthetic dataset guarantees that each product sold in a year has the same proportion of occurrence as that in the eBay real dataset.

2) The second type of synthetic datasets also contains transaction data at 10 times the rate as the seller S_1 on a given day. It contains 12 months transaction data. However, each transaction in the second type synthetic dataset is randomly selected from S_1's eBay dataset with 12,000 transactions in 90 days. Moreover, in this dataset type, we generate two datasets $\boldsymbol{SD_2}$ and $\boldsymbol{SD_3}$. In each of them, the proportion of transactions related to sales of a product or a product category is unlikely to be the same as the eBay real dataset.

The purpose of generating $\boldsymbol{SD_1}$, $\boldsymbol{SD_2}$ and $\boldsymbol{SD_3}$ is to test the performance of our proposed approaches under the circumstances of exceptionally large volumes of transactions both on a single day and over a longer time period (e.g., 12 months).

4.3.2 The Experimental Results

4.3.2.1 Experiment Setup

We compare the proposed *eaR-tree*, *eaP-tree* and *eH-tree* approaches in the experiments. All the trees have the same page size of 512 bytes. The svo (strong version

overflow) for both the *eaP-tree* and the *eH-tree* is set to 5/6. The svo is used to control the number of alive entries in an internal node after splitting [17]. In addition, each of the *eaR-tree*, the *eaP-tree* and the *eH-tree* is implemented using C++ running on a Lenovo Y560 laptop with an Intel Core i5 CPU (2.20GHz), 2GB RAM, Windows 7 Professional operation system and MySql 5.1.35 relational database.

To evaluate the performance of our proposed approaches, we measure the number of accessed nodes for computing the value of *Product Category based Trust* (PCT) and the value of *Similar Transaction Amount based Trust* (STAT) proposed in trust vector (see subsection 3.3.2). Computing the trust value of a specific product (TIST) has a procedure very similar to the computation of PCT. The only difference is that in the computation of TIST, it needs to further search the actual record in the database. Thus we will not test for TIST. Similarly, we will not compare the performance of the three approaches in computing the TPT values, as they all have the same time complexity $O(k * d)$ (see subsection 4.2.2.4).

As stated before, the products sold by S_1 and S_2 exist in multiple product categories. For S_1, we select a product category "*Apple MP3 player (iPod)*" at layer 5 in the product category hierarchy with numerous transactions. Moreover, S_1 also has lots of transactions at both layer 3 (i.e., "*Audio device*") and layer 2 (i.e., "*Multimedia, Entertainment technology*"), in the product category hierarchy (see Fig. 3.1). For S_2, we select a product category "*DSLR Camera*" at layer 6, "*Digital Camera*" at layer 5 and "*Photo, video technology*" at layer 2 respectively in the product category hierarchy (also see Fig. 3.1).

4.3.2.2 Results of eBay datasets

The first set of experiments is conducted on the real eBay datasets containing two different sellers as described above. We evaluated our three proposed approaches in the following two scenarios:

Scenario 1: Buyer B_1 plans to buy an "*Apple mc526ll/a iPod nano 16GB*" for about $\$150$ from the seller S_1. In order to measure the number of accessed nodes,

when computing the value of PCT (*Product Category based Trust*), we assume that B_1 specifies a transaction amount range of [\$100,\$200], and a time range "*the latest one month*", "*the latest two months*", and "*the latest three months*", respectively. Note that, for simplicity, all the changes of CTT queries on the time dimension will be the same as the above for computing both PCT and STAT. Also, we assume B_1 adjusts the product category to layers 5, 3 and 2 respectively for computing PCT.

To measure the number of accessed nodes for computing the value of STAT (*Similar Transaction Amount based Trust*), we assume B_1 also plans to buy another product "*Samsung Galaxy tab 16GB Android Tablet*" for about $\$400$ from seller S_1. We measured the number of accessed nodes at each of four possible transaction amount ranges. They are [\$350,\$450] (difference = $\$100$), [\$300,\$500] (difference = $\$200$), [\$300,\$600] (difference = $\$300$) and [\$250,\$650] (difference = $\$400$). Note that the difference in price, instead of the price itself, may affect the number of accessed nodes. All parameters set in Scenario 1 for computing PCT and STAT are summarised in Table 4.2 and the experimental results of Scenario 1 are plotted in Fig. 6.2.

Scenario 2: The buyer B_2 plans to buy a "*Canon EOS T3i Digital Camera*" at a price of about $\$670$ from the seller S_2. B_2 specifies a transaction amount range of [\$600,\$700] for computing the value of PCT (*Product Category based Trust*), and adjusts the product category to layers 6, 5 and 2 respectively for computing PCT. The four possible transaction amount ranges for computing STAT (*Similar Transaction Amount based Trust*) are [\$600,\$700], [\$500,\$700], [\$500,\$800] and [\$500,\$900]. All parameters set in Scenario 2 for computing PCT and STAT are summarised in Table 4.3 and the experimental results of Scenario 2 are plotted in Fig. 6.3.

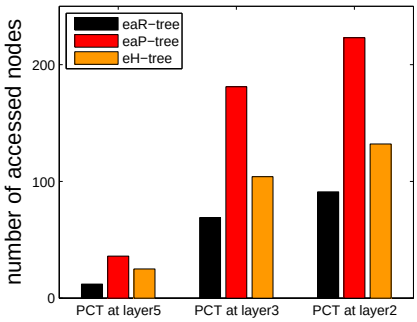
We adjusted the product category to layer 3 and layer 2 respectively in queries to measure the number of accessed nodes for computing PCT, with the same price range and time range as the previous experiment, and the results are also plotted from Fig. 6.2(a) to 6.2(c). In addition, Fig. 6.2(d) to 6.2(f) show the numbers of accessed nodes for computing STAT at a specific price range and a specific time range. Fig. 6.3 illustrates the three approaches operate on the data of another seller S_2.

Table 4.2: The parameters set in Scenario 1 for computing PCT and STAT

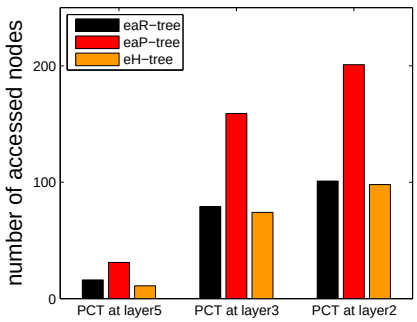
	Product Category	Transaction Amount Range	Transaction Time Range
PCT (*Product Category based Trust*)	*Apple MP3 player (iPod)* (layer 5) *Audio device* (layer 3) *Multimedia, Entertainment technology* (layer 2)	[$100,$200]	"*the latest one month*" "*the latest two months*" "*the latest three months*"
STAT (*Similar Transaction Amount based Trust*)	NO	[$350,$450] (difference=100) [$300,$500] (difference=200) [$300,$600] (difference=300) [$250,$650] (difference=400)	"*the latest one month*" "*the latest two months*" "*the latest three months*"

Table 4.3: The parameters set in Scenario 2 for computing PCT and STAT

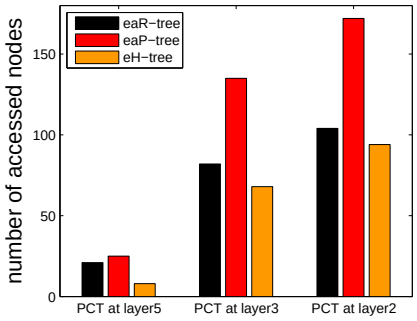
	Product Category	Transaction Amount Range	Transaction Time Range
PCT (*Product Category based Trust*)	*DSLR Camera* (layer 6) '*Digital Camera*' (layer 5) '*Photo, video technology*' (layer 2)	[$600,$700]	"*the latest one month*" "*the latest two months*" "*the latest three months*"
STAT (*Similar Transaction Amount based Trust*)	NO	[$600,$700] (difference=100) [$500,$700] (difference=200) [$500,$800] (difference=300) [$500,$900] (difference=400)	"*the latest one month*" "*the latest two months*" "*the latest three months*"



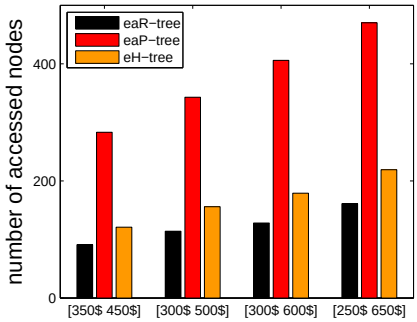
(a) The computation of PCT at different layers in one month



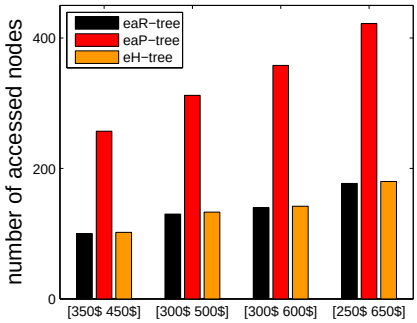
(b) The computation of PCT at different layers in two months



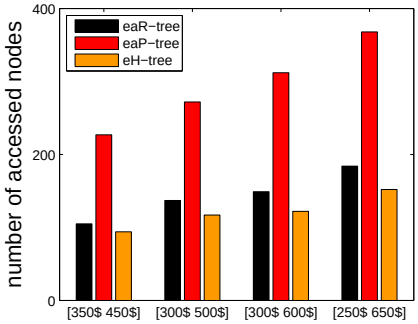
(c) The computation of PCT at different layers in three months



(d) The computation of STAT in one month

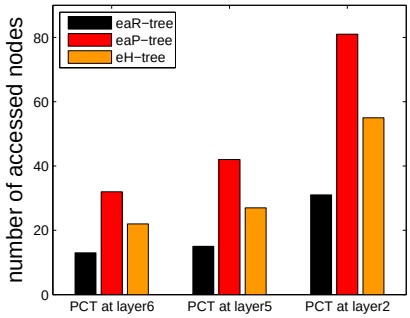


(e) The computation of STAT in two months

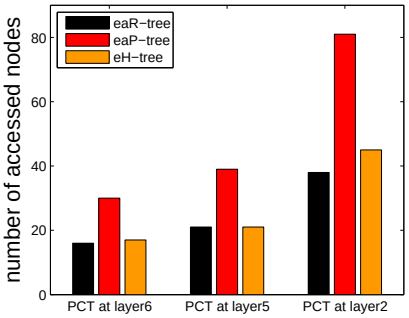


(f) The computation of STAT in three months

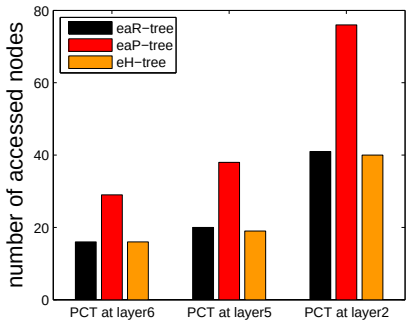
Figure 4.7: The performance of three proposed approaches in Scenario 1 (S_1, eBay dataset)



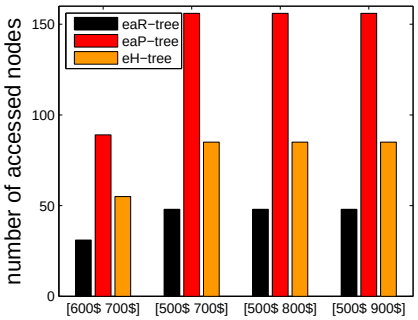
(a) The computation of PCT at different layers in one month



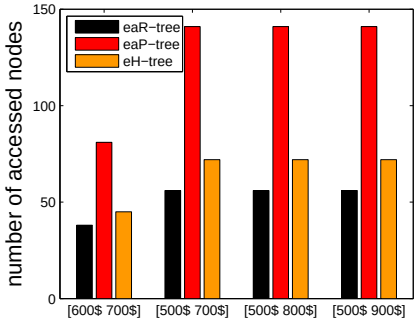
(b) The computation of PCT at different layers in two months



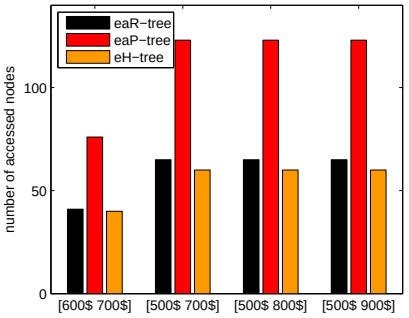
(c) The computation of PCT at different layers in three months



(d) The computation of STAT in one month



(e) The computation of STAT in two months



(f) The computation of STAT in three months

Figure 4.8: The performance of three proposed approaches in Scenario 2 (S_2, eBay dataset)

4.3.2.3 Results on synthetic datasets

We also conducted the same experiments on the three synthetic datasets $\boldsymbol{SD_1}$, $\boldsymbol{SD_2}$ and $\boldsymbol{SD_3}$, which contain transactions distributed in 12 months.

For computing the PCT and STAT, we assume that two buyers B_1 and B_2 both specify the same time range "*the latest 3 months*", "*the latest 6 months*", and "*the latest 12 months*", respectively. The results are plotted from Fig. 4.9 to Fig. 4.11.

4.3.3 Discussions of Results

From our experimental results, we can observe:

(1) Compared with the *eaR-tree*, the performance of the *eaP-tree* and the performance of the *eH-tree* all show a different trend corresponding to a buyer's CTT query. From Fig. 6.2(a) to 6.2(c), we can see that when the time range in a query increases (i.e., the left border shifts to the left), the number of accessed nodes in an *eaP-tree* decreases. As stated before, the right border is fixed to "*now*" in CTT computation. Correspondingly, in the *eaP-tree*, the number of accessed nodes is also fixed for computing the right border VRA. However, when the left border shifts to the left, the number of accessed nodes for computing the left border VRA decreases. This is because the children of all the records in I-nodes, whose starting time is later than the left border of the time range, will not be visited. If the time range in a query covers the whole time period, in such a case, only the right border VRA needs to be computed for answering a CTT query.

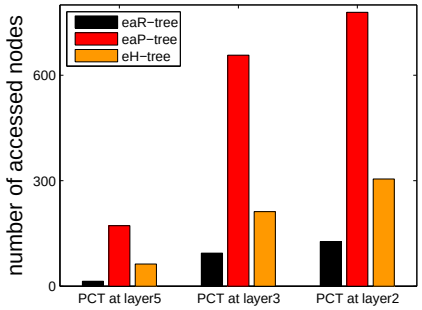
In addition, we can observe the same feature from the *eH-tree*, which improves the *eaP-tree*. Surprisingly, the *eH-tree* has a smaller number of accessed nodes in CTT computation than the *eaP-tree* in all cases. The reasons for this are twofold. As illustrated in subsection 4.2.4.3, on one hand, the *eH-tree* adopts the *aP$^+$-tree* to reduce the number of accessed nodes for the left border VRA. On the other hand, in *eH-tree*, for the right border VRA, the search is done in the

fully ordered transaction amount space maintained in the *aB$^+$-tree*. Unlike the above two approaches, when the time range in a CTT query becomes larger, the number of accessed nodes for the *eaR-tree* increases linearly due to more MBRs overlapped by the enlarged query range.

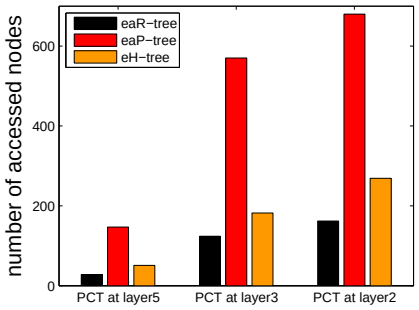
(2) The *eaR-tree* has the best performance for over 50% CTT queries. The reason is that the added data fields $count\_r$ and sum_r in each record of L-nodes in the *eaR-tree* allow operations on the index of product space rather than the whole transaction space on each day, which can reduce the number of accessed nodes. By contrast, both the *eaP-tree* and the *eH-tree* extend the *aP-tree*. As illustrated in subsection 4.2.3.3, they index all the transactions, and cannot essentially aggregate repeated transactions that occurred on a day, leading to inferior performance.

(3) According to the results on two eBay real datasets and three large-scale synthetic datasets, we can draw the conclusion that the *eaR-tree* has the best performance when the CTT query region for the Transaction Time dimension is small (e.g., in Fig. 6.2(a) and 6.2(d), where the queries of the transaction time range are regarding one month). By contrast, the *eH-tree* has the best performance when the CTT query region of the Transaction Time dimension becomes larger (e.g., in Fig. 6.2(c) and 6.2(f), where the queries of the transaction time range are regarding three months).

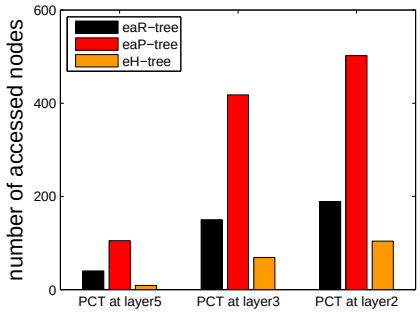
(4) Unlike the *eaR-tree* and the *eaP-tree*, the performance of the *eH-tree* is the least affected by the large-scale of synthetic datasets containing transactions distributed in 12 months. The reason is that when transaction time extends to a longer period (e.g., 12 months), it means a larger query region for the *eaR-tree* and more alive nodes for the *eaP-tree* when computing the right border VRA. By contrast, the *eH-tree* only needs to search the additional *aB$^+$-tree* which is fully ordered in the transaction amount space for the right border VRA, and thus a long time period in CTT computation does not much affect the number of accessed nodes.



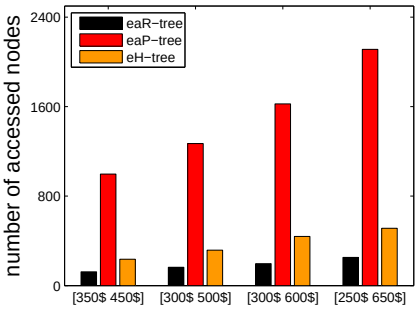
(a) The computation of PCT at different layers in 3 months



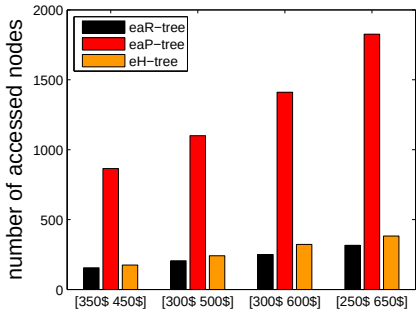
(b) The computation of PCT at different layers in 6 months



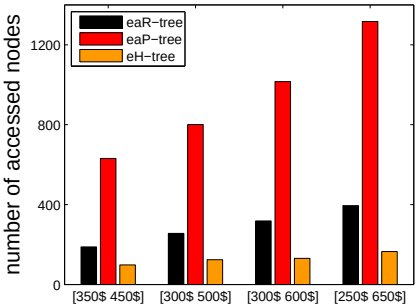
(c) The computation of PCT at different layers in 12 months



(d)The computation of STAT within 3 months

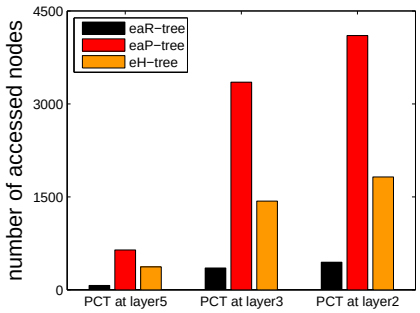


(e) The computation of STAT in 6 months

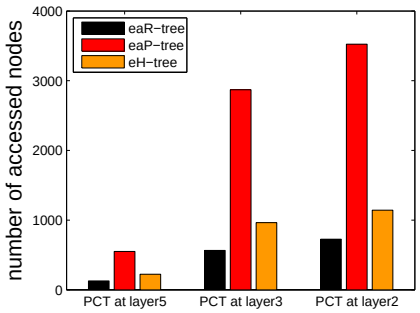


(f) The computation of STAT in 12 months

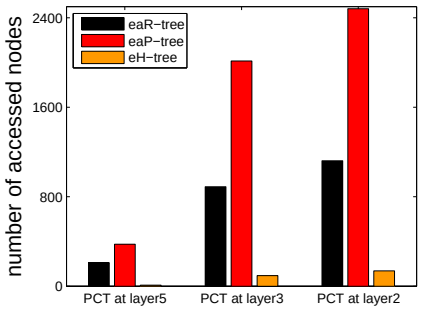
Figure 4.9: The performance of three proposed approaches on SD_1 (synthetic dataset)



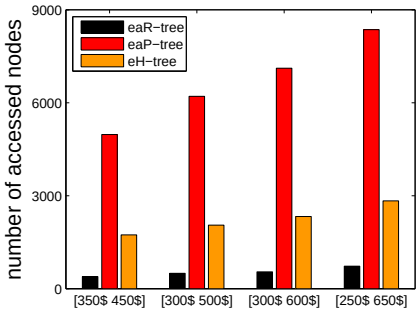
(a) The computation of PCT at different layers in 3 months



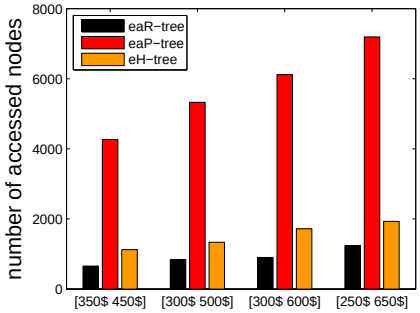
(b) The computation of PCT at different layers in 6 months



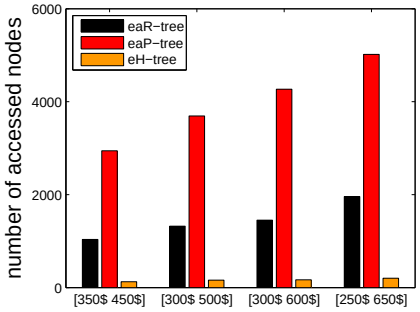
(c) The computation of PCT at different layers in 12 months



(d) The computation of STAT in 3 months

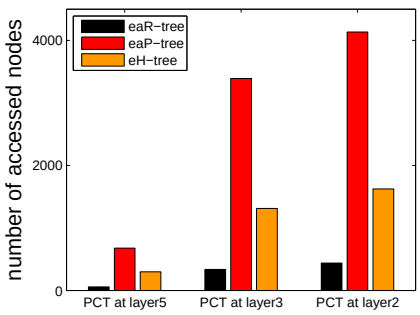


(e) The computation of STAT in 6 months

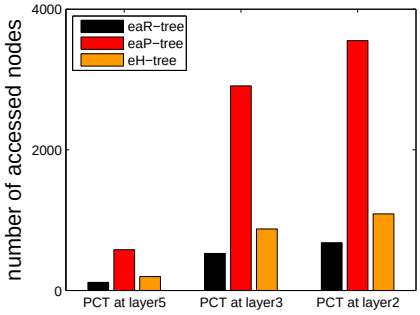


(f) The computation of STAT in 12 months

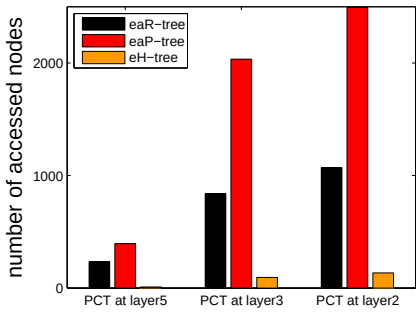
Figure 4.10: The performance of three proposed approaches on SD_2 (synthetic dataset)



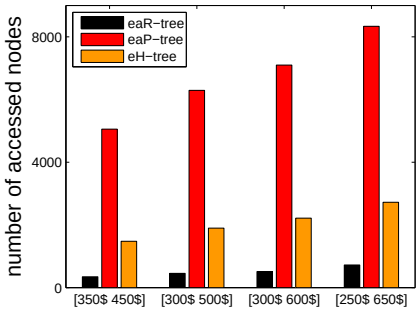
(a) The computation of PCT at different layers in 3 months



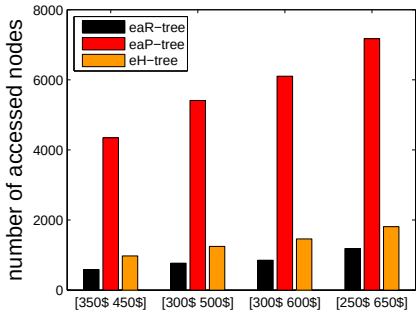
(b) The computation of PCT at different layers in 6 months



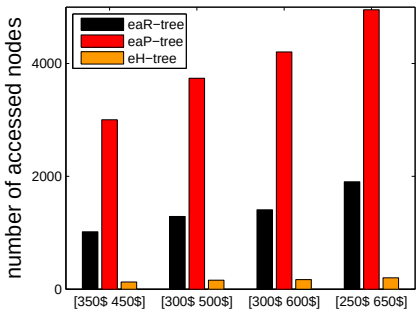
(c) The computation of PCT at different layers in 12 months



(d) The computation of STAT in 3 months



(e) The computation of STAT in 6 months



(f) The computation of STAT in 12 months

Figure 4.11: The performance of three proposed approaches on SD_3 (synthetic dataset)

4.4 Summary

In the literature, many researchers have pointed out that it is necessary to introduce context factors to trust evaluation in e-commerce environments [164, 158, 159, 121, 150, 149, 74, 111]. To this end, in Chapter 3, we target the context-aware transaction trust evaluation problem and have proposed a trust vector which can be used to outline the reputation profile of a seller. It aims to identify the *context imbalance* problem in forthcoming transactions that can cause a huge monetary losses for victim buyers. To the best of our knowledge, this is the first solution in the literature to the computation of contextual transaction trust (CTT) in e-commerce environments.

In this chapter, we first modelled CTT computation as a two-dimensional (2D) Range Aggregate (RA) problem. Then, we presented how to extend two-dimensional RA for CTT Computation. Finally, we have proposed three new approaches to contextual transaction trust (CTT) computation in e-commerce environments, namely, the *eaR-tree*, the *eaP-tree* and the *eH-tree*. From the result of our experiments, we can draw a conclusion that, among these three approaches, the *eaR-tree* has better performance when the query region for the transaction time dimension is small. The *eH-tree* has better performance when the CTT query for the transaction time dimension becomes larger. This is an important advantage when processing large-scale datasets containing transactions for a 12 month period.

In next chapter, we will propose a new data structure *CMK-tree* for CTT computation. In particular, the new structure may further reduce the number of accessed nodes in response to CTT queries. In our proposed *eH-tree*, *aP$^+$-tree* is used for computing the left border VRA. Therefore, it still has the same drawbacks as the *eaP-tree* with low efficiency when the query range for the Transaction Time dimension is small. If we can record the number of intersecting intervals (VRA) continuously over time, the total number of accessed nodes for both the left border VRA and the right border VRA will be reduced.

Chapter 5

An Efficient Approach to the Computation of Reputation Profile

In Chapter 3, we have presented a trust vector consisting of three values for representing Contextual Transaction Trust (CTT). In the computation of CTT values, three identified important *context dimensions*, including Product Category, Transaction Amount and Transaction Time, are taken into account. In the meantime, the computation of each CTT value is based on both past transactions and the forthcoming transaction. In particular, with different parameters specified by a buyer regarding context dimensions, different sets of CTT values can be calculated. As a result, all these trust values can outline the reputation profile of a seller that indicates the dynamic trustworthiness of a seller in different products, product categories, price ranges, time periods, and any necessary combination of them. We term this new model as *ReputationPro*. Nevertheless, in *ReputationPro*, the computation of reputation profile requires new data structures for appropriately indexing the pre-computation of aggregates over large-scale ratings and transaction data in three context dimensions as well as novel algorithms for promptly answering buyers' CTT requests.

To solve this challenging problem, in this chapter, we propose a new index scheme *CMK-tree* by extending the two-dimensional *K-D-B-tree* [114] that indexes spatial data to support efficient computation of CTT values. The *CMK-tree* is not only applicable to the three context dimensions that are either linear or hierarchical, but also takes into account the characteristics of the transaction-time model, i.e. transactions data is

inserted in chronological order. Moreover, the proposed data structures can index each specific product traded in a time period in order to compute the trustworthiness of a seller in selling a product. Finally, the experimental results illustrate that the *CMK-tree* is superior in efficiency of computing CTT values to all three existing approaches in the literature (the approaches proposed in Chapter 4). In particular, while answering a buyer's CTT queries for each brand-based product category, the *CMK-tree* has almost linear query performance.

This chapter is organised as follows: Section 5.1 presents our proposed new index scheme *CMK-tree* in detail; Section 5.2 presents CTT computation algorithm based on our proposed *CMK-tree*; Section 5.3 provides the analysis of the structure and performance of *CMK-tree*; Section 5.5 depicts the experiments on *CMK-tree*; and Section 5.5 concludes our work in this chapter.

5.1 The Proposed CMK-tree

This section introduces the *CMK-tree* — a disk-based index structure that supports efficient computation for a buyer's CTT queries. While Section 5.1.1 summarises limitations of existing approaches to two-dimensional (2D) Range Aggregate (RA) after being extended to solve the CTT computation problem. To overcome all these limitations, we propose a *MK-tree* — an extended multi-version *"domain 0"* two-dimensional K-D-B-tree [114]. Then, with the third dimension Product Category taken into account, a *CMK-tree* is formed. While Section 5.1.2 describes the structure of the *CMK-tree*, the process of *CMK-tree* construction is provided in Section 5.1.3.

5.1.1 The Limitations of Existing Approaches

5.1.1.1 The Summary of Two-Dimensional (2D) Range Aggregate (RA)

In Section 2.4.2, we have reviewed the existing approaches to two-dimensional (2D) RA problem, including *aR-tree* [63, 104], *aP-tree* [135], *MVSB-tree* [166, 168] and

the *BA-tree* [167]. Instead of introducing them again, this subsection focuses on identifying the limitations of these techniques in resolving our targeted problem.

(a) As pointed out in Section 2.4.2, a serious problem of the *aR-tree* is that its performance significantly degrades when answering a large query region, since in such a case there are more MBRs overlapping with the query region (see Fig. 2.4(b)).

(b) In order to answer an RA query, the *aP-tree* indexes all the objects [168, 173]. As shown in Fig. 2.3, in the traditional RA problem, one point in a two-dimensional space represents only one object (e.g., a car). However, in our targeted CTT computation problem, one point may represent multiple such objects. For example, it is quite common that a seller has multiple transactions with the same price selling the same product on a day. Here the x-axis represents days, and the transactions occurred on the same day have the same x-coordinate. In this case, the proposed new index scheme should take this characteristic into account. Unlike the *aP-tree*, the new index does not need to index all the transactions; rather, multiple repeated transactions should be aggregated, and then the new index only needs to store the aggregation results.

(c) Although the *MVSB-tree* considers that a point may represent multiple such objects, it overlooks the inserted objects themselves [168]. If it is applied to the CTT computation problem, each specific product cannot be indexed. As a result, computing the trustworthiness of a seller in selling a product (i.e. TIST) cannot be fulfilled. Furthermore, the *MVSB-tree* is particularly designed for solving the dominance-sum problem. Thus, four dominance-sum queries are needed to answer an RA query (see Fig.2.6). By contrast, our proposed new index scheme answers only two VRA queries for the same purpose, and thus improves efficiency (see Fig. 2.5).

(d) Moreover, the *BA-tree* achieves linear performance when answering each dominance sum query. However, as pointed out by [168], it does not fit the transaction-

Table 5.1: The summary of limitations

Approaches	Limitations
aR-tree	Performance degrades with a larger query region
aP-tree	Cannot essentially aggregate the same objects
MVSB-tree	Overlooks the inserted objects themselves; Based on four dominance-sum queries
BA-tree	Does not fit the transaction-time model

time model where the records of transactions are inserted in chronological order.

Table 5.1 summarises the limitations of existing approaches to two-dimensional (2D) Range Aggregate (RA) after being extended to solve CTT computation problem.

5.1.1.2 The Characteristics of CMK-tree

In Section 4.2, we have proposed three disk-based index schemes for CTT computation, i.e. *eaR-tree*, *eaP-tree* and *eH-tree*. All these approaches can meet the requirements of answering a buyer's CTT queries on the dynamic trustworthiness of a seller in different product categories, price ranges and time periods. However, they have poor performance in some cases. Specifically, as the *eaR-tree* extends the *aR-tree* [63, 104], the query cost for *eaR-tree* depends on the size of the CTT query region formed by the price range and transaction time range: the larger the query region, the worse the performance in answering a CTT query. For both the *eaP-tree* and the *eH-tree* that extend the *aP-tree* [135], they index all the transactions and cannot essentially aggregate repeated transactions that occurred on a day, leading to inferior performance.

Towards efficient CTT computation, we propose a new disk-based index scheme *CMK-tree* and a new query algorithm. In this new index scheme, the above problems will be solved. Like the *BA-tree* [167], the *CMK-tree* extends the *K-D-B-tree*, however, it adopts a different extension strategy that is particularly designed to efficiently support CTT computation. Combined with our targeted CTT computation problem, we summarise some important and special characteristics of the CMK-tree as below:

- In the traditional two-dimensional RA problem [135] (see Fig. 2.3), one point represents one object only (e.g., a car). By contrast, as a common case in e-commerce environments, a seller may have multiple transactions with the same price on a given day selling the same product, i.e. one point may represent multiple such transactions. The *CMK-tree* does not index all transactions but aggregates the repeated transactions on a given day, which sell the same product;

- Some existing approaches to the two-dimensional RA problem overlook the inserted objects. Unlike these approaches, the *CMK-tree* guarantees that each specific product can be indexed in order to compute the trustworthiness of the seller in selling a product (i.e., TIST, see Section 3.3.2);

- The CTT computation has the same characteristic as the transaction-time model [168], i.e. the records of newly happened transactions are inserted in chronological order. The *CMK-tree* adopts multi-version structure [17] to effectively deal with transaction-time model;

- Only two Vertical Range Aggregate (VRA) queries [135] are carried out to answer a CTT query based on the *CMK-tree*. This is more efficient than the *MVSB-tree* which needs to carry out four dominance-sum queries for a RA query.

Finally, considering that each point in a two-dimensional space represents one transaction or a set of transactions, for further analysis, we differentiate the following three cases. Let us refer to a two-dimensional space plotted in Fig. 2.3. Note that all the notations or symbols follow the definitions given in Section 3.3.1.

Case 1: *Given one point at* (t_i, ta_i)*, it may represent only one transaction that occurred on a day* t_i *with the transaction amount* ta_i*.*

Case 2: *As mentioned above, one point at* (t_i, ta_i) *may represent a set of repeated transactions that occurred on a day* t_i *selling the same product with the same price* ta_i*. In such a case, we need data structures to aggregate these repeated transactions.*

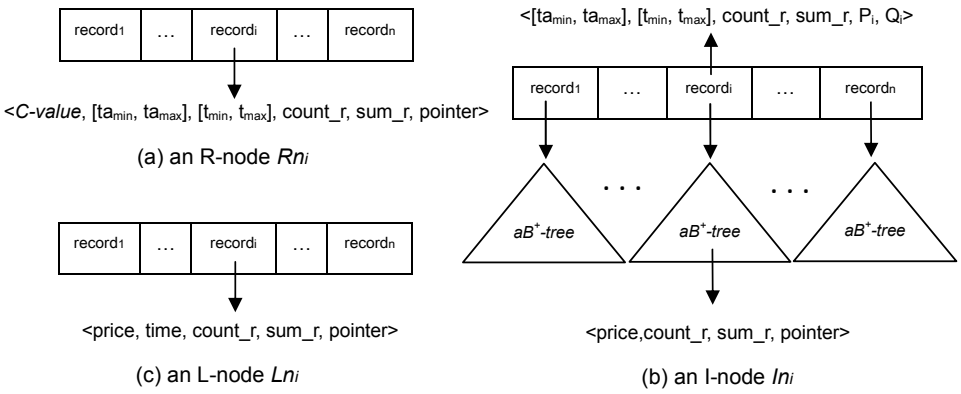


Figure 5.1: The node structure of a *CMK-tree*

> ***Case 3:*** *Given one point at* (t_i, ta_i)*, it may represent a set of transactions that occurred on a given day* t_i *selling different products with the same price* ta_i*. In such a case, they should be regarded as different transactions and aggregated separately.*

Note that if the prices of transactions selling the same product are different, we regard them as different transactions and aggregate them separately.

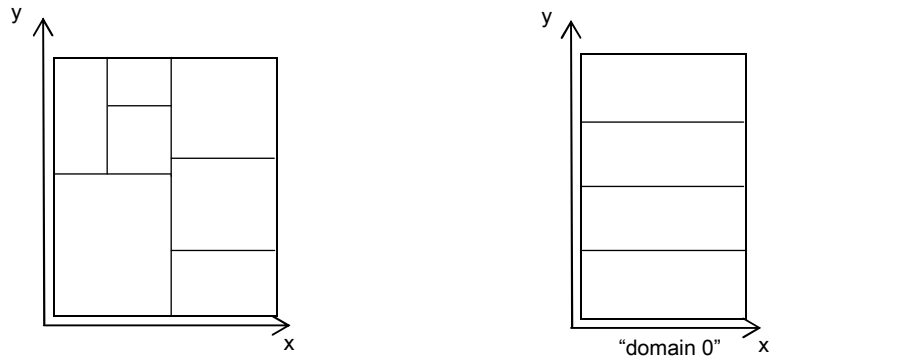
5.1.2 Structure of the CMK-tree

As depicted in Fig. 4.2, there are also three types of nodes in the *CMK-tree*: *R-node* (Rn), *I-node* (In) and *L-node* (Ln), each of which can have multiple records depending on the node capacity. Generally speaking, one *CMK-tree* consists of a *C-tree* and multiple *MK-trees* that are external to the *C-tree*. The *C-tree* consists of R-nodes, and an *MK-tree* consists of I-nodes and L-nodes.

Following Step 2 in the extension process described in Section 4.1.2, each record in an R-node Rn_i has the form

$$< \textit{C-value}, [ta_{min}, ta_{max}], [t_{min}, t_{max}], count\_r, sum\_r, pointer >,$$

As mentioned in Section 4.2.2.1, all R-nodes form an *N-ary tree*, and we term it as a *C-tree (product* ***C****ategory-tree)*.



(a) The 2-D-B-tree partitions the two-dimensional space

(b) The space cannot be split along "domain 0" (e.g. x-axis) in the 2-D-B-tree

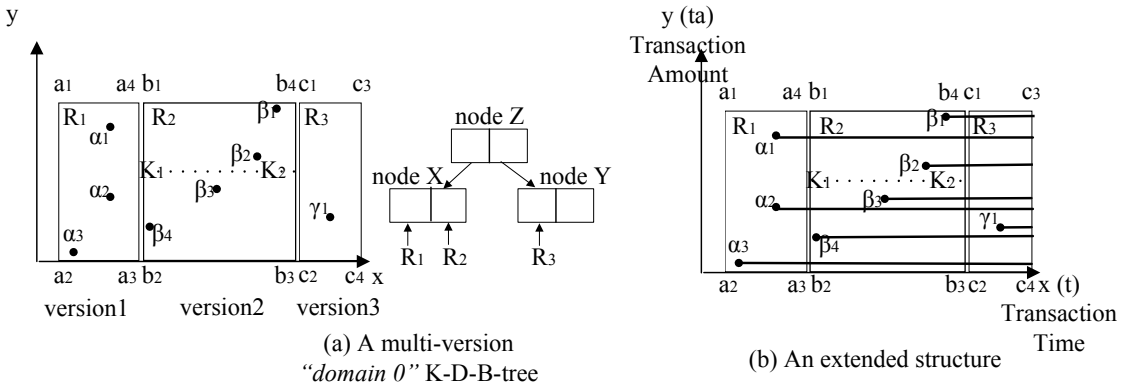
Figure 5.2: A special case of *2-D-B-tree*

In addition to a *C-tree* consisting of R-nodes, following Step 3 in the extension process, each record in an R-node at the brand-based product category layer (i.e. the bottom of the *C-tree*) points to a subtree that is external to the *C-tree*. Specifically, the design of each such subtree is based on extending the original two-dimensional *K-D-B-tree* or *2-D-B-tree* [114] that is used for indexing spatial data.

The *2-D-B-tree* partitions a two-dimensional space into multiple nonintersecting rectangles (see Fig. 5.2(a)). Each record in a node in the *2-D-B-tree* corresponds to a rectangular space. Unlike the general structure depicted in Fig. 5.2(a), a special case *"domain 0" K-D-B-tree* has been proposed in [114]. In particular, for a general *K-D-B-tree*, a rectangular space can be split along any dimension (e.g., x-axis or y-axis). By contrast, for a *"domain 0" K-D-B-tree*, the space cannot be split along a specific dimension. For example, in Fig. 5.2(b), instead of dividing x-axis, the split can only be operated along the y-axis.

In the *CMK-tree*, the idea of *"domain 0" K-D-B-tree* [114] is adopted to generate each subtree to extend the *C-tree*. Since the x-axis (Transaction Time dimension) continuously moves to the right in our targeted problem, each subtree can be considered as a multi-version structure that makes *partial persistence*[1] [168] a *"domain 0" 2-D-*

[1]*partial persistence* implies that updates are only applied to the latest version of the data structure, creating a linear ordering of versions.



(a) A multi-version *"domain 0"* K-D-B-tree

(b) An extended structure

Figure 5.3: *MK-tree* – An extended multi-version *"domain 0"* two-dimensional *K-D-B-tree*

B-tree. In order to further demonstrate the structure of a subtree, we assume that each point in a two-dimensional space depicted in Fig. 5.3(a) represents a transaction, and all the transactions belong to the same brand-based product category. Correspondingly, in the subtree generated by these transactions (see Fig. 5.3(a)), a record in the node X surrounds the rectangular $a_1a_2a_3a_4$ (R_1). This is the first version of *"domain 0" 2-D-B-tree* as illustrated in Fig. 5.2(b). Another record in the node X surrounds the rectangular $b_1b_2b_3b_4$ (R_2) which is the second version. The record in the node Z at a higher level surrounds a larger rectangular space formed by $a_1a_2b_3b_4$. Furthermore, like the transformation given in Section 2.4.2.2, each transaction will generate an interval along the Transaction Time dimension (see Fig. 5.3(b)). As an extended structure, each record simultaneously maintains the transactions whose generated intervals intersect with the left border of the corresponding rectangle as well as the aggregates of transactions. For example, in Fig. 5.3(b), the record surrounding the rectangle $b_1b_2b_3b_4$ also stores the aggregates of transactions α_1, α_2 and α_3 whose generated intervals along the Transaction Time dimension intersect with the left border b_1b_2 as well as the indexes of these three transactions for computing the trustworthiness of the seller in selling a specific product. To facilitate discussion, we term such a subtree, i.e. an extended ***M**ulti-version "domain 0" two-dimensional **K**-D-B-tree*, as *MK-tree*.

Next, we introduce the I-nodes and the L-nodes of an *MK-tree*. Based on the above description, the record in an I-node In_i (see Fig. 5.1(b)) has the form

$$< [ta_{min}, ta_{max}], [t_{min}, t_{max}], count\_r, sum\_r, P_i, Q_i > ,$$

where $[ta_{min}, ta_{max}]$ and $[t_{min}, t_{max}]$ surround each nonintersecting rectangle; the term P_i is a pointer pointing to its child, which is an I-node or an L-node; the term Q_i is another pointer pointing to an *aB$^+$-tree* that derives from the *B$^+$-tree* [15]. As stated before, the purpose of building an *aB$^+$-tree* is to index the transactions whose generated intervals along the Transaction Time dimension intersect with the left border of the rectangle surrounded by $[ta_{min}, ta_{max}]$ and $[t_{min}, t_{max}]$. In the meantime, the aggregates over these transactions are maintained in $count\_r$ and sum_r. Specifically, each *aB$^+$-tree* is built in the separate transaction amount space. Like the *B$^+$-tree*, the records in an *aB$^+$-tree* are kept sorted based on their transaction amount (price). Also, each node in an *aB$^+$-tree* has the same structure that consists of multiple records, each of which has the form $< price, count\_r, sum\_r, pointer >$ (see Fig. 5.1(b)). The $pointer$ points to its child, but for the records at leaf level, $pointer$ points to the transaction record stored in the database. Note that each generated *aB$^+$-tree* is based on the transactions in a brand-based product category. In addition, unlike the original *B$^+$-tree*, the number of records to be inserted in an *aB$^+$-tree* may be larger than the number of distinct values of transaction amount (y-coordinates) due to the reason illustrated in *Case 3* in Section 5.1.1.2.

Moreover, when building an *MK-tree*, in order to avoid duplication of *aB$^+$-trees*, the I-nodes actually include two types: (1) the I-node pointing to *aB$^+$-trees*, and (2) the I-node without pointing to *aB$^+$-trees*. To differentiate the above two situations, we call the I-node '*L-I-node (In(L))*' if its children are L-nodes, and '*I-I-node (In(I))*' otherwise. Therefore, the I-node depicted in Fig. 5.1(b) is an L-I-node $In(L)_i$. These two node structures will become clearer after introducing insertions in the next subsection.

Each record appearing in an L-node Ln_i (see Fig. 5.1(c)) also contains $count\_r$ and sum_r because of aggregating the repeated transactions with the same price on a given day which sell the same product. It has the following form

$$< price, time, count\_r, sum\_r, pointer > ,$$

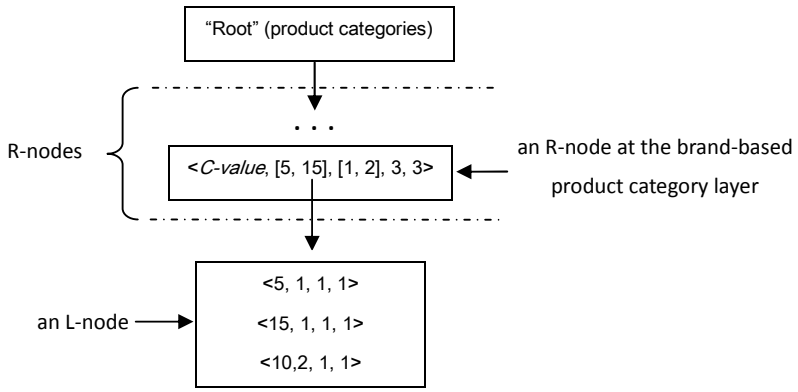


Figure 5.4: The state of a *CMK-tree* after inserting three transactions

where the *pointer* points to the transaction record stored in the database.

5.1.3 The Construction of a CMK-tree

This section formally describes the insertion of transactions for the *CMK-tree*, including the path in the product category hierarchy (*C-hrchy*$_i$), transaction amount (ta_i), transaction time (t_i) and the rating for the transaction (r_i). In the meantime, the transaction records are inserted in chronological order.

5.1.3.1 Insertion

Before inserting the data of a newly happened transaction into a *CMK-tree*, a path is first searched in a *C-tree* from top (the product category root) to bottom (the brand-based product category) (see Fig. 4.2) based on the *C-hrchy* of the transaction. If the product in the transaction belongs to a new product category on which the seller has no prior transactions, the new records are generated for this product category as well as its sub-categories and inserted to the corresponding R-nodes. Otherwise, the set of ranges and aggregates (i.e., $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$, sum_r) maintained in each record along the path are updated accordingly. After that, the insertion operations should be performed in an *MK-tree* pointed by the corresponding record in an R-node at the brand-based product category layer. Fig. 5.4 depicts the state of a *CMK-tree*

after inserting the data of three transactions trading different products, which belong to the same product category with the same *C-hrchy*. In addition, the information $<$ ta_i (transaction amount), t_i (transaction time), r_i (rating) $>$ of three transactions is $< 5, 1, 1 >$, $< 15, 1, 1 >$ and $< 10, 2, 1 >$.

5.1.3.2 Split

The split of an R-node is relatively simple. Unless stated otherwise, in the following example, we assume the capacity of all the nodes is five. In addition, the field sum_r in all the records is ignored, and we only use the field $count\_r$ as the example to illustrate the aggregation process. Fig. 5.5(a) shows an example of the R-node split where Rn_1 is an R-node containing a record $<$ *C-value*$_p$, $[10, 60], [1, 5], 32$ $>$ at a higher level of the *C-tree*. The R-node Rn_1 points to its child node Rn_2 containing five records, i.e. from *C-value*$_{ch_1}$ to *C-value*$_{ch_5}$. Here we use *C-value*$_p$ and *C-value*$_{ch}$ to denote the parent product category and the child product category, respectively. $[10, 60]$ is the transaction amount range, and $[1, 5]$ is transaction time range. 32 is the total number of transactions in the current product category represented by *C-value*$_p$.

When a new record $<$ *C-value*$_{ch_6}$, $[15, 15], [6, 6], 1 >$ (i.e., a new product category) is inserted to the R-node Rn_2, it overflows, and then this record is moved to a new R-node Rn_3. In the meantime, Rn_3 is pointed to by another new record generated in Rn_1, and the data fields $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$ and sum_r in this record are updated to reflect the ranges and aggregates of its child node (see Fig. 5.5(a)).

The split of either an L-node or an I-node that occurs in an *MK-tree* is more complicated, and includes two situations, respectively.

1) L-node split

Situation 1: If the record to be inserted in an L-node has the same transaction time (i.e., x-coordinate) as a record existing in it, the L-node splits according to the transaction amounts of all the records it contains.

Fig. 5.5(b) illustrates the split of an L-node Ln_1 after inserting a new record $<$

$30, 2, 1 >$, which leads to a new L-I-node $In(L)_1$. The number 30 is the *transaction amount*, 2 is the *transaction time* and 1 is the field $count\_r$ in sequence. Note that the inserted record first needs to be checked whether the transactions with the same price selling the same product have already been indexed by the records in Ln_1. If there exists repeated transactions, instead of splitting the L-node Ln_1, it is only to update $count\_r$ and sum_r in the corresponding record within Ln_1. For the L-I-node $In(L)_1$, in order to get the full partition on the entire transaction amount space, the boundary is set to the intermediate value of the maximal transaction amount in the new L-node Ln_1 and the minimal transaction amount in the L-node Ln_2. For instance, in Fig. 5.3(b), the y-coordinate of a boundary K_1K_2 equals to $\lfloor \frac{\mathbf{y}_{\beta_2}+\mathbf{y}_{\beta_3}}{\mathbf{2}} \rfloor$. Hence, the two records in a generated L-I-node $In(L)_1$ are $< [0, 13], [1, 2], 0, Q_1 >$ and $< [13, \infty), [1, 2], 0, Q_2 >$, respectively, where both Q_1 and Q_2 point to an *aB$^+$-tree*.

- ***aB$^+$-tree split*:** When an L-node pointed by a record in the L-I-node is split into two L-nodes, correspondingly, the *aB$^+$-tree* pointed by this record also needs to split. For example, we assume that all the transactions represented by the points in Fig. 5.3(b) are in the same brand-based product category. If the rectangle $b_1b_2b_3b_4$ splits into two rectangles $b_1K_1K_2b_4$ and $K_1b_2b_3K_2$, the original *aB$^+$-tree* splits into two *aB$^+$-trees* that store the information of one point (i.e., α_1) and two points (i.e., α_2, α_3) in $a_1a_2a_3a_4$, respectively.

Situation 2: If the record to be inserted in an L-node has a different transaction time (i.e., x-coordinate), a new L-node is generated.

Fig. 5.5(c) illustrates that if the record inserted to the L-node Ln_1 is $< 30, 3, 1 >$, a new L-node Ln_2 is generated. In the meantime, an L-I-node $In(L)_1$ is generated with two records pointing to Ln_1 and Ln_2. The two records in the I-node $In(L)_1$ are $< [1, 15], [1, 3), 0, Q_1 >$ and $< [0, \infty), [3, \infty), 9, Q_2 >$.

- ***Generate a new aB$^+$-tree*:** In Fig. 5.5(c), the record $< [0, \infty), [3, \infty), 9, Q_2 >$ in the I-node $In(L)_1$ surrounds a new rectangle. Accordingly, a new *aB$^+$-tree*

pointed by Q_2 has to be generated to index the transactions in the same brand-based product category whose generated intervals along the Transaction Time dimension intersect with the left border $3 : [0, \infty)$. In addition, the number 9 (i.e. $count\_r$) denotes the total number of such transactions.

In Fig. 5.5(c), we assume that two records $< 5, 1, 1 >$ and $< 5, 2, 2 >$ in the L-node Ln_1 index the transactions selling the same product but traded at different time; another two records $< 15, 1, 1 >$ and $< 15, 2, 3 >$ index the transactions selling different products but traded at the same price. The new *aB$^+$-tree* pointed by Q_2 is equivalent to the *aB$^+$-tree* pointed by Q_1 after inserting four different records: $< 5, 3 >$, $< 10, 2 >$, $< 15, 1 >$ and $< 15, 3 >$. However, since the record $< [1, 15], [1, 3), 0, Q_1 >$ in the I-node $In(L)_1$ represents the initial rectangle, the *aB$^+$-tree* pointed by Q_1 is null in this example and the field $count\_r$ is 0. Here the above four records do not include the Transaction Time dimension, as each *aB$^+$-tree* is built in a separate Transaction Amount dimension. The insertion and split of an *aB$^+$-tree* are the same as those of a *B$^+$-tree* [15].

Notice that 3 in the record $< 5, 3 >$ is the aggregated value for the field of $count\_r$ in two records $< 5, 1, 1 >$ and $< 5, 2, 2 >$, as they index the transactions selling the same product. Also, the records $< 15, 1 >$ and $< 15, 3 >$ index the transactions selling different products, and thus they should be inserted separately.

- ***Merge aB$^+$-trees*:** Several *aB$^+$-trees* may need to be merged to generate a new *aB$^+$-tree*. In Fig. 5.5(b), assume another record $< [0, \infty), [3, \infty), 10, Q_3 >$ is to be inserted in the L-I-node $In(L)_1$. Since two records $< [0, 13], [1, 2], 0, Q_1 >$ and $< [13, \infty), [1, 2], 0, Q_2 >$ in the original L-I-node $In(L)_1$ have the same time range $[1, 2]$, the two *aB$^+$-trees* pointed by Q_1 and Q_2 respectively first need to be merged to form a new *aB$^+$-tree*. Then, the newly generated *aB$^+$-tree* is pointed by Q_3. The above operations are performed on all the records with the same time range.

Here we need to emphasise that since the L-node split in Situation 2 leads to the

problem of space utilisation, in order to guarantee the minimum space utilisation for each L-node larger than 50%, Situation 2 happens only when all the L-nodes are at least half full.

2) I-node split

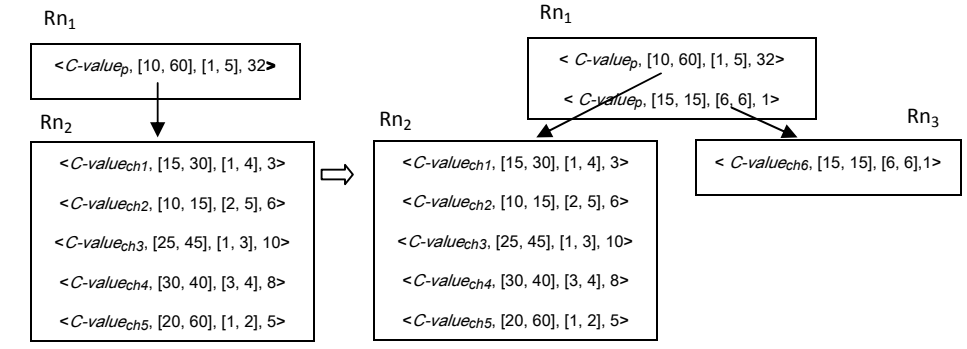
Situation 1: If the record to be inserted in an I-node has the same transaction time range as an existing record in that I-node, all the records with the same transaction time range as the inserted record will be moved to a newly generated I-node. In addition, if each record in an I-node has the same transaction time range as the inserted record, the same as the L-node split in Situation 1, the I-node splits according to the transaction amount ranges of all the records it contains.

In Fig. 5.5(d), for example, if the record $< [15, 30], [3, \infty), 9, Q_5 >$ in the L-I-node $In(L)_1$ splits into two new records $< [15, 20], [3, \infty), 5, Q_5 >$ and $< [20, \infty), [3, \infty), 4$ $, Q_6 >$, the above two records lead to the overflow of $In(L)_1$. Then, the node $In(L)_1$ will continue to split into two L-I-nodes: a new $In(L)_1$ and a new $In(L)_2$. In the meantime, another I-I-node $In(I)_3$ with two records $< [0, 50], [1, 3), 0 >$ and $< [0, \infty), [3, \infty), 19 >$ $(19 = 10 + 9 = 10 + 5 + 4)$ is generated to point to the node $In(L)_1$ and the node $In(L)_2$, respectively.

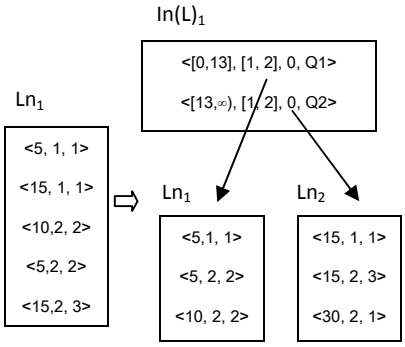
Situation 2: If the record to be inserted in an I-node has a different transaction time range, similar to the L-node split in Situation 2, a new I-node is generated to contain the inserted record. Such a strategy is to guarantee the maximum space utilisation for an I-node.

In Fig. 5.5(e), after the record $< [0, \infty), [4, \infty), 19 >$ inserted to $In(L)_1$, a new L-I-node $In(L)_2$ is generated. Meanwhile, an I-I-node $In(I)_3$ with two records $< [0, 50], [1, 4), 0 >$ and $< [0, \infty), [4, \infty), 19 >$ is generated to point to the I-nodes $In(L)_1$ and $In(L)_2$, respectively. Notice that each record in node $In(I)_3$ will not include a pointer Q_i to avoid duplication of the *aB$^+$-tree*. This is because the same *aB$^+$-tree* has already been indexed by its child node $In(L)_2$, which is an L-I-node.

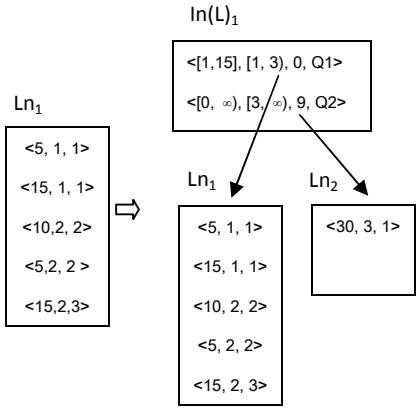
Algorithm 2 presents pseudo-code for the complete insertion process.



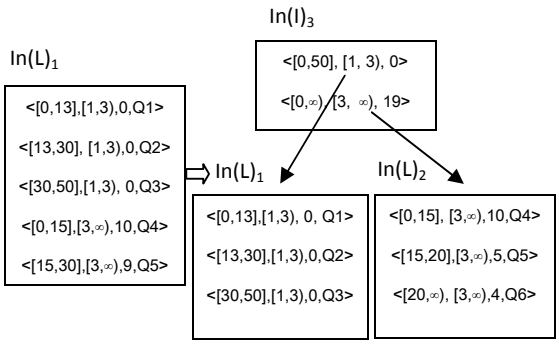
(a) The split of an R-node



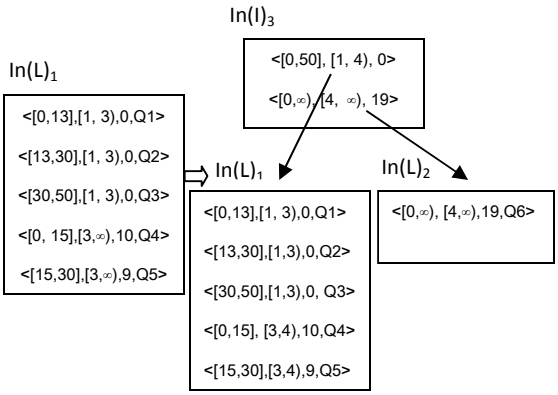
(b) The split of an L-node in Situation 1



(c) The split of an L-node in Situation 2



(d) The split of an I-node in Situation 1



(e) The split of an I-node in Situation 2

Figure 5.5: The construction of a *CMK-tree*

Algorithm 2 The CMK-tree Construction

Input: A transaction TR_i includes *C-hrchy*$_i$, ta_i, t_i and r_i.
Output: CMK-tree

1: // construct a *C-tree*
2: Starting from the "Root" of the product category hierarchy
3: Determine the path in *C-tree* based on the *C-hrchy*$_i$ of each transaction TR_i.
4: **for all** R-nodes along the path **do**
5: **if** The product traded in TR_i belongs to a new product category on which the seller has no prior transactions **then**
6: i. insert the generated new record in corresponding R-node.
7: ii. if the R-node overflows, split.
8: **else**
9: update corresponding ranges and aggregates maintained in a record
10: **end if**
11: **end for**
12: // construct the *MK-trees* that are external to the *C-tree*
13: **for all** L-nodes and I-nodes in the path from bottom up **do**
14: **if** the node is the L-node **then**
15: i. if transaction time t_i is different from any transactions in the L-node, and this L-node is at least half full, then generate a new L-node.
16: ii. if transaction time t_i is the same as a record in the L-node, and repeated transactions have already been indexed in this L-node, then update corresponding $count\_r$ and sum_r.
17: iii. otherwise, insert the transaction TR_i
18: iv. if the L-node overflows, split it
19: **else if** the node is the I-node **then**
20: i. insert the generated new records
21: ii. if the I-node overflows, split it
22: **end if**
23: **end for**

5.2 The Proposed CTT Computation Algorithm

Basically, the CTT computation algorithm answers a buyer's typical CTT queries covering three transaction dimensions based on our proposed *CMK-tree*. The processing of CTT computation starts by locating product category in the *C-tree* according to the *C-value* (i.e. product category) in a buyer's query. Then, it computes the left border VRA and the right border VRA (see Fig 2.5(b)) in one or several *MK-trees*, respectively, depending on the number of brand-based product categories that are included

Algorithm 3 CTT Computation Algorithm in a CMK-tree

Input: A typical CTT query with specific product category, transaction amount range ($[ta_1, ta_2]$) and transaction time range ($[t_1, t_2]$).

Output: CTT value

1: CTT=0
2: $count\_r_1$=0, sum_r_1=0, $count\_r_2$=0, sum_r_2=0
3: The Searching starts from the "Root" of product category hierarchy
4: Determine the layer in product category hierarchy based on the CTT query on product category, and return corresponding record R in an R-node.
5: begin Search(CMK-tree, R)
6: **if** the record is in an R-node **then**
7: **Case 1:** CTT=CTT+$\frac{sum\_r}{count\_r}$, $count\_r$ and sum_r are from the record
8: **Case 2:** CTT=CTT
9: **Case 3:** Search(CMK-tree, $R_{childnode}$)
10: **else if** the record is in an I-node **then**
11: Let rec_1 be the index record whose rectangle contains $t_1 : [ta_1, ta_2]$ // for the left border VRA
12: Let rec_2 be the index record whose rectangle contains $t_2 : [ta_1, ta_2]$ // for the right border VRA
13: **while** neither rec_1 nor rec_2 is a record in L-I-nodes **do**
14: i. rec_1 be its child whose rectangle contains $t_1 : [ta_1, ta_2]$
15: ii. rec_2 be its child whose rectangle contains $t_2 : [ta_1, ta_2]$
16: **end while**
17: **for all** rec_1s **do**
18: **if** t_1 equals to the left border of rec_1 **then**
19: only search *aB$^+$-tree* that is pointed by rec_1 for left border to aggregate $count\_r_1$ and sum_r_1
20: **else**
21: the search conducts on both the *aB$^+$-tree* and the L-node pointed by rec_1 to aggregate $count\_r_1$ and sum_r_1
22: **end if**
23: **end for**
24: **for all** rec_2s **do**
25: the search conducts on both *aB$^+$-tree* and L-node pointed by rec_2 to aggregate $count\_r_2$ and sum_r_2
26: **end for**
27: CTT=CTT+$\frac{sum\_r_2 - sum\_r_1}{count\_r_2 - count\_r_1}$
28: **end if**
29: end Search

in a CTT query.

For example, to answer a typical CTT query: <product-category: "*Audio device*", price-range: "$100-$200", time-range: "*the latest 6 months*" >, all the sub-categories

of the product category specified in a CTT query are first considered (see Fig. 4.2). As illustrated in Section 4.2.2.4, since each record in an R-node contains a transaction amount range ($[ta_{min}, ta_{max}]$) and a transaction time range ($[t_{min}, t_{max}]$) for its corresponding product category, there are three cases that should be differentiated. Please refer to Section 4.2.2.4. For *Case 3*, in the following, we take the computation of a VRA $2 : [0, 5]$ as an example to introduce the search process in an *MK-tree*.

As depicted in Fig. 5.5(d), the I-nodes, the rectangle represented by which contains $2 : [0, 5]$, are iteratively searched until reaching the layer of L-I-nodes, and thus the record $< [0, 13], [1, 3), 0, Q_1 >$ in the L-I-node $In(L)_1$ is selected. In order to compute the VRA $2 : [0, 5]$, both the *aB$^+$-tree* and the L-node pointed by the above record need to be searched, and the VRA equals to the sum of the two search results. Note that instead of visiting only one record as introduced in the above example, the search for computing the VRA may be executed on several *aB$^+$-trees* as well as L-nodes pointed by the corresponding records, respectively, depending on the number of the rectangles overlapped by the query range in the Transaction Amount dimension. For instance, to compute another VRA $3 : [10, 30]$ based on Fig. 5.5(d), two records $< [0, 15], [3, \infty), 10, Q_4 >$ and $< [20, \infty], [3, \infty), 4, Q_6 >$ in the L-I-node $In(L)_2$ are selected for conducting the further searches. The aggregation results ($count\_r$ and sum_r) in the record $< [15, 20], [3, \infty), 4, Q_5 >$ can be used directly, as the transaction amount range $[15, 20]$ in that record falls into the query range $[10, 30]$ in the Transaction Amount dimension.

If the transaction time for a VRA equals to the left border of a rectangle represented by the selected record in an L-I-node, search only performs on the *aB$^+$-tree* pointed by this record. For example, in Fig. 5.5(e), to compute the VRA **3** : $[5, 10]$, the record $< [0, 15], [\mathbf{3}, 4), 10, Q_4 >$ in the L-I-node $In(L)_1$ is selected. Instead of searching L-node, only the *aB$^+$-tree* pointed by the above record is searched. However, since the right border in CTT computation is always fixed to the point "*now*", the search for computing the right border VRA is performed on both the *aB$^+$-trees* and L-nodes pointed by the selected records.

Algorithm 3 describes the process of CTT computation in a *CMK-tree*.

5.3 Structure and Performance Analysis

In this section, we will provide an analytical study on the *CMK-tree*, focusing on its structure and query performance. The symbols and their meanings used in our analysis are explained in Table 5.2.

5.3.1 Important Properties

Property 4: Each *MK-tree* in a CMK-tree represents a two-dimensional space formed by Transaction Amount and Transaction Time for all the transactions in a brand-based product category. All the records in each layer of I-nodes in an *MK-tree* fully partition the corresponding two-dimensional space into multiple nonintersecting rectangles.

Proof: Assume all the transactions in a brand-based product category are represented by a number of points in two-dimensional space depicted in Fig. 5.3(a). Since the insertions come in the nondecreasing time order, an new insertion only happens in the latest version of *"domain 0" 2-D-B-tree*, for example, the version 3 in Fig. 5.3(a). In the meantime, the division within each version can only be operated along the Transaction Amount dimension (y-axis). As shown in Fig. 5.5(b) and Fig. 5.5(c), each record in an L-I-node corresponds to either a new version for *"domain 0" 2-D-B-tree* (e.g., $< [0, \infty), [3, \infty), 9, Q_2 >$) or a partition in the latest version (e.g, $< [0, 13], [1, 2], 0, Q_1 >$). Obviously, these records partition the complete two-dimensional space. More importantly, the rectangles formed by them are nonintersecting. In addition, except the records in L-I-nodes, the records in I-I-nodes (a higher level) still fully partition the two-dimensional space into multiple nonintersecting rectangles, for example, the records in node $In(I)_3$ as depicted in Fig. 5.5(d) and Fig. 5.5(e).

Property 5: To answer a CTT query for each brand-based product category, the *CMK-tree* delivers almost linear query performance.

Table 5.2: List of symbols

symbol	explanation
S	a seller
$Trans$	the transaction set for the seller S
n	the number of past transactions contained in $Trans$
m[1]	the number of brand-based product categories (see Fig. 3.1) to which n past transactions belong
n_i[2]	the number of points in a two-dimensional space formed by the transactions in a brand-based product category.
h	the height of the product category hierarchy for a newly happened transaction
n_h	the number of points in a two-dimensional space formed by the transactions in $Trans$ which are in the same brand-based product category as the newly happened transaction
nc_L	the capacity of an L-node in a *CMK-tree*
nc_I[3]	the capacity of an I-node in a *CMK-tree*

[1] m: The number of brand-based product category determines the size of *C-tree*

[2] n_i: For m brand-based product categories, we have $\sum_{i=1}^{m} n_i \leq n$. This is because one point may represent a set of repeated transactions that occurred on a day (see *Case 2* in Section 5.1.1.2). In practice, $\sum_{i=1}^{m} n_i$ may be much less than n.

[3] nc_I: The difference of node capacity between I-I-node and L-I-node is ignored in our discussion.

Proof: Now, let us go back to the CTT computation algorithm given in Section 5.2, where the method of computing the left border VRA and the right border VRA is adopted while answering a CTT query. In order to clearly understand Property 5, we first examine the performance of computing each VRA. Property 4 has illustrated that the transaction amount ranges and transaction time ranges of the records in the I-nodes in an *MK-tree* do not intersect each other. Thus, when computing a VRA, the search traverses from top to bottom in an *MK-tree* until reaching the layer of L-I-nodes. Then, one or several corresponding records in L-I-nodes are chosen. Finally, both the *aB*$^+$-*trees* and the leaf nodes pointed to by these records are searched. To sum up, the structure of *MK-tree* achieves logarithmic time cost ($O(\log n)$) for computing each VRA. Hence, to answer a buyer's CTT queries for a specific brand-based product category, the query of *CMK-tree* is almost linear. Also, this property has been demonstrated in the experiments, the results of which are to be introduced in Section 5.4.

When a buyer performs *"roll-up"* operations, the search iteratively performs from *Case 1* to *Case 3* introduced in Section 4.2.2.4 in the descendants of the current R-node, and finally one or several *MK-trees* are selected. The search then continues in the selected *MK-trees* twice for computing the left VRA and the right border VRA, respectively. Note that the number of *MK-trees* may be much less than m in practice. Therefore, the process has the linearithmic time cost ($O(n \log n)$) in total. However, the *CMR-tree* has better performance than all three approaches that were proposed in Chapter 4 (see Section 5.4). This is a significant advantage in answering CTT queries.

5.3.2 Theorems and Proofs

Next, we analyse the space utilisation of the *CMK-tree*, which is important to evaluate disk-based index schemes. We adopt the same analysis method as in [65] and consider the predictability of space utilisation, i.e. minimum space utilisation of each node.

Lemma 1: The minimum space utilisation is no less than $\frac{nc_L}{2}$ for an L-node; The minimum space utilisation is no less than $\frac{nc_I}{3}$ on average for an I-node.

Proof: Let t_1, t_2 and t_3 be three different time periods. Suppose an initial state that all the records in the L-node Ln_1 have the same transaction time t_1. For a new record with the transaction time t_2 to be inserted in Ln_1, there are two cases. (1) If space utilisation of Ln_1 is no less than $\frac{nc_L}{2}$, a new L-node Ln_2 is established. (2) Otherwise, the new record with the transaction time t_2 is inserted into Ln_1 until it overflows. Then, the node Ln_1 splits into a new Ln_1 and a new L-node Ln_2. The space utilisation of each generated L-node is still no less than $\frac{nc_L}{2}$. All the records in the two L-nodes are within the same time range $[t_1, t_2]$. In this case, if the new records with the transaction time t_2 continue to be inserted in an L-node, either Ln_1 or Ln_2 is selected as the targeted L-node depending on the transaction amount of the new record. Note that both Ln_1 and Ln_2 might split again during insertion, but the minimum space of any generated new L-nodes is no less than $\frac{nc_L}{2}$, and the records maintained in each node are within a time range $[t_1, t_2]$. The above operations are repeated until a record with the transaction time t_3 is inserted. This is because a new L-node is established for that record. So, the minimum space utilisation is no less than $\frac{nc_L}{2}$ for an L-node. In fact, except the L-nodes with the minimum space utilisation no less than $\frac{nc_L}{2}$, there is at most one L-node with the space utilisation less than $\frac{nc_L}{2}$. In particular, this specific L-node includes the records that are most recently inserted. For example, the newly generated L-node maintains only one record with the transaction time t_3.

To estimate the space utilisation of I-nodes, we consider the worst case. Let t_1, t_2, t_3 and t_4 be four different time periods. Still, we suppose an initial state that a full I-node In_1 with nc_I records includes only one record with the transaction time range $[t_1, t_2]$. The rest of the records in In_1 have the same transaction time range $[t_3, t_4]$. If a new record with a transaction time range $[t_3, t_4]$ to be inserted in the I-node In_1, the node In_1 overflows. Then, it splits into a new In_1 maintaining the one record with the transaction time range $[t_1, t_2]$ and another full I-node Ln_2. All the records in node Ln_2 have the same time range $[t_3, t_4]$. In such a case, if a new record with the transaction time range $[t_3, t_4]$ continues to be inserted in an I-node, the node Ln_2 is selected as the targeted I-node. Then, the node In_2 overflows and splits into two I-nodes according to

the transaction amount ranges of all the records it contains. Hence, due to continuous splits of I-nodes, the records in the original full I-node In_1 are distributed in three different I-nodes. Therefore, we can conclude that, in the worst case, the minimum space utilisation is no less than $\frac{nc_I}{3}$ on average for an I-node.

Lemma 2: The height of an *MK-tree* in the *CMK-tree* is at most $\lceil \log_{\lceil \frac{nc_I}{3} \rceil}^{\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor} \rceil + 1$ $(\forall i \in [1, m])$. The number of *aB*$^+$*-trees* in the *CMK-tree* pointed by L-I-nodes is at most $\sum_{i=1}^{m} \lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor$.

Proof: First, we examine the height of an *MK-tree* that is built based on the transactions in a brand-based product category. For m *MK-trees* in a *CMK-tree*, we focus on analysing one of them. Suppose the past transactions in a brand-based product category form n_i $(\forall i \in [1, m])$ points in a two-dimensional space. In Lemma 1, we have proved that the minimum space utilisation is no less than $\frac{nc_L}{2}$ for an L-node. In addition, there may exist one L-node with space utilisation less than $\frac{nc_L}{2}$. Hence, the number of L-nodes in an *MK-tree* is at most $\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor$. The above L-nodes will be indexed by I-nodes, each of which has the minimum space utilisation no less than $\frac{nc_I}{3}$ on average. Hence, the height of an *MK-tree* in the *CMK-tree* is at most $\lceil \log_{\lceil \frac{nc_I}{3} \rceil}^{\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor} \rceil + 1$. Since the number of transactions traded by the seller S in m brand-based product categories has an imbalanced distribution, each *MK-tree* has a different height. The *CMK-tree* is an unbalanced tree.

Second, we examine the total number of *aB*$^+$*-trees*, each of which is pointed by a record in an L-I-node. There are at most $\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor$ L-nodes in an *MK-tree*, and each L-node is pointed by a record in an L-I-node. Thus, the number of *aB*$^+$*-trees* in an *MK-tree* is also $\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor$ at most. Therefore, for m *MK-trees* in the *CMK-tree*, the total number of *aB*$^+$*-trees* is at most $\sum_{i=1}^{m} \lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor$.

Theorem 1: In a *CMK-tree*, the number of nodes accessed by an insertion operation is $O(h) + O(\log_{nc_I}^{\frac{n_h}{nc_L}}) + 1$.

Proof: For an insertion operation in the *CMR-tree*, we consider the insertion cost in the worst case. As described in Section 5.1.3.1, an insertion operation first searches a path

in the *C-tree* based on the *C-hrchy* of the newly happened transaction. The number of accessed nodes is $O(h)$. Then, the insertion operation traverses an *MK-tree*. Lemma 2 illustrates that the height of an *MK-tree* in the *CMK-tree* is at most $\lceil \log_{\lceil \frac{nc_L}{3} \rceil}^{\lfloor \frac{n_i}{\lceil \frac{nc_L}{2} \rceil} \rfloor} \rceil + 1$ $(\forall i \in [1, m])$. Hence, the number of accessed nodes for inserting a newly happened transaction in an *MK-tree* is $O(\log_{nc_h}^{\frac{n_h}{nc_L}}) + 1$. As a result, the total number of accessed nodes is $O(h) + O(\log_{nc_I}^{\frac{n_h}{nc_L}}) + 1$ for an insertion operation.

5.4 Experiments on CMK-tree

In this section, we introduce the results of the experiments conducted on four large datasets, which compare the proposed *CMK-tree* with three existing approaches *eaR-tree*, *eaP-tree* and *eH-tree* [173] (also see Section 4.2) with regards to the aspects of both efficiency in CTT computation. Note that the effectiveness of our proposed trust vector based approaches has already been studied both analytically and empirically in Chapter 3. In particular, the trust vector based approach can reflect a seller's dynamic trustworthiness in different transaction contexts and identify risks potentially existing in a forthcoming transaction, thus outperforming single-value trust valuation methods [120, 154] and a prior trust vector based approach [149].

5.4.1 Datasets

As mentioned in Section 4.3.1, with eBay APIs, we have obtained detailed feedback and transaction data for up to 90 days of the sellers who are selling some popular products with the largest number of reviews. We finally selected two sellers S_1 and S_2 who had totally around 12,000 transactions (approx. 133 transactions per day) and 4,000 transactions (approx. 44 transactions per day) respectively within 90 days. The selection of S_1 and S_2 allows performing both *"roll-up"* operations in product category hierarchy when doing finer-grained analysis on a seller's transaction reputation.

Considering that only 90 days real transaction data of a seller can be obtained from

eBay, and the time range in a CTT query can be *"the latest 6 months"* or *'the latest 12 months"*, we generated four large synthetic datasets $\mathbf{SD_1(S_1)}$, $\mathbf{SD_2(S_1)}$, $\mathbf{SD_3(S_2)}$ and $\mathbf{SD_4(S_2)}$ based on the transaction data of eBay sellers S_1 and S_2. In each synthetic dataset, we expanded the time period of transactions and the daily volume of transactions so as to test the performance of our proposed approaches under the circumstances with exceptionally large volumes of transactions. Specifically, the above four synthetic datasets are further categorised into two types.

(1) **Type I** includes $\mathbf{SD_1(S_1)}$ for S_1 and $\mathbf{SD_3(S_2)}$ for S_2: For each Type I synthetic dataset, we first duplicated the transaction data of each seller 10 times on a given day and thus obtained the transaction data of 10 times as much as the corresponding real dataset. Then, we continued duplicating the newly obtained transactions data of 90 days for about three times for the rest nine months (actually $365-90 = 275$ days). E.g., the data of the $91st$ day duplicates the one of the $1st$ day. Consequently, with the initial transaction data of 90 days, we obtained the transactions of 12 months. As a result, two datasets contain about $480,000$ and $160,000$ transactions in total, respectively. Type I synthetic datasets guarantee that the proportion of each sold product is the same on a daily basis in both the synthetic dataset and the corresponding real dataset.

(2) **Type II** includes $\mathbf{SD_2(S_1)}$ for S_1 and $\mathbf{SD_4(S_2)}$ for S_2: For each Type II synthetic dataset, a transaction was randomly selected from the corresponding sellers eBay real dataset of 90 days. This process was repeated until the size of transaction data on a day within 90 days is 10 times as much as that on the same day in the real dataset. Then we duplicated the data of 90 days for 365 days (12 months). In each Type II synthetic dataset, basically the proportion of a transaction selling a product on a day or in a month is different to that in the corresponding real dataset.

5.4.2 Experiment Setup

The parameters used in the experiments are as follows: the same page size of 1KB applies to all five index schemes; it is 4 bytes for each of the transaction amount,

Table 5.3: List of selected products from two popular sellers

seller	selected products
S_1	*Apple iPod nano 16GB (mc696ll/a)* at a price of \$150, *Dell Laptop (Inspiron i17rv-3529dbk)* at a price of \$650, *Canon Powershot Digital Camera (sx40 hs)* at a price of \$380
S_2	*Canon EOS DSLR Camera (T3i)* at a price of \$670, *Kodak Pocket Video Camera (Zi8)* at a price of \$240, *Brother Laser Printer (HL-2220)* at a price of \$90

transaction time, $count\_r$ and sum_r in a record and the *C-value* is of 8 bytes. In addition, each approach is implemented using VC++ 6.0 running on a Lenovo Y560 laptop with an Intel Core i5 CPU (2.20GHz), 2GB RAM, Windows 7 Professional operation system and MySql 5.1.35 relational database.

In contrast to Section 4.3.2, in order to intuitively reflect CTT query performance, we measure the CTT values computation time instead of the number of accessed nodes in the experiments. For each seller, we generated the corresponding queries on either *Transaction Item Specific Trust* (TIST) or *Product Category based Trust* (PCT), covering three transaction dimensions (denoted as **3D CTT queries**), and the queries on *Similar Transaction Amount based Trust* (STAT), covering two transaction dimensions (denoted as **2D CTT queries**).

To generate *3D CTT queries*, based on eBay datasets, we first selected 5 popular products traded by the sellers, each of which has two characteristics: (1) *"roll-up"* operations can be performed continuously at least 3 times along a path in the product category hierarchy; and (2) each product category along the path contains a large number of transactions. Table 5.3 lists the selected products for two popular sellers. Then, we set the time range in CTT queries to be "*the latest 1 month*" and computed their TIST values. After that, for each of 5 selected products, *"roll-up"* operations were performed continuously 3 times along a path in the product category hierarchy to generate the *3D queries* on PCT. To generate the price range in each PCT query, we adopted a strategy to partition the transaction amount range of the current product category that is included in a PCT query. Specifically, as each product category in

the product category hierarchy maintains its corresponding transaction amount range $[ta_1, ta_2]$, we partitioned this transaction amount range into 3 equal intervals, and each interval is regarded as an input for the price range in a PCT query. Thus, each product category corresponds to 3 PCT queries with different price ranges. In total, there are 45 ($5 \times 3 \times 3$) *3D queries* on PCT values. For instance, the generated PCT queries for the product "*Apple iPod nano 16GB (mc696ll/a)*" sold by S_1 (see Table 5.3) are <product-category: "*Apple MP3 player (iPod)*", price-range: "$100-$200", time-range: "*the latest 1 month*" > (i.e. PCT at layer 5) and <product-category: "*MP3 player*", price-range: "$150-$300", time-range: "*the latest 1 month*" > (i.e. PCT at layer 4). Our experiments also tested the TIST and PCT queries at three different time ranges of *"the latest 3 months"*, *"the latest 6 months"* and *"the latest 12 months"*, respectively. Thus, there are totally 200 (i.e. $(45 + 5) \times 4$) *3D CTT queries* tested for each seller.

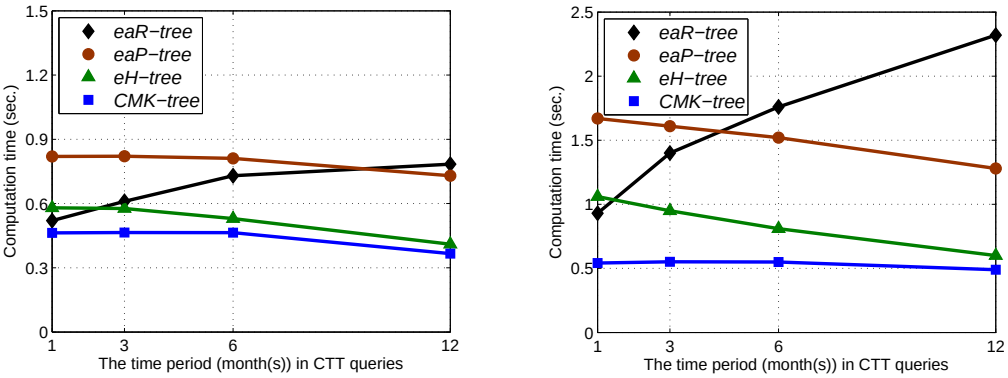
To generate *2D CTT queries*, 45 different price ranges for the above PCT queries were used. In the meantime, the experiments also tested 4 different time ranges of *"the latest 1 month"*, *"the latest 3 months"*, *"the latest 6 months"* and *"the latest 12 months"*. Thus, there are totally 180 (i.e. 45×4) *2D CTT queries* tested for each seller.

5.4.3 The Experimental Results

This section includes the results of the comparison between the *CMK-tree* and three existing approaches *eaR-tree*, *eaP-tree* and *eH-tree* proposed in Section 4.2, in terms of CTT values computation time, storage space consumption and the time for tree construction. The experimental results are obtained from the execution on four large synthetic datasets, and the computation time in answering CTT queries is the average of the results of 5 independent runs.

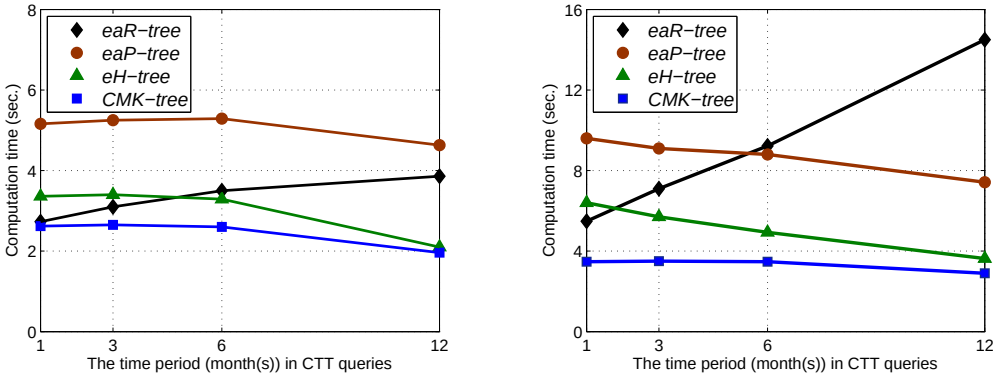
Fig. 5.6 and Fig. 5.7 plot the CTT values computation time of the *eaR-tree*, the *eaP-tree*, the *eH-tree* and the *CMK-tree* in *3D CTT queries* and *2D CTT queries*. First of all, from Fig. 5.6, we can observe:

(1) Compared with other approaches, the performance of the *eaR-tree* shows a



(a) 3D CTT queries on SD1(S1)

(b) 3D CTT queries on SD2(S1)



(c) 2D CTT queries on SD1(S1)

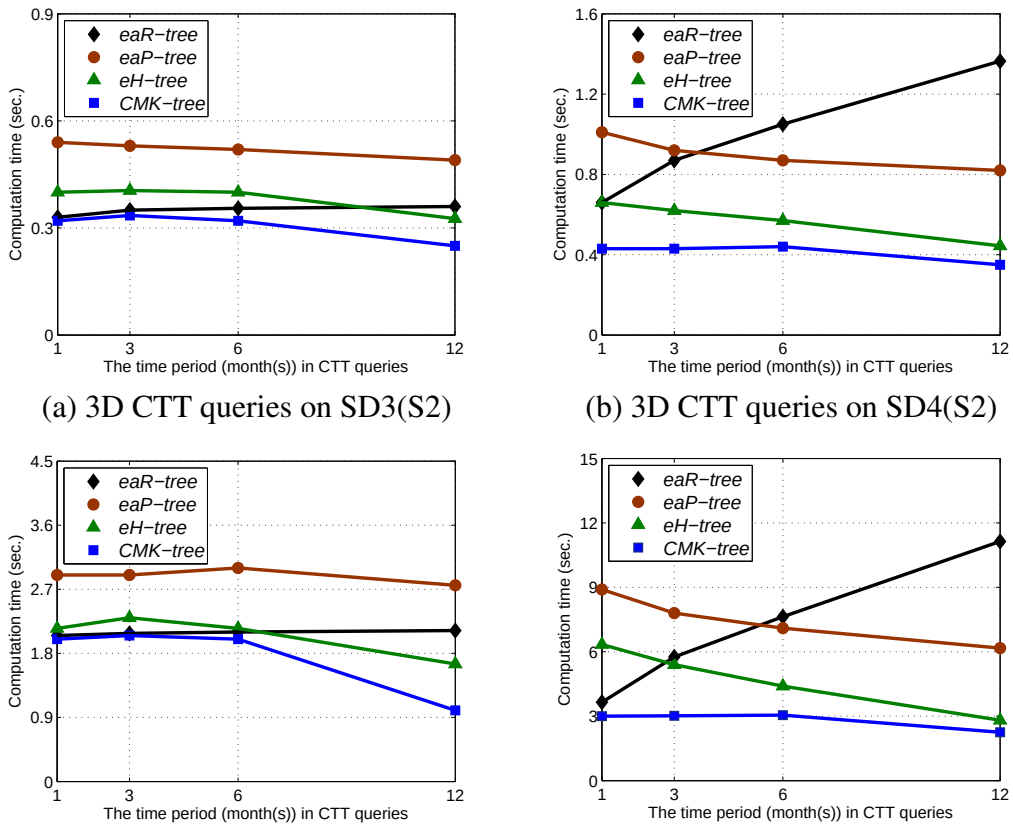
(d) 2D CTT queries on SD2(S1)

Figure 5.6: The query performance on two datasets *SD1(S1)* and *SD2(S1)* derived from seller S_1

different trend in CTT values computation time on both $SD_1(S_1)$ and $SD_2(S_1)$.

When the time range in a CTT query becomes larger, it means a larger query region for the *eaR-tree*, and the computation time increases almost linearly because more MBRs are overlapped by the expanded query range. In particular, if the time range in CTT queries is *"the latest 12 months"*, the *eaR-tree* has the worst performance in most cases.

In addition, a similar trend can be observed from the results of *eaP-tree*, *eH-tree* and *CMK-tree*, since they are all based on two VRA queries (see Section 2.4.2.2) to answer a CTT query. When the time range in a query expands, the computation time is stable or even in decline. In CTT queries, as the time range refers to the latest time



(a) 3D CTT queries on SD3(S2)

(b) 3D CTT queries on SD4(S2)

(c) 2D CTT queries on SD3(S2)

(d) 2D CTT queries on SD4(S2)

Figure 5.7: The query performance on two datasets *SD3(S2)* and *SD4(S2)* derived from seller S_2

period, the right border is always fixed to the time point "*now*". Correspondingly, the time of computing the right border VRA is also fixed for *eaP-tree*, *eH-tree* and *CMK-tree*. However, when the time range in a query expands (i.e. the left border shifts to the left), the time for computing the left border VRA decreases. This is because the above three approaches take advantage of the multi-version structure (see Section 5.1.2), and the versions with their starting time later than the left border of the time range will not be visited. Note that *eaP-tree* and *eH-tree* extend a multi-version *B-tree* (MVBT) [17], and *CMK-tree* extends a multi-version *"domain 0" 2-D-B-tree*. As illustrated in Fig. 5.6 and Fig. 5.7, if the time range in a query covers the longest time period, such as *"the latest 12 months"*, most versions of *"domain 0" 2-D-B-trees*

(see Fig. 5.3) have their starting time later than the left border of the time range. In such a case, only a few nodes need to be visited for computing the left border VRA. Consequently, the computation time for each of three approaches *eaP-tree*, *eH-tree* and *CMK-tree* drops to their minimum.

(2) The *CMK-tree* proposed in this chapter is superior in the efficiency of computing CTT values to the *eaR-tree*, *eaP-tree* and *eH-tree* on both $SD_1(S_1)$ and $SD_2(S_1)$.

The *eaR-tree* extends the *aR-tree* [104], and its performance greatly depends on the regions surrounded by the transaction time range and the price range in a CTT query. The *eaP-tree*, which extends the *aP-tree* [135], indexes all transactions and cannot essentially aggregate repeated transactions, leading to inferior performance. The *eH-tree*, which improves the *eaP-tree*, is faster than the *eaP-tree* in computing CTT values. The reasons are twofold. On one hand, the *eH-tree* adopts the *aP$^+$-tree* to reduce the time for computing the left border VRA. On the other hand, in *eH-tree*, the search is done in the fully ordered transaction amount space maintained in an additional *aB$^+$-tree* for computing the right border VRA (see Section 4.2.4.3). However, as the *eH-tree* is still based on the *eaP-tree*, it does not fundamentally resolve the problem existing in the *eaP-tree*. Moreover, the *eH-tree* takes longer time in constructing the aggregation index and consumes more storage space than the *eaP-tree* (see Fig. 5.8 and Fig. 5.9).

By contrast, the *CMK-tree* cannot only index each specific product traded in a time period, but also aggregate repeated transactions. More importantly, it delivers shorter and almost stable computation time for answering CTT queries. On average, for the 200 *3D CTT queries* based on Type I synthetic dataset $SD_1(S_1)$, the *CMK-tree* reduces computation time by 33.5% of the *eaR-tree*, by 44.8% of the *eaP-tree*, and by 16.2% of the *eaH-tree*; for the 180 *2D CTT queries*, the *CMK-tree* reduces computation time by 25.2% of the *eaR-tree*, by 51.5% of the *eaP-tree*, and by 18.8% of the *eaH-tree*. In addition, the improvement is more obvious on Type II synthetic dataset $SD_4(S_2)$. On average, for the 200 *3D CTT queries*, the *CMK-tree* reduces computation time by 66.7% of the *eaR-tree*, by 64.9% of the *eaP-tree*, and by 37.6% of the *eaH-tree*; for the 180 *2D CTT queries*, the *CMK-tree* reduces computation time by 63.2% of the

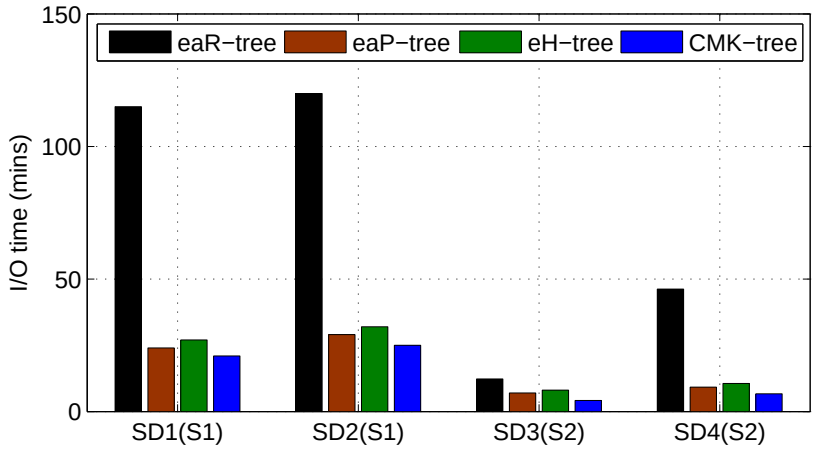


Figure 5.8: I/O time for different index schemes construction on four datasets

eaR-tree, by 61.8% of the *eaP-tree*, and by 35.4% of the *eaH-tree*.

From Fig. 5.7 plotting the results executed on $SD_3(S_2)$ and $SD_4(S_2)$ for S_2, we can draw the same conclusion as the one from the results on datasets $SD_1(S_1)$ and $SD_2(S_1)$ for S_1. On average, for the 200 *3D CTT queries* based on Type I synthetic dataset $SD_3(S_2)$, the *CMK-tree* reduces computation time by 12.2% of the *eaR-tree*, by 41.1% of the *eaP-tree*, and by 20.0% of the *eaH-tree*; for the 180 *2D CTT queries*, the *CMK-tree* reduces computation time by 15.6% of the *eaR-tree*, by 39.0% of the *eaP-tree*, and by 17.1% of the *eaH-tree*. On average, for the 200 *3D CTT queries* based on Type II synthetic dataset $SD_4(S_2)$, the *CMK-tree* reduces computation time by 58.0% of the *eaR-tree*, by 54.4% of the *eaP-tree*, and by 28.1% of the *eaH-tree*; for the 180 *2D CTT queries*, the *CMK-tree* reduces computation time by 59.8% of the *eaR-tree*, by 62.2% of the *eaP-tree*, and by 40.3% of the *eaH-tree*.

We also have tested the storage space consumption and I/O time for constructing these aggregation indexes on four synthetic datasets. As plotted in Fig. 5.8, the *CMK-tree* takes shorter time in construction than three existing approaches on four datasets (see Table 5.4 for detailed percentage). Overall, the proposed *CMK-tree* leads to 12%-84% time reduction in index construction. However, as plotted in Fig. 5.9, the *CMK-tree* consumes much storage space in some cases (6 out of 12 cases in Table 5.5)

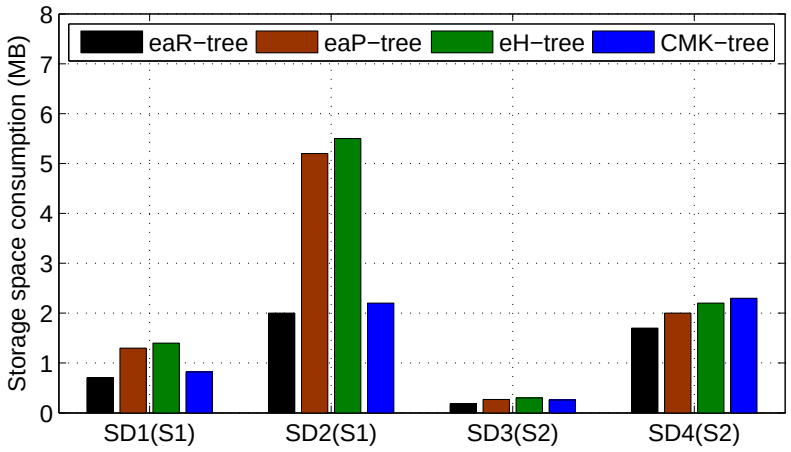


Figure 5.9: The storage space consumption for different index schemes on four datasets

even if it aggregates repeated transactions (see Table 5.5 for detailed percentage). For example, on $SD_4(S_2)$ dataset, the *CMK-tree* increases 5%-35% in storage space consumption. This is because, in order to achieve nearly linear query performance, the *CMK-tree* continuously records the transactions whose generated intervals along the Transaction Time dimension intersect with the left border of a rectangle using multiple *aB*$^+$*-trees*. Note that each rectangle is formed by a version of *"domain 0" 2-D-B-tree* as shown in Fig. 5.3.

Table 5.4: The comparison of constructing time between *CMK-tree* and the existing index schemes on four datasets

	$SD1(S1)$	$SD2(S1)$	$SD3(S2)$	$SD4(S2)$
CMK-tree vs. *eaR-tree*	18 : 100	21 : 100	34 : 100	16 : 100
CMK-tree vs. *eaP-tree*	88 : 100	86 : 100	60 : 100	73 : 100
CMK-tree vs. *eH-tree*	78 : 100	78 : 100	52 : 100	63 : 100

Table 5.5: The comparison of storage space consumption between *CMK-tree* and the existing index schemes on four datasets

	$SD1(S1)$	$SD2(S1)$	$SD3(S2)$	$SD4(S2)$
CMK-tree vs. *eaR-tree*	117 : 100	110 : 100	144 : 100	135 : 100
CMK-tree vs. *eaP-tree*	63 : 100	42 : 100	96 : 100	115 : 100
CMK-tree vs. *eH-tree*	59 : 100	40 : 100	87 : 100	105 : 100

5.5 Summary

In this chapter, we first summarised the limitations of existing approaches to two-dimensional (2D) Range Aggregate (RA) after being extended to solve CTT computation problem. To overcome all these limitations, we have proposed an *MK-tree* — an extended multi-version "domain 0" two-dimensional *K-D-B-tree* or "domain 0" *2-D-B-tree* [114]. Then, with the third dimension *C-tree* taken into account, a *CMK-tree* is formed. Four important and remarkable characteristics of the CMK-tree are summarised below:

(1) The *CMK-tree* does not index all transactions; rather, it aggregates repeated transactions on a given day, which sell the same product.

(2) The *CMK-tree* guarantees that each specific product can be indexed in order to compute the trustworthiness of a seller in selling a product.

(3) Each *MK-tree* in *CMK-tree* fully partitions a two-dimensional space into multiple nonintersecting rectangles. In order to achieve linear query performance, each point in the two-dimensional space generates an interval along the Transaction Time dimension. The intersections between the generated intervals and the corresponding rectangles are also recorded in *MK-tree*. A similar idea can be found in designing the BA-tree [167]. However, compared with *BA-tree*, *CMK-tree* adopts a different extension strategy — a multi-version structure [17], to effectively deal with the transaction-time model.

(4) Only two Vertical Range Aggregate (VRA) queries [135] are carried out to answer one CTT query based on the *CMK-tree*. This is more efficient than the *MVSB-tree* based on four dominance-sum queries.

After that, we provided an analytical study on the structure of *CMK-tree* as well as its query performance. Finally, our experiments have testified that the *CMK-tree* achieves nearly linear query performance in answering a buyer's CTT query. This is particularly significant to large-scale transaction data processing.

Chapter 6

Strategies for Storage Space Reduction in CTT Computation

Trust management is an important but complicated issue in e-commerce environments. In Chapter 3, we have pointed out that most existing trust evaluation models compute a single value to reflect the general trustworthiness of a seller without taking any transaction context into account. Consequently, consumers may be easily deceived by the potential risk existing in a forthcoming transaction, e.g., *context imbalance problem*.

In Chapter 4, we proposed Contextual Transaction Trust computation (termed as *CTT computation*) which is considered as an effective approach to resolve this problem. In particular, CTT computation is to compute a seller's reputation profile to indicate his/her dynamic trustworthiness in different products, product categories, price ranges, time periods, and any necessary combination of them. We also term this new model as *ReputationPro*. Nevertheless, in order to promptly answer a buyer's requests on the results of CTT computation, it requires additional storage space to store the precomputed aggregation results over large-scale ratings and transaction data of the seller. In practice, a seller usually has a large volume of transactions. Moreover, with significant increase of historical transaction data (e.g., one or two years), the size of that additional storage space can become much larger. Towards solving the above problem, in this chapter, we propose several strategies for storage space reduction in CTT computation. As a result, our proposed *ReputationPro* model will become scalable to large-scale e-commerce websites in terms of both efficiency and storage space

consumption.

This chapter is organised as follows. Section 6.1 presents a novel model for CTT computation with fixed storage space, and corresponding experiments to validate its effectiveness. The index scheme *CMK-tree*, proposed in Chapter 5, is considered as the most efficient approach to CTT computation. In particular, while answering a buyer's CTT queries for each brand-based product category, the *CMK-tree* has almost linear query performance. Then, in Section 6.2, we further extend the *CMK-tree* and propose a *CMK-tree*^{RS} approach to reducing the storage space allocated to each seller. Apart from the *CMK-tree*^{RS} approach, in fact, a fundamental idea of reducing the storage space is to delete aggregation results which are generated based on the ratings and transaction data from remote history, e.g., "*the 12 months ago*". Note that the deletions are also reasonable, as the earlier ratings are less important for evaluating a seller's recent behaviour. Therefore, in Section 6.3, we aim to propose deletion strategies for CTT computation based on the *CMK-tree*. With our proposed deletion strategies, the additional storage space consumption can be restricted to a limited range. Finally, we have conducted experiments to illustrate both advantages and disadvantages of the proposed deletion strategies. Section 6.4 summarises our work in this chapter.

6.1 A Novel Model for CTT Computation with Fixed Storage Space

In Chapters 4 and 5, several index schemes have been proposed for CTT computation, which adopt a single fine time granularity and aggregate the ratings by days. However, with continuous growth in transaction time (e.g., one year or two years) and significant increase of historical transaction data and ratings, the aggregation index with a single fine time granularity does not scale in terms of storage space. Here the *aggregation index* refers to the index containing some aggregates of ratings. In this section, we extend *HTA*^{FS} model [165] and propose a new solution CTT^{FS} model for CTT com-

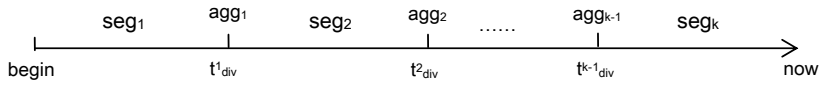


Figure 6.1: A general structure of *HTA*^{FS} model

putation. The CTT^{FS} model guarantees the fixed storage space for aggregation index as well as good performance in answering CTT queries. Then, we present the results of experiments conducted on both an eBay dataset and a large-scale synthetic dataset to validate our proposed structure and approach.

6.1.1 Hierarchical Temporal Aggregation with Fixed Storage Space (HTA^{FS})

In Section 2.4.2, we have reviewed the well-known approaches to two-dimensional Range Aggregate (RA) problem including range-temporal aggregation (point aggregate in two-dimensional space with one as the time dimension). In addition, a more general problem over two-dimensional RA is spatio-temporal aggregation (three dimensions with one as the time dimension). In the literature, a number of approaches have been proposed to solve the above aggregation problems [167, 134, 168]. However, they all use a single time granularity without space limitation.

In contrast to the above existing approaches, Zhang et al. [165] propose a Hierarchical Temporal Aggregation model with fixed storage space (denote as *HTA*^{FS}) to control storage space of aggregation index over data streams. Fig. 6.1 depicts the general structure of the *HTA*^{FS} model to deal with points aggregation in a one-dimensional space. A k-level time hierarchy, where $gran_1$ is at the coarsest time granularity (e.g., by days) and $gran_k$ is at the finest granularity (e.g., by seconds). Suppose that the *HTA*^{FS} model divides the time space $[begin, now)$ into k segments. Each segment seg_i ($i = 1, 2, ..., k$) maintains the corresponding aggregations with the time granularity $gran_i$. The term *begin* denotes the starting time and *now* denotes the increasing current time. New objects are inserted with the point "*now*" moving to the right in the

x-axis. The constraint for the *HTA*^{FS} model is that the size of available storage space is fixed. When the size of the total storage becomes more than a threshold S, older information is aggregated at a coarser granularity of time.

6.1.2 Specific Requirements in CTT Computation

Before introducing our proposed CTT^{FS} model, we first point out the specific requirements in CTT computation.

- First, in the traditional multiple dimensional aggregation problem, each dimension is linear. By contrast, the dimension of Product Category in CTT computation has a hierarchical structure.
- Second, based on the general structure of approaches for CTT computation given in Section 4.2.2, when inserting a new record (i.e. a new product category) into a *C-tree* (i.e. product category tree), it may affect the allocated storage space of all the subtrees that are external to the *C-tree*.

Now, let us consider two cases. Suppose that all the past transactions of a seller belong to $m(m \geq 2)$ brand-based product categories (see Fig. 3.1). Hence, m subtrees are generated corresponding to the m brand-based product categories. Note that the storage space used for storing aggregation index in the Product Category dimension, i.e., *C-tree*, will be ignored, and we assume the aggregation index formed by all the subtrees that are external to *C-tree* consume relatively larger storage space.

Case 1: If a newly occurred transaction belongs to any one of the m existing brand-based product categories, it is unreasonable to allocate average storage space to each subtree, since a seller would have imbalanced number of transactions in each brand-based product category;

Case 2: If a newly occurred transaction belongs to a new product category on which the seller has no prior transactions, it is necessary to allocate additional storage

space to store the aggregation index so as to know the trust value of the seller in this product category or the corresponding sub-category. As a result, the insertion operation surely affects space allocation for the previous m subtrees.

6.1.3 The Proposed CTT^{FS} Model

Our proposed CTT^{FS} model consists of two parts: dynamical storage space allocation (Algorithm 4) and hierarchical temporal aggregation (Algorithm 5).

6.1.3.1 Algorithms Description

Algorithm 4: Dynamical storage space allocation

This algorithm aims to appropriately allocate storage space for aggregation index formed by all the subtrees that are external to a *C-tree*.

As illustrated in subsection 4.2.2.1, in our proposed tree structure for CTT computation (see Fig. 4.2), each record in R-nodes of *C-tree* also maintains the number of ratings $count\_r$ and the sum of ratings sum_r in the corresponding product category. Suppose that the products sold by a seller in all past transactions belong to m brand-based product categories. Each brand-based product category includes $count\_r_j$ ($j \in 1, 2, ..., m$) transactions. Thus, $\sum_{j=1}^{m} count\_r_j$ equals the total number of all past transactions. If the total storage size allocated to a seller is S, we further consider space allocation in two cases as illustrated before:

- ***Case 1***: *The newly occurred transaction belongs to an existing brand-based product category*. For the existing m brand-based product categories, each subtree should be allocated $\frac{count\_r_j * S}{\sum_{j=1}^{m} count\_r_j}$ $(j \in 1, 2, ..., m)$ disk pages for storing aggregation index.

- ***Case 2***: *The newly occurred transaction belongs to a new brand-based product category*. In such a case, there are $m + 1$ brand-based product categories,

and each subtree should be allocated $\frac{count\_r_j * S}{\sum_{j=1}^{m+1} count\_r_j}$ $(j \in 1, 2, ..., m, m+1)$ disk pages for storing aggregation index.

Algorithm 5: Hierarchical temporal aggregation

However, each of the two cases in Algorithm 4 leads to a problem of compression of the allocated storage space (i.e. disk pages). Hence, the general idea of the *HTA*^{FS} model will be adopted to deal with the subtree, the size of which is larger than the allocated storage space. For the sake of simplicity, we assume the size of the storage space allocated for each subtree to store aggregation index is S_j. For m brand-based product categories, we have $\sum_{j=1}^{m} S_j = S$.

If the size of the aggregation index of a subtree becomes larger than the corresponding allocated space S_j, we need to have a further division. As illustrated before, each subtree that is external to product category hierarchy maintains the aggregates $count\_r$ and sum_r in Transaction Amount and Transaction Time dimensions. At the beginning, each subtree uses a single finer time granularity $gran_1$ in the Transaction Time dimension. When performing hierarchical temporal aggregation operations, we choose a new dividing time t_{newdiv}. Here, different from the *HTA*^{FS} model, the information larger than the size $S_j/2$ will be aggregated at a coarser time granularity $gran_2$. Consequently, the CTT^{FS} model forms two segments seg_1 and seg_2 with different time granularities in the Transaction Time dimension. In addition, more time granularities $gran_k$ $(k > 2)$ can be introduced to further compress the storage space.

For update operations in the CTT^{FS} model, we should first find all the transactions and corresponding ratings before t_{newdiv} in seg_2. Then, we standardise these transactions and the corresponding ratings at a coarser time granularity $gran_2$. Finally, we move standardised results to seg_1. If the seg_1 is not null, seg_1 is updated; otherwise we directly move them to seg_1. The above operations are similar to those defined in Algorithm 1 proposed in the *HTA*^{FS} model [165].

6.1.3.2 eaR-tree^{FS}– A New eaR-tree with Fixed Storage Space for CTT Computation

In this subsection, we use the *eaR-tree* as an example to explain how to apply our proposed CTT^{FS} model with fixed storage space in CTT computation. In Section 4.2.2, based on *aR-tree* [63, 104], we proposed an *eaR-tree* for CTT computation. In the *eaR-tree*, each subtree that is external to the *C-tree* is an *aR-tree* containing aggregates $count\_r$ and sum_r. Next, we illustrate how to construct an new *eaR-tree* with fixed storage space (*eaR-tree*^{FS}) for CTT computation.

Insertion: When inserting a newly occurred transaction into the *eaR-tree*^{FS}, Algorithm 4 first reallocates storage space for the aggregation index formed by each subtree (i.e., *aR-tree*). If the size of aggregation index of a subtree is larger than the allocated space, then Algorithm 5 will be applied to restrict the storage space. For the sake of simplicity, in our work, *eaR-tree*^{FS} only adopts 2-level time granularities and sets *the coarse time granularity at the month level* and *the fine time granularity at the day level*. As a result, different from original *eaR-tree*, each subtree in *eaR-tree*^{FS} is divided into 2 *aR-trees* logically with different time granularities (seg_1 and seg_2). Therefore, for m brand-based product categories, there are at most $2m$ *aR-trees* with limited storage space in the complete *eaR-tree*^{FS} model.

Query: To answer a CTT query, in the new structure *eaR-tree*^{FS}, the search is first conducted on the *C-tree* (the product category tree). However, there may be an additional query in a subtree of *eaR-tree*^{FS} that is external to the product category hierarchy depending on the query range in the transaction time dimension. That is because a subtree of *eaR-tree*^{FS} may logically have two *aR-trees* with different time granularities. For example, if a CTT query in transaction time dimension is within range $[t_{newdiv}, now)$, only an *aR-tree* in the subtree needs to be searched in corresponding brand-based product category; otherwise we need to search two *aR-trees*. In the following section, we will illustrate that the number of accessed nodes can be significantly reduced in our proposed new structure when answering a CTT query.

6.1.4 Experiments

In this section, we introduce the experiments conducted on an eBay dataset and a large-scale synthetic dataset, which aim to evaluate the efficiency of our proposed model.

6.1.4.1 Experiment Setup

We used the datasets depicted in Section 4.3.1. For sake of simplicity, we only selected the seller S_1 who had the largest trading volume among these popular sellers. S_1 had around 12000 transactions (approx. 4000 transactions per month) in total within 90 days. The products sold by S_1 distribute in multiple product categories; but most products are in the category of '*Information, Communication and Media technology*' (see Fig. 3.1). Meanwhile, our experiments are also conducted on the synthetic dataset $\boldsymbol{SD_1}$ based on S_1. $\boldsymbol{SD_1}$ contains transaction data for a 12 month period, and there are about $480,000$ transactions in total. $\boldsymbol{SD_1}$ guarantees that each product sold in a year has the same proportion of occurrences as that in the eBay dataset.

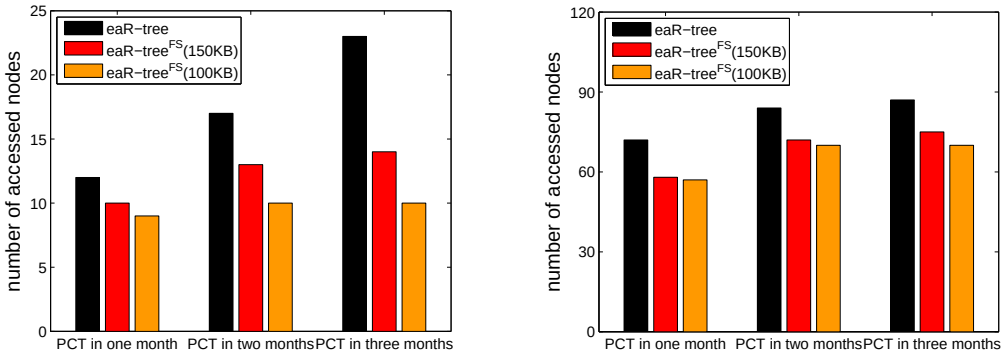
Moreover, all the parameters and experimental environments are the same as that used in Section 4.3.1. Considering S_1 had numerous transactions in the product categories '*Apple MP3 player (iPod)*' at layer 5 and '*Audio device*' at layer 3 in the product category hierarchy (see Fig. 3.1). Thus, we assume a scenario that a buyer plans to buy an '*Apple mc526ll/a iPod nano 16GB*' for about $\$150$ from this seller, and then s/he specifies and adjusts the 'product category' at the above two layers in product category hierarchy (i.e. PCT values in our proposed trust vector).

6.1.4.2 Experimental Results

Before presenting the experimental results, we need to emphasise that aggregation index cannot be compressed without any restriction, i.e., the size of the aggregation index for *eaR-tree*^{FS} should be larger than the size of *by-month* aggregation in the transaction time dimension for *eaR-tree*. Meanwhile, the size of aggregation index for *eaR-tree*^{FS} should be smaller than the size of the original *eaR-tree* (aggregation

Table 6.1: The size of aggregation index for the *eaR-tree* aggregated *by-day* and the *eaR-tree* aggregated *by-month*

	eaR-tree (by day)	*eaR-tree* (by month)
The size of aggregation index over the eBay dataset	205 KB	75 KB
The size of aggregation index over the synthetic dataset	2800 KB	250 KB



(a) PCT at layer 5

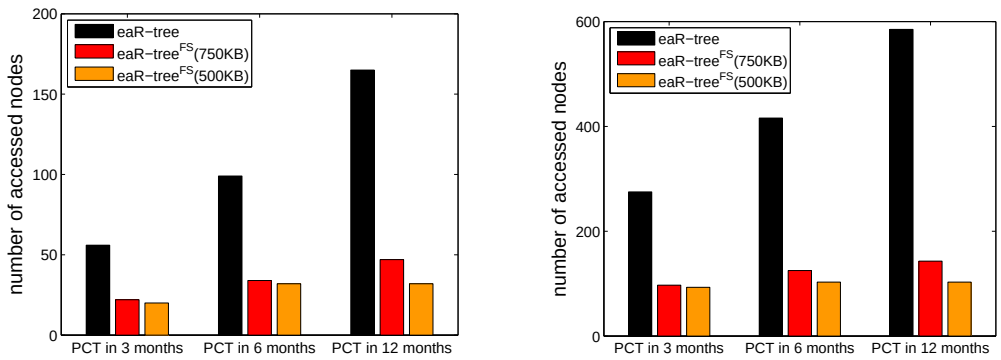
(b) PCT at layer 3

Figure 6.2: The performance of our proposed model over eBay dataset

by-day). Table 6.1 lists the sizes of aggregation index of the *eaR-tree* aggregated *by-day* and the *eaR-tree* aggregated *by-month* over eBay dataset and synthetic dataset, respectively.

Results on the eBay dataset: When computing the value of PCT on the eBay dataset, we also assume that the buyer specifies a transaction amount range of [$100,$200], and time ranges of "*the latest one month*", "*the latest two months*" and '*the latest three months*', respectively. For the *eaR-tree*^{FS} model, according to Table 6.1, we set the fixed storage space S to be 100 KB and 150 KB, respectively.

As shown in Fig. 6.2, for all three tree structures, when the time range in a query becomes larger, the number of accessed nodes increases linearly. This is in line with the nature of the original *aR-tree* [104]. As the amount of space allocated to the *eaR-tree* decreases, i.e., from *eaR-tree (150 KB)* to *eaR-tree (100 KB)*, more transactions



(a) PCT at layer 5 (b) PCT at layer 3

Figure 6.3: The performance of our proposed model over synthetic dataset

and ratings in history are aggregated by *month*. Thus, the complete index size becomes smaller, and the corresponding number of accessed nodes decreases. On average, compared with *eaR-tree*, the number of accessed nodes for computing PCT at layer 5 (see Fig. 6.2(a)) decreases by 29% for *eaR-tree^{FS}(150 KB)* and 44% for *eaR-tree^{FS}(100 KB)*. The number of accessed nodes for computing PCT at layer 3 (see Fig. 6.2(b)) decreases by 16% for *eaR-tree^{FS}(150 KB)* and 19% for *eaR-tree^{FS}(100 KB)*.

Results on the synthetic dataset: We also conducted the same experiments on the synthetic dataset $\boldsymbol{SD_1}$, which contain transactions distributed over a 12 month period. For computing the PCT, we assume that the buyer specifies time ranges of "*the latest 3 months*", "*the latest 6 months*", and "*the latest 12 months ago*", respectively. For the *eaR-tree^{FS}* model, according to Table 6.1, we set the fixed storage space S to be 500 KB and 750 KB, respectively.

From the results plotted in Fig. 6.3, we can draw the same conclusion as the experiments on the eBay dataset. On average, compared with *eaR-tree*, the number of accessed nodes for computing PCT at layer 5 (see Fig. 6.3(a)) decreases by 68% for *eaR-tree^{FS}(750 KB)* and 74% for *eaR-tree^{FS}(500 KB)*. Similarly, the number of accessed nodes for computing PCT at layer 3 (see Fig. 6.3(b)) decreases by 71% for *eaR-tree^{FS}(750 KB)* and 76% for *eaR-tree^{FS}(500 KB)*.

Summary: In CTT^{FS} model, the past transactions and ratings are aggregated at differ-

ent time granularities, which provides a trade-off between aggregation hierarchies and storage space. From the results of the experiments conducted on both an eBay dataset and a synthetic dataset, we can conclude that the new model not only restricts the allocated storage space, but also significantly reduces the number of accessed nodes in responding to buyers' CTT queries.

The CTT^{FS} model provides a meaningful attempt which takes advantage of *HTA*^{FS} model to reduce storage space consumption for CTT computation. However, it simultaneously brings some problems including operability in real applications and accuracy of CTT computation. In the following section, based on the *CMK-tree* proposed in Chapter 5 (i.e. the most efficient approach to CTT computation), the above problems will be discussed and resolved.

6.2 The Proposed CMK-tree^{RS}

As illustrated in Section 5.1, in the *CMK-tree*, transaction data and ratings are aggregated at the granularity of days. Though it consumes smaller storage space than the actual data, with significant increase of historical transaction data, the size of the *CMK-tree* will become much larger. In this section, we introduce a new approach *CMK-tree*^{RS}, which reduces storage space consumption for a *CMK-tree*, and offers great benefit to trust management with millions of sellers.

In Section 6.1, we have pointed out that the Hierarchical Temporal Aggregation model with fixed storage space (*HTA*^{FS}) [165] can be applied to CTT computation, in order to control the size of the storage space allocated to a seller for storing aggregation index. However, as a matter of fact, it is difficult to select the size of the fixed storage space, since the number of distinct products as well as the number of product categories in the transactions traded by different sellers are different. Now let us consider an example. Assume two sellers S_1 and S_2 have the same volume of transaction data over a period of time. The transactions traded by S_1 are widely distributed in multiple product categories, but seller S_2 has numerous repeated transactions belonging to only

a few product categories. For S_1, it is necessary to allocate a relatively large storage space to store aggregation index so as to obtain his/her trustworthiness in each product category and a corresponding sub-category. By contrast, for S_2, it is not necessary to allocate the storage space with the same size as S_1, because most transactions are repeated, leading to less storage space consumption for storing aggregation index.

In practice, the time range in CTT queries is usually *"the latest 1 month"*, *"the latest 3 months"*, *"the latest 6 months"*, or *"the latest 12 months"*. When adopting a *CMK-tree* for CTT computation, the above time ranges can be searched, but the *CMK-tree* consumes a large storage space as the aggregation granularity in the Transaction Time dimension is "days". Alternatively, if we aggregate ratings at a coarse time granularity of months, as depicted in Fig. 5.3(a), the number of points in a two-dimensional space formed by transactions further decreases, since more repeated transactions exist in a month than on a day. In such a case, the size of storage space consumed for a *CMK-tree* can be reduced to a large extent. However, such a coarse aggregation granularity leads to a serious problem regarding the accuracy of CTT computation. For instance, if the current time is *"10th August"*, to answer a buyer's CTT query with the specified time range as *"the latest 1 month"*, the result can be computed based on either the data of *"August"* only or the data of both *"August"* and *"July"*. However, neither way can guarantee the accuracy of CTT results. In the worst case, the ratings for computing a CTT value of *"the latest 1 month"* come from one day only or one month plus *"29 days"* (if one month contains *"30 days"*). On average, the CTT result comes from the ratings of 1 month $\pm \frac{1}{2}$month.

In the proposed approach *CMK-tree*^{RS}, a new strategy is adopted, which aggregates ratings with different time granularities for different time periods. In other words, in the *CMK-tree*^{RS}, *the ratings within the latest* t *days* are aggregated *at a fine time granularity of days*, and *the ratings of* t *days ago* are aggregated *at a coarse time granularity of weeks*. Taking into account the problem of accuracy of CTT values as mentioned in the above, our work provides a tradeoff between storage space consumption and accuracy. In addition, in practice, t is set to be *"90 days"*, considering the

typical time ranges in CTT queries are *"the latest 1 month"*, *"the latest 3 months"*, *"the latest 6 months"* and *"the latest 12 months"*. Therefore, for CTT queries regarding a seller's trustworthiness of *"the latest 1 month"* or *"the latest 3 months"*, there is no accuracy problem in the time dimension. For the queries of *"the latest 6 months"*, on average, ratings included in computation cover 6 months $\pm \frac{1}{2}$week. Likewise, in the case of *"the latest 12 months"*, on average, the ratings used in CTT computation cover 12 months $\pm \frac{1}{2}$week.

6.2.1 Construction of a CMK-tree^{RS}

This section describes the process of *CMK-tree*^{RS} construction. The algorithm for *CMK-tree*^{RS} construction adds the following operations to Algorithm 2, which is given in Section 5.1.3 and is used for building a *CMK-tree*:

• If $t_{now} - t_{begin} \leq t + 1$, insert the data of a newly happened transaction into the initial *CMK-tree* based on Algorithm 2. The term t_{begin} denotes the starting time and the term t_{now} denotes the current time. As new transactions happen everyday, in order to guarantee the ratings within the latest t days to be aggregated at a fine time granularity, firstly, it is necessary to perform insertion operations leading to the aggregations of $t + 1$ days. Then, the ratings of t days ago are moved to the aggregation index at a coarse time granularity. Note that this order of operations is necessary because if it is to first remove ratings for the transactions that happened t days ago, the ratings to be aggregated at a fine time granularity will include $t - 1$ days only, affecting the accuracy of CTT values.

• Otherwise, find all the transactions that happened no later than $t_{now} - t$ whose corresponding ratings are aggregated by days. Then, the Hierarchical Temporal Aggregation (*HTA*) operations are performed to form the *CMK-tree*^{RS}. Finally, continue to insert the data of a newly happened transaction into the new *CMK-tree*^{RS} based on Algorithm 2.

Next, we explain *HTA* operations in detail. In general, *HTA* operations aim to split one *MK-tree* in the initial *CMK-tree* into two *MK-trees* with ratings to be aggregated at different time granularities. To facilitate discussion, we term the two generated *MK-trees* as *MK-tree*^{day} and *MK-tree*^{week}, respectively. Suppose that the number of transactions traded by a seller at the time point t_{now} belongs to m brand-based product categories, namely, the initial *CMK-tree* at the time point t_{now} includes m *MK-trees*. In the meantime, there are m' $(m' \leq m)$ brand-based product categories which contain the transactions with the transaction time no later than $t_{now} - t$, and their corresponding ratings are aggregated with the time granularity of days. Then, the *HTA* operations are performed in the above m' *MK-trees*. For the sake of simplicity, we focus on introducing the following *HTA* operations within one of m' *MK-trees* contained in the initial *CMK-tree*.

- Remove the records maintained in an *MK-tree*, which can index the transactions that happened t days ago. Note that our work also guarantees the space utilisation of each node in the *MK-tree* during performing removal operation. For instance, as shown in Fig. 5.5(e), if the records $<[0,13],[1,3),0,Q_1>$, $<[13,30],[1,3),0,Q_2>$ and $<[30,50],[1,3),0,Q_3>$ in the I-node $In(L)_1$ are removed, the remaining records in $In(L)_1$ as well as the record in $In(L)_2$ are merged. In the meantime, the records in the I-node $In(I)_3$ are updated accordingly. In addition, in a more complex case, if the removed transactions with the transaction time falling into the time range of some records, a new *aB$^+$-tree* is generated[1]. For example, to remove the transactions that occurred before the time point 2 in Fig. 5.5(b), briefly speaking, we need to (1) delete the corresponding records in both Ln_1 and Ln_2, (2) merge the remaining records in these two nodes, (3) generate a new *aB$^+$-tree*, and (4) update the records in $In(L)_1$.

[1]The purpose of generating a new *aB$^+$-tree* is given in Section 5.1.3.2. Each generated *aB$^+$-tree* is to keep aggregation index of transactions in the same brand-based product category whose generated intervals along the Transaction Time dimension intersect with the left border of the corresponding rectangle.

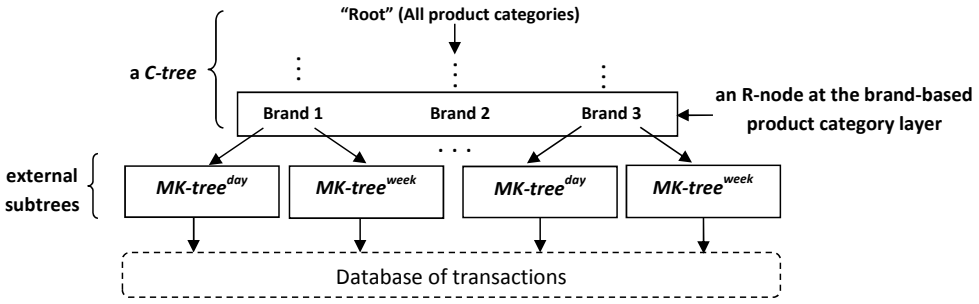


Figure 6.4: The *CMK-tree*^{RS} structure

- Standardise the data of transactions that happened t days ago. A standardization operation is to set the transactions occurred in the same week with the same x-coordinate. As stated before, the standardization operation essentially leads to a smaller size *CMK-tree*^{RS}. In Section 6.2.2, we will further explain how and why the *CMK-tree*^{RS} can also reduce the time of computing CTT values.
- Generate the second *MK-tree*, namely *MK-tree*^{week}, using the standardised results. Consequently, instead of having only one *MK-tree*, each subtree that is external to the *C-tree* has both an *MK-tree*^{day} and an *MK-tree*^{week} in the new *CMK-tree*^{RS}. Fig. 6.4 illustrates the general structure of the *CMK-tree*^{RS}. Note that if there already exist two *MK-trees* with aggregation index at different time granularities in an external subtree, the *MK-tree*^{week} is updated by inserting the standardised results.

6.2.2 CTT Computation Based on CMK-tree^{RS}

When answering a CTT query based on the new structure *CMK-tree*^{RS}, like Algorithm 3 given in Section 5.2, the *C-tree* is first searched. Then, it computes the left border VRA and the right border VRA in the subtrees that are external to the *C-tree*. Moreover, each external subtree in *CMK-tree*^{RS} has up to two *MK-trees* with aggregated ratings at different time granularities (see Fig. 6.4). Therefore, the computation of

the left border VRA and the right border VRA may also be performed in *MK-tree*^{day} and *MK-tree*^{week} respectively, depending on the query range in the Transaction Time dimension. If the specified time range in a CTT query is $[t_{now} - t, t_{now})$, only the *MK-tree*^{day} is searched; otherwise, the CTT computation algorithm searches both the *MK-tree*^{day} and the *MK-tree*^{week} in the corresponding external subtrees.

Now we introduce an example for further explanation. Suppose that the current time refers to the day of *"20th July"* and t is set to *"90 days"*. For an external subtree in the *CMK-tree*^{RS} that includes two *MK-trees*, the *MK-tree*^{day} maintains the aggregated ratings with the time range [*21st April, 20th July*) and the *MK-tree*^{week} maintains the aggregated ratings with the time range [*1st January, 21st April*). In this case, if a buyer's CTT query has *"the latest 6 months"* as the time range, the search for computing the left border VRA is performed on the *MK-tree*^{week}. By contrast, for a *CMK-tree* with an *MK-tree* as each external subtree, the *MK-tree* maintains the aggregated ratings for around 200 days in total (i.e. from *1st January* to *20th July*) with the time granularity of days. Though the *CMK-tree*^{RS} has two subtrees: *MK-tree*^{day} and *MK-tree*^{week}, they are much smaller in size. On one hand, the *MK-tree*^{day} includes the aggregates of ratings in a short period of time (i.e. 90 days *vs* 200 days). On the other hand, the *MK-tree*^{week} stores the aggregations for the remaining period of time at a coarse time granularity of "weeks" rather than "days". Therefore, based on the *CMK-tree*^{RS}, the time of computing both the left border VRA and the right border VRA can be reduced. The experiment results to be introduced in Section 6.2.3 also have illustrated both significant storage space reduction and performance improvement of the *CMK-tree*^{RS} in answering buyers' CTT queries.

6.2.3 Experiments

This subsection focuses on comparing the *CMK-tree* and the *CMK-tree*^{RS} in the aspects of both CTT computation time and storage space consumption.

As depicted in Section 5.4.1, the experiments are also conducted on four large

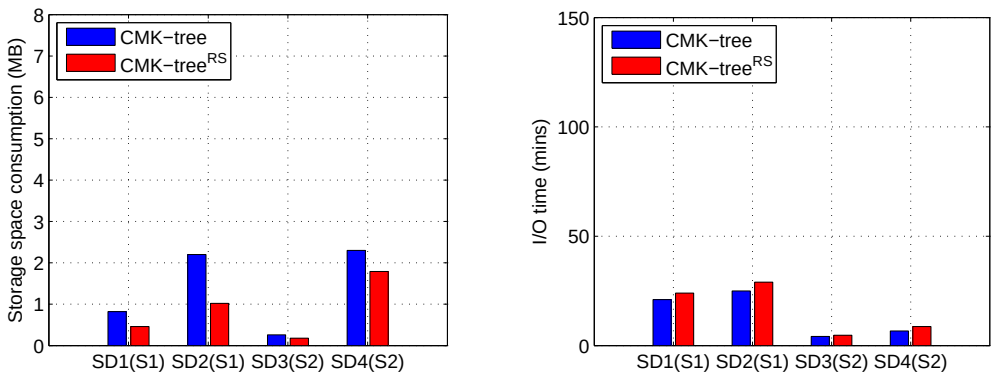


Figure 6.5: The storage space consumption and construction time for *CMK-tree* and *CMK-tree^{RS}* on four datasets

synthetic datasets $\mathbf{SD_1(S_1)}$, $\mathbf{SD_2(S_1)}$, $\mathbf{SD_3(S_2)}$ and $\mathbf{SD_4(S_2)}$ based on the transaction data of eBay sellers S_1 and S_2. For each seller, we generated the corresponding queries on either *Transaction Item Specific Trust* (TIST) or *Product Category based Trust* (PCT), covering three transaction dimensions (denoted as **3D CTT queries**), and the queries on *Similar Transaction Amount based Trust* (STAT), covering two transaction dimensions (denoted as **2D CTT queries**). Note that there are 200 *3D CTT queries* and 180 *2D CTT queries*. For fair comparison, all the parameters and experimental environments are the same as those used in Section 5.4.2.

Storage space reduction: The Table 6.2 lists the percentage of storage space reduction of the *CMK-tree^{RS}* compared to the *CMK-tree* (also see Fig. 6.5). Overall, the *CMK-tree^{RS}* reduces 23%-53% in storage space consumption on four synthetic datasets. On average, the reduction is about 38%.

Computation time improvement: the computation time of the two approaches are plotted in Figure 6.6 and Figure 6.7. In all cases, the *CMK-tree^{RS}* is faster than the *CMK-tree* in computing CTT values. On average, for the 200 *3D CTT queries* based on four datasets $SD_1(S_1)$, $SD_2(S_1)$, $SD_3(S_2)$ and $SD_4(S_2)$, the *CMK-tree^{RS}* reduces computation time by 10.4%, 18.0%, 11.8% and 12.4% of the *CMK-tree*, respectively; for the 180 *2D CTT queries*, the *CMK-tree^{RS}* reduces computation time by 25.0%,

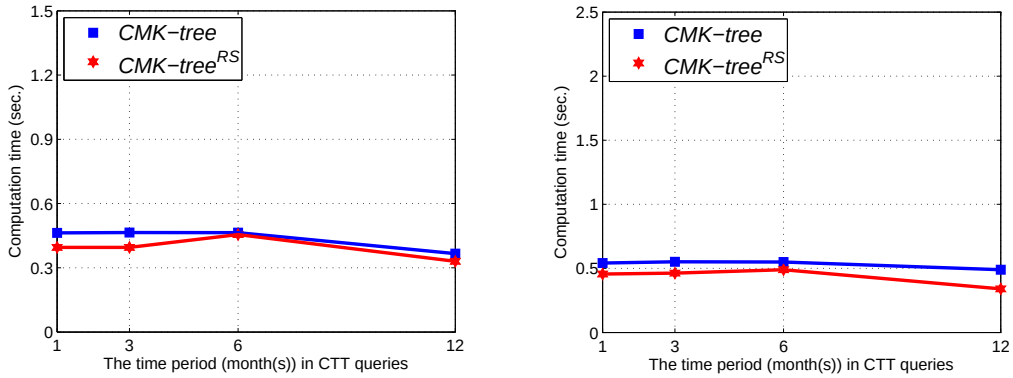
Table 6.2: *CMK-tree vs CMK-tree*^{RS}

databases	percentage of storage space reduction	accuracy of CTT values
$SD1(S1)$	44%	minimal difference: 0 ; maximal difference: 0.0015 error rate: 1.6%
$SD2(S1)$	53%	minimal difference: 0 ; maximal difference: 0.001 error rate: 1%
$SD3(S2)$	31%	minimal difference: 0 ; maximal difference: 0.028 error rate: 4.2%
$SD4(S2)$	23%	minimal difference: 0 ; maximal difference: 0.0034 error rate: 3.2%

29.3%, 24.1% and 30.0% of the *CMK-tree*, respectively.

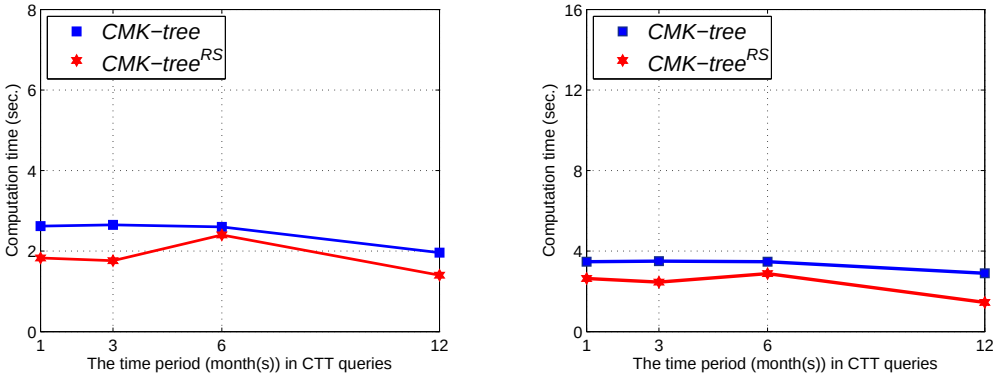
As explained in Section 6.2.2, compared with the *CMK-tree*, the search in the *CMK-tree*^{RS} for computing the left border VRA and the right border VRA is performed in two subtrees, i.e., *MK-tree*^{day} and *MK-tree*^{week}, but they are much smaller in size than the original *MK-tree* maintained in the *CMK-tree*. Thus, the search time can be reduced in computing two VRA queries. Note that if the time range in a CTT is *"the latest 1 month"* or *"the latest 3 months"*, only the *MK-tree*^{day} is searched for computing both the left border VRA and the right border VRA. Consequently, the *CMK-tree*^{RS} also delivers almost stable computation time for answering CTT queries at the above two time ranges. When the time range in CTT queries is *"the latest 6 months"*, the *MK-tree*^{day} is still searched for computing the right border VRA, but the *MK-tree*^{week} is searched for computing the left border VRA leading to a longer computation time[2]; therefore, we can observe that the computation time increases in this case for the *CMK-tree*^{RS}. However, if the time range in CTT queries is *"the latest 12 months"*, like the *eaP-tree*, the *eH-tree* and the *CMK-tree*, the computation time delivered by the *CMK-tree*^{RS} also drops to the minimum, since only a few nodes are visited for computing the left border VRA.

[2]The *MK-tree*^{day} maintains the aggregated ratings of the latest 3 months, and the *MK-tree*^{week} maintains the aggregated ratings of the remaining 9 months. Although 9 month transaction data and ratings are aggregated at a coarse time granularity of weeks in the *MK-tree*^{week}, it still has a larger size than *MK-tree*^{day}. Thus, the search in the *MK-tree*^{week} consumes more time than that in the *MK-tree*^{day}.



(a) 3D CTT queries on SD1(S1)

(b) 3D CTT queries on SD2(S1)



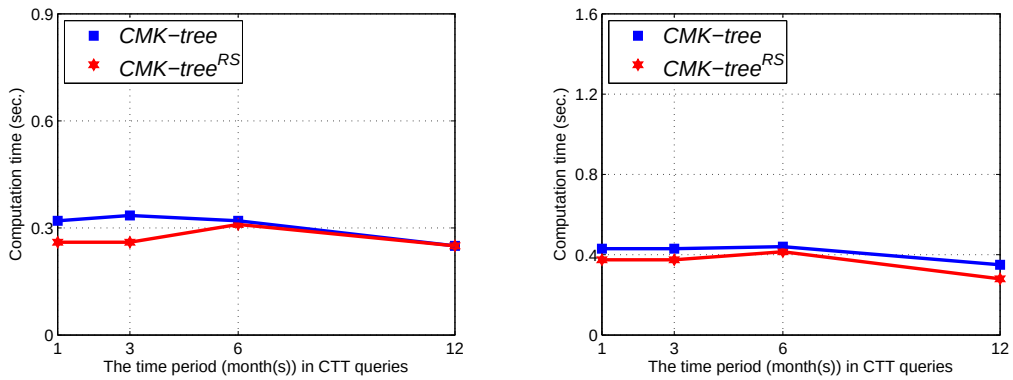
(c) 2D CTT queries on SD1(S1)

(d) 2D CTT queries on SD2(S1)

Figure 6.6: The query performance of *CMK-tree* and *CMK-tree^{RS}* on two datasets *SD1(S1)* and *SD2(S1)* derived from seller S_1

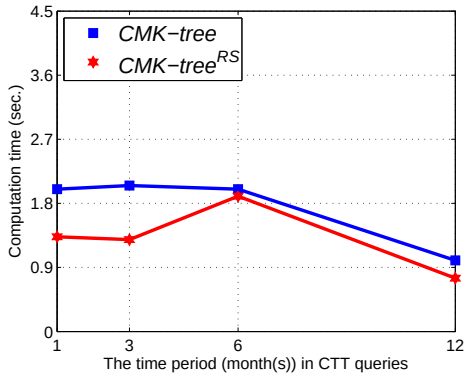
Accuracy of CTT values: In the meantime, as we have pointed out at the beginning of Section 6.2, the *CMK-tree^{RS}* reduces the storage space consumption at the expense of the accuracy of CTT values, which exists in the results of the CTT queries regarding *"the latest 6 months"* or *"the latest 12 months"*. Thus, we examined the differences of the results delivered by the *CMK-tree* and the *CMK-tree^{RS}* of the 95 CTT queries (i.e. 50 *3D CTT queries* and 45 *2D CTT queries*) regarding the above two time ranges, i.e., a totally 190 CTT queries are examined. Specifically, in Table 6.2, we list the maximal and minimal differences as well as the error rate for the 190 CTT queries on four datasets. Overall, the *CMK-tree^{RS}* leads to 0.001-0.028 as the maximal difference[3] in

[3]It can be expected that the maximal difference exists in computing the value of *3D CTT queries* with the parameter of product category at a low layer in the product category hierarchy, since a small

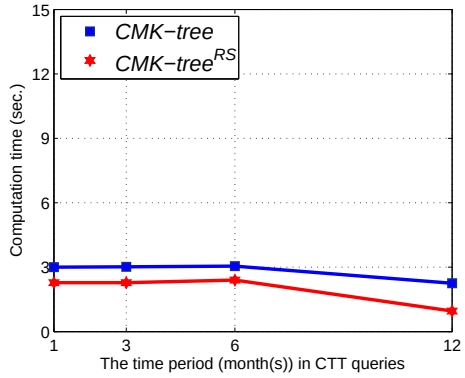


(a) 3D CTT queries on SD3(S2)

(b) 3D CTT queries on SD4(S2)



(c) 2D CTT queries on SD3(S2)



(d) 2D CTT queries on SD4(S2)

Figure 6.7: The query performance of *CMK-tree* and *CMK-tree*^{RS} on two datasets *SD3(S2)* and *SD4(S2)* derived from seller S_2

CTT values in $[0, 1]$ and 2.5% as the average error rate. Therefore, we conclude that the *CMK-tree*^{RS} brings a little loss to the accuracy of CTT computation with much gain in storage space reduction (23%-53% reduction) and computation time improvement.

6.3 The Deletion Strategies for CTT Computation

Generally speaking, in order to reduce storage space consumption for CTT computation, the approaches introduced in Sections 6.1 and 6.2 are to aggregate ratings and transaction data at different time granularity. Although storage space can be reduced

amount of ratings involve in the computation. When performing *"roll-up"* operations, more transaction data and ratings are involved in computation. As a result, the computation error rate can be reduced.

to some extent, they does not fundamentally resolve the problem of the large storage space consumption. Furthermore, it brings the problem of accuracy for computing CTT values.

Actually, compared with aggregating ratings and transaction data at different time granularity, a fundamental idea for storage space reduction is to delete index records that are generated based on ratings and transaction data from remote history (e.g., "*the 12 months ago*"). In other words, it is reasonable to generate the aggregation index based on the ratings and transaction data within a specified time period (e.g., "*the latest 12 months*"). In practice, the time range in CTT queries usually refers to the latest time period; therefore, the earlier ratings that are less important for evaluating a seller's recent behaviour can be eliminated. More importantly, it restricts the additional storage space consumption to a limited range.

As mentioned before, the index scheme *CMK-tree* is considered as the most efficient approach to CTT computation. In particular, while answering a buyer's CTT queries for each brand-based product category, the *CMK-tree* has almost linear query performance. With the new transactions of a seller occurred everyday, their corresponding transaction records are continuously added to the database in chronological order. In the meantime, the *CMK-tree* will be updated accordingly. However, in the following, we will explain that, in order to achieve nearly linear query performance, the deletion operations in the *CMK-tree* become quite complicated. Therefore, in this section, we propose three deletion strategies for the *CMK-tree*. As a result, the *CMK-tree* can be more effectively applied to large-scale e-commerce websites in terms of efficiency and storage space consumption for CTT computation.

6.3.1 The Multi-Version Structure

In the literature, the multi-version structure is an effective means to support aggregation operations along temporal dimension, e.g., *aP-tree* [135] and *MVSB-tree* [168]. A multi-version structure is to make *partial persistence* a basic (tree) structure. For ex-

ample, the *aP-tree* is an extended *Multi-Version B+-tree (MVBT)* [17] and the *MVSB-tree* is an *Multi-Version SB-tree* [161]. As stated before, the transaction records of a seller are continuously added to the database in chronological order which implies that the updates of generated index for CTT computation only affect the latest index records. Therefore, the design of index schemes for CTT computation can also adopt the multi-version structure.

In fact, without supporting insertions to the historical (old) versions is an important characteristic for the multi-version structure. However, it simultaneously makes the application of multi-version structure based index schemes be limited, because the new points do not necessarily arrive in chronological order for some cases. To solve this problem, Tao et al. propose the *double logarithmic method* [135] which supports both insertions and deletions to the "history" for multi-version structure. In particular, for deletions, *double logarithmic method* takes advantage of an invertible deletion strategy [101]. In Section 6.3.2, we will apply the idea of the invertible deletion strategy and propose a *'naive' deletion strategy* for CTT computation. In addition to this, we introduce two other deletion strategies: the *space-effective deletion strategy* and the *time-effective deletion strategy*.

6.3.2 Our Proposed Deletion Strategies

Next, let us consider the structure of a *CMK-tree* where multiple *MK-trees* are the major components. In Section 5.3, we have proved that the *CMK-tree* has almost linear query performance when answering a buyer's CTT queries for each brand-based product category, and this property largely depends on the well-designed *MK-tree*.

From the transformation plotted in Figure 5.3(b), in order to achieve linear query performance, each point[4] generates an interval along the *Transaction Time* dimension. Meanwhile, all versions of *"domain 0" 2-D-B-trees* index the points whose generated

[4]In CTT computation, one point at (t_i, ta_i) may represent a set of repeated transactions that occurred on a day t_i selling the same product with the same price ta_i. Thus, we use the term *'point'* rather than *'transaction'* for accurate description.

intervals intersect with the left border of them as well as aggregates of corresponding points. Thus, if the index records for points in a version of *"domain 0" 2-D-B-tree* are deleted, all its subsequent versions will be affected. For instance, in Figure 5.3(b), while deleting the index record in *MK-tree* for the point α_1 in version 1 of *"domain 0" 2-D-B-tree*, the updates are also performed in version 2 and version 3, since they no longer need to index α_1 whose generated intervals intersect with their left border.

Property 6: The insertions in an *MK-tree* only affect the latest version of *"domain 0" K-D-B-tree*, but the deletions in *MK-tree* affect all the subsequences of current version.

Clearly, due to Property 6, the deletions in the *CMK-tree* become quite complicated. Therefore, in the following, we will introduce three different deletion strategies for the *CMK-tree* so as to guarantee the generated aggregation index based on the ratings and transaction data within a specified time period, such as "*12 months*".

6.3.2.1 'Naive' Strategy

In [101], Overmars proposes a general invertible deletion strategy that periodically rebuilds the entire data structure. The main idea is that it has two insertion-only data structures, a *main* structure M and a *ghost* structure G. To insert an item, the insertion is performed in M. To delete an item, instead of removing it from M, the item is actually inserted to another structure G. Then, to avoid the size of M and G becoming much larger, the entire data structure is rebuilt periodically, i.e., building a new main structure M with the remaining items (the items are in M but not in G) and a new empty ghost structure G. The whole process is also termed as *global rebuilding*.

As mentioned in Section 6.3.1, Tao et al. propose a *double logarithmic method* [135] that adopts the idea of invertible deletion strategy to make an *aP-tree* fully dynamic, so that the *aP-tree* (a multi-version structure) can support insertions to the historical versions as well as the deletions. Here the discussion of the above dynamic data structure is beyond the scope of this paper, as the insertions are not performed in historical versions for CTT computation. In addition, we emphasise that the ghost structure is

not needed for the 'naive' strategy. Instead, we only need to periodically rebuild the entire *CMK-tree* for each seller. To ease the overhead of rebuilding, the time period can be set to *a month* or even longer. In other words, for the 'naive' deletion strategy, the generated aggregation index may include "*12 months plus one month*, i.e., *13 months*" (taking "*12 months*" as the specified time period). The following two steps are conducted continuously for the 'naive' strategy.

Step 1: Perform insertions to the *CMK-tree* until all the transactions in past $m + n$ days have completed;

Step 2: Perform rebuilding the entire *CMK-tree* based on the ratings and transaction data of a seller in latest m days. Then, execution resumes from Step 1.

Summary: The idea of the 'naive' strategy is simple, and it avoids complex deletion operations introduced in Property 6. Furthermore, this strategy can be easily extended to make the *CMK-tree* fully dynamic [135]. However, it consumes more storage space for storing *CMK-tree* that is generated based on a longer time period. Also, periodically rebuilding it for each seller is time-consuming.

Different from the 'naive' strategy, the deletions are performed by *days* for the other two strategies. Specifically, in real operations, after all new transactions in the current day have completed and the *CMK-tree* is updated accordingly, the index records for the transactions completed on the first day, e.g., *"12 months ago"*, in the *CMK-tree* should be deleted so as to maintain the transaction information of a seller in a specific time period, e.g., *"12 months"*.

6.3.2.2 Space-Effective Strategy

Basically, the space-effective strategy is to delete the index records in a *CMK-tree* directly. For the sake of simplicity, we take deleting the index record for the point α_3 as an example. Suppose that the point α_3 (see Fig. 5.3(b)) includes the information: *C-hrchy*$_{\alpha_3}$, ta_{α_3}, t_{first}, $count\_r_{\alpha_3}$ and $sum\_r_{\alpha_3}$ ($|count\_r_{\alpha_3}| \geq 1$, $|sum\_r_{\alpha_3}| \geq 1$).

Note that *C-hrchy*$_{\alpha_3}$ (defined in Section 3.3.1) represents the path in product category hierarchy to which the point α_3 belongs. As the products traded in the transactions are at the bottom of product category hierarchy, the value of *C-hrchy*$_{\alpha_3}$ equals the *C-value* in the corresponding brand-based product category. t_{first} represents time of the first day, e.g., *"12 months ago"*. The following five steps are conducted for the space-effective strategy:

Step 1: A top-down (from R-nodes to L-nodes) search is performed in the *CMK-tree* to find the index record for the point α_3 based on the information of t_{first}, *C-hrchy*$_{\alpha_3}$, ta_{α_3}. For Step 1, the detailed search process for locating an index record is given in Section 5.2.

Step 2: Remove the index record in an L-node for the point α_3. After removing the index record, the utilisation of corresponding node may be quite low, i.e. the node contains only a few records. In Step 4, we will improve the node utilisation via merging the sibling nodes.

Step 3: Update multiple *aB*$^+$*-trees*. In Property 6, we have explained that when the deletions are performed in an old version of *"domain 0" 2-D-B-tree*, all its subsequent versions are affected. That is because the information of intersections between the generated intervals and the corresponding rectangles also have to be updated. As introduced in Section 5.1.2, multiple *aB*$^+$*-trees* are built to maintain the information of intersections; therefore, they should be updated accordingly, i.e. removing and merging records in the *aB*$^+$*-trees*. Note that removing and merging records in an *aB*$^+$*-tree* are the same as those of a *B*$^+$*-tree* [15].

Step 4: Improve the nodes utilisation after removing some node records in Step 2. In Section 5.3, we have proved that the minimum space utilisation is no less than $\frac{nc_L}{2}$ for an L-node in a *CMK-tree* where nc_L is the capacity of an L-node. Thus, if an L-node utilisation is less than $\frac{nc_L}{2}$ after removing some records, the utilisation of its sibling node will first be checked. During the process, if the

sum utilisation of current L-node and its sibling node is more than nc_L, these two nodes will not be merged. Otherwise, the records in these two L-nodes need to be merged. Meanwhile, update and merge the records in an I-node (as well as the ancestors of this I-node) which point to that two L-nodes.

Step 5: Update the ranges and aggregates from bottom up accordingly, i.e. update $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$, sum_r maintained in corresponding R-nodes and L-nodes (see Fig. 5.1(a) and (b)). Here, in contrast to insertions where the *CMK-tree* is updated from top (R-node, the product category root) to bottom (L-node, pointing to the transaction record stored in the database), the updates are performed from bottom up for deletions.

Finally, the above five steps are performed in all the index records in the *CMK-tree* for the points with the same transaction time t_{first}. In practice, in order to ease the overhead, Step 4 and Step 5 can be executed when all the index records for the transactions within a brand-based product category, which meet the requirements, have already been deleted.

Summary: An important advantage of space-effective deletion strategy is that there is no extra storage space needed during deletion operations. However, inevitably, this strategy is still time-consuming, since it does not fundamentally resolve the problem caused by Property 6.

6.3.2.3 Time-Effective Strategy

Generally speaking, both 'naive' deletion strategy and space-effective deletion strategy are time-consuming, as they do not provide solutions to the problem caused by Property 6. In particular, for 'naive' deletion strategy, the deletions themselves have been avoided; for space-effective deletion strategy, searching and updating the corresponding *aB$^+$-trees* in Step 3 are time-consuming. To solve this problem, a basic idea is to add indexes to manage the multiple *aB$^+$-trees* separately which can improve the performance of deletion operations.

Now, let us consider the structure of an I-node depicted in Fig. 5.1(b), where each record in an I-node maintains: $[ta_{min}, ta_{max}]$, $[t_{min}, t_{max}]$, $count\_r$, sum_r and two pointers. Note that one pointer points to an L-node, and the other points to an *aB$^+$-tree*. $t_{min} : [ta_{min}, ta_{max}]$ implies the left border of the rectangle surrounded by a record; $[ta_{min}, ta_{max}]$ is transaction amount range within which the *aB$^+$-tree* is built; $count\_r$ and sum_r are the aggregates of points (transactions) whose generated intervals intersect with $t_{min} : [ta_{min}, ta_{max}]$.

In the time-effective deletion strategy, we will design and add a new index to manage that multiple *aB$^+$-trees* pointed by the I-nodes. Specifically, the new index is built in following ways:

1) *The pairs $< t_{min}, ta_{max} >$ in the record of an I-node is adopted as the unique ID to identify each aB$^+$-tree.* Therefore, each record in the new index has the form $< t_{min}, ta_{max}, pointer >$, where the pointer points to an *aB$^+$-tree*.

2) *The new index is built in each brand-based product category.* In a *CMK-tree*, the *aB$^+$-trees* are included in the *MK-trees* which are external to a *C-tree* (see Figure 4.2). Thus, each record in an R-node at the brand-based product category layer, i.e. the bottom of the *C-tree*, simultaneously points to the new index.

3) *The new index can be organised to form a 2-D matrix.* Fig. 6.8 shows an example of the 2-D matrix. When searching the *aB-trees* that are effected by a deletion, they are easily located via this 2-D matrix. For instance, to delete the point α_3 with $< t_{first}, ta_{\alpha_3} >$ ($t_{first} < t_{min_1}$, $ta_{max4} < ta_{\alpha_3} < ta_{max2}$), the *aB-trees* with IDs $< t_{min1}, ta_{max2} >$, $< t_{min2}, ta_{max5} >$, $< t_{min3}, ta_{max6} >$, $< t_{min4}, ta_{max9} >$ are selected. Then, the corresponding *aB-trees* will be updated accordingly. Note that as the time increases, the 2-D matrix need to be updated.

Except for Step 3 in space-effective strategy, the time-effective strategy has the same deletion process as space-effective strategy. However, after introducing the new index, the performance of deletion operations can be improved. Also, their performance will be demonstrated in the experiments introduced in Section 6.3.3.

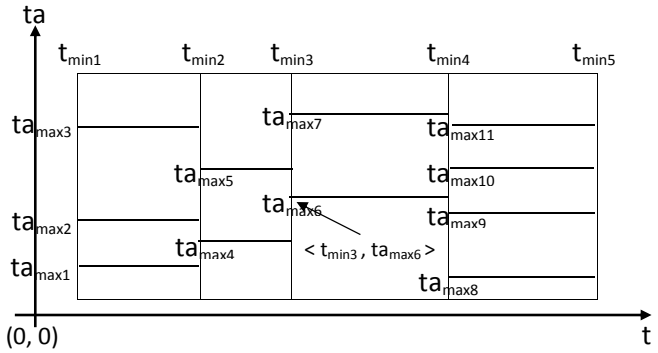


Figure 6.8: The 2-D matrix formed by the new index

Summary: For the time-effective deletion strategy, the new index is added so as to reduce time-consuming for deletions in a *CMK-tree*. In this way, the information of intersections between the generated intervals and the corresponding rectangles (see Fig. 5.3(b)) will be managed separately. To some extent, this strategy can be regarded as a trade-off between time consumption and storage space consumption.

6.3.3 Experiments

This section evaluates the advantages and disadvantages of the three proposed deletion strategies.

6.3.3.1 Experiment Setup

The experiments conducted on four large-scale synthetic datasets are also used in Section 5.4.1 and Section 6.2.3. The four large synthetic datasets $\mathbf{SD_1(S_1)}$, $\mathbf{SD_2(S_1)}$, $\mathbf{SD_3(S_2)}$ and $\mathbf{SD_4(S_2)}$ are generated from two real-world datasets containing transaction data of "*12 months*". Furthermore, in order to evaluate our proposed deletion strategies, we further extended four synthetic datasets by randomly selecting *a month* of transactions in each dataset as transactions of "*12 months ago*". In this way, we obtained another four datasets $\mathbf{SD_1(S_1)'}$, $\mathbf{SD_2(S_1)'}$, $\mathbf{SD_3(S_2)'}$ and $\mathbf{SD_4(S_2)'}$ containing transaction data of "*13 months*". To sum up, the purpose of generating these synthetic datasets is to test the performance of three deletion strategies under the circumstance

with a large volume of transactions in both each day and a long time period.

In addition, the experiments aim to compare the three deletion strategies from two aspects: deletion time consumption and storage space consumption. The *CMK-tree* as well as the deletion strategies are implemented using C++ running on a Lenovo Y560 laptop with an Intel Core i5 CPU (2.20GHz), 2GB RAM, Windows 7 Professional operation system and MySql 5.1.35 relational database.

6.3.3.2 Results

Storage space consumption: Table 6.3 presents the additional storage space consumption of three deletion strategies based on four datasets, i.e. from $SD_1(S_1)$ to $SD_4(S_2)$.

(a) For the 'naive' deletion strategy, for instance, the storage space consumption based on $SD_1(S_1)$ equals to the size of the *CMK-tree* built on $SD_1(S_1)'$ (including *13 months* transaction data) subtracts the size of the *CMK-tree* built on $SD_1(S_1)$ (including *12 months* transaction data). Overall, it leads to 0.02 MB–0.23 MB additional storage space consumption;

(b) For the space-effective deletion strategy, as illustrated in Section 6.3.2.2, there is no extra storage space required during deletion operations;

(c) For the time-effective deletion strategy, the storage space consumption equals to the size of added new index. Overall, it leads to 10 KB–150 KB for constructing additional new index.

Time consumption of deletions: Table 6.4 presents time consumption of three deletion strategies based on four datasets, i.e. from $SD_1(S_1)$ to $SD_4(S_2)$.

(a) For the 'naive' strategy, the deletion time is equivalent to the constructing time of the *CMK-tree* built on four datasets (including "*12 months*" transaction data).

Table 6.3: Storage space consumption for three different deletion strategies based on four datasets

datasets	'naive' strategy	space-effective strategy	time-effective strategy
$SD_1(S_1)$	the size of *CMK-tree* built on $SD_1(S_1)'$: 0.89 MB the size of *CMK-tree* built on $SD_1(S_1)$: 0.82 MB extra space required: 0.07 MB	no extra space required	extra space required: 40 KB
$SD_2(S_1)$	the size of *CMK-tree* built on $SD_2(S_1)'$: 2.4 MB the size of *CMK-tree* built on $SD_2(S_1)$: 2.2 MB extra space required: 0.2 MB	no extra space required	extra space required: 140 KB
$SD_3(S_2)$	the size of *CMK-tree* built on $SD_3(S_2)'$: 0.28 MB the size of *CMK-tree* built on $SD_3(S_2)$: 0.26 MB extra space required: 0.02 MB	no extra space required	extra space required: 10 KB
$SD_4(S_2)$	the size of *CMK-tree* built on $SD_4(S_2)'$: 2.53 MB the size of *CMK-tree* built on $SD_4(S_2)$: 2.3 MB extra space required: 0.23 MB	no extra space required	extra space required: 150 KB

Table 6.4: Time consumption of three different deletion strategies based on four datasets (I/O)

datasets	'naive' strategy	space-effective strategy	time-effective strategy
$SD_1(S_1)$	21 mins (the *CMK-tree* construction time)	4.5 secs	4.2 secs (6.7% less)
$SD_2(S_1)$	25 mins (the *CMK-tree* construction time)	15.6 secs	13.0 secs (16.7% less)
$SD_3(S_2)$	4.2 mins (the *CMK-tree* construction time)	1.5 secs	1.4 secs (6.7% less)
$SD_4(S_2)$	6.7 mins (the *CMK-tree* construction time)	19.0 secs	16.2 secs (14.7% less)

Overall, it leads to 4.2 mins–25 mins for constructing the corresponding *CMK-tree*. For the other two strategies, we test the time of deleting the index records for the transactions completed on the first day, respectively.

(b) For the space-effective strategy, the total deletion time is 1.5 secs–21.0 secs based on four datasets;

(c) For the time-effective strategy, the total deletion time is 1.4 secs–15.2 secs based on four datasets. In all cases, the time-effective strategy reduces deletion time by 6.7%–16.7% of space-effective strategy. On average, the reduction is about 11.2%.

Summary: According to the experimental results, we conclude that, compared with the other two strategies, the time-effective deletion strategy brings a small increase in the storage space with much gain in the improvement of time consumption in deletion operations.

6.4 Summary

In our previous work, several index schemes have been proposed for CTT computation, which uses additional storage space to store the precomputed aggregation results over large-scale ratings and transaction data of a seller. With significant increase of historical transaction data (e.g., one or two years), the size of that additional storage space can become much larger.

Towards solving the above problem, in this chapter, we have firstly proposed a novel model for CTT computation with fixed storage space named CTT^{FS}. The CTT^{FS} model extends the *HTA*^{FS} model [165] to reduce storage space consumption, but it is difficult to be applied in real applications due to difficulty of selecting the size of the fixed storage space. To overcome this problem, we have then proposed another approach *CMK-tree*^{RS} based on extending the *CMK-tree* to reduce the storage space allocated to each seller. The *CMK-tree*^{RS} brings a little loss to the accuracy of CTT

computation with much gain in storage space reduction and computation time improvement. Finally, apart from aggregating ratings and transaction data at different time granularity, we proposed three different deletion strategies that aim to delete index records that are generated based on ratings and transaction data from "*12 months ago*".

Based on our proposed strategies, the new context-aware transaction trust evaluation model *ReputationPro* can be more effectively applied to large-scale e-commerce websites in terms of efficiency and storage space consumption.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In recent years, e-commerce applications have emerged to change our daily life. More and more people prefer online shopping due to its convenience. However, when a buyer looks for a product out of a large pool of products provided by different sellers, in addition to functionality, the trust of a seller is also a key factor for product selection. To this end, effective trust evaluation is in great demand to provide valuable information to buyers, enabling them to select trustworthy sellers.

In this thesis, two major aspects regarding context-aware transaction trust computation in e-commerce environments have been studied.

1. One aspect of the work presented in this thesis is a trust vector based approach to context-aware transaction trust evaluation.

   (a) In contrast to most existing trust management models that compute a single trust value, a trust vector is computed for a seller. The trust vector consists of three major elements, which are called Contextual Transaction Trust (CTT) values. They are *Transaction Item Specific Trust (TIST)*, *Product Category based Trust (PCT)* and *Similar Transaction Amount based Trust (STAT)*. With our proposed trust vector, the reputation profile of a seller can be outlined to indicate his/her dynamic trustworthiness in different products, product categories, price ranges, time periods, and any necessary combination of them. Further, we term this new model as *ReputationPro*.

According to our review of the articles on context-aware trust evaluation approaches, our proposed *ReputaionPro* is a typical heuristic-based multi-context trust evaluation model which provides more detailed and comprehensive trust information of a seller.

(b) A set of methods have been proposed to calculate transaction context similarity, which can be used to infer the trustworthiness of a forthcoming transaction, in particular, when there are no or not enough ratings from the past transactions with the same context as the forthcoming transaction. Moreover, with combination of similarity calculation approaches, we have provided the solutions to the computation of that proposed trust vector. Finally, based on our experiments and analysis, the *ReputaionPro* model can identify risks potentially existing in a forthcoming transaction, thus outperforming single-value trust valuation methods and a prior trust vector based approach.

2. The other aspect of the work presented in this thesis concerns efficient computation of a seller's reputation profile.

(a) In order to clearly outline a seller's reputation profile, if a user or buyer can specify/adjust layers in product category, price range as well as transaction time range, accordingly, different ratings from different transaction contexts need to be taken into account for computation. On the other hand, in real e-commerce applications, a popular seller usually sells a wide variety of products distributed in a number of product categories. In addition, a large number of buyers can be accessing one seller's reputation data simultaneously with regard to their potential forthcoming transactions in various contexts. Taking all these factors into account, the computation of a seller's reputation profile incurs a high computational complexity. Thus, efficient algorithms are in high demand to facilitate buyers' context-aware trust enquiries on each element of the trust vector.

Towards efficient computation of CTT values, four index schemes have been proposed. In the literature, our targeted CTT computation problem is similar to the traditional RA (Range Aggregate) problem in spatial data warehouses. Therefore, at the beginning, we have extended the approaches to the two-dimensional (2D) RA problem as the preliminary solutions for CTT computation. They all meet the requirements of CTT computation, but have low efficiency in computing CTT values in some cases. Based on our analysis of the limitations of existing approaches to two-dimensional RA after being extended to solve CTT computation problem, a new disk-based index scheme and a new query algorithm are proposed. Compared with the preliminary solutions, while answering a buyer's CTT queries for each brand-based product category, the new index scheme has almost linear query performance.

(b) In addition, to solve large storage space consumption of proposed index schemes, several strategies are adopted for storage space reduction in CTT computation. These strategies include aggregating ratings and transaction data at different time granularity as well as deleting the index records that are generated based on the ratings and transaction data from remote history. Consequently, the *ReputationPro* model can be more effectively applied to large-scale e-commerce websites in terms of efficiency and storage space consumption.

7.2 Future Work

In relation to context-aware trust evaluation in e-commerce environments, some research problems remain open for which will be solved in our future work.

(a) Our proposed *ReputationPro* model is based on three identified important transaction context dimensions, *Transaction Category (Product Category)*,

Transaction Amount and *Transaction Time*. With regard to any possible new transaction context dimension, the approaches to extend the *CMK-tree* will be studied.

(b) In the literature, the design of index structures for two-dimensional RA problem is also studied based on *bit-wise* machines (e.g. the *CRB-tree* reviewed in Section 2.4.2). The *CRB-tree* has good performance in answering RA queries, and can further reduce the space consumption. Therefore, we will take the CTT computation based on *bit-wise* models as one of our future research directions, targeting more efficient solutions.

(c) One of our major concerns for CTT computation is to promptly answer buyers' CTT requests, since a large number of buyers can be accessing one seller's reputation data simultaneously with regard to their potentially forthcoming transactions in various contexts. Apart from the proposed index structures in this thesis, we consider to use the query workload patterns [36] to further improve the query performance.

(d) Effective tools are also needed to assist buyers to analyse the CTT values. The computed CTT values aiming at outlining a seller's reputation profile are comprehensive and multi-dimensional. To better use such trust information, the analysis methods and tools need to be developed. Among them, a visualisation tool is necessary to visualise the computed results. For example, the zoom in/out analysis allows users to examine the CTT values for related contexts and facilitates them comparing the trustworthiness of sellers.

Appendix A

Notations Used in This Thesis

Table A.1: Notations Used in Chapter 3

Notation	Representation	First occurrence
ta	the amount of the transaction	Section 3.3.1
ta_f	the amount of a forthcoming transaction	Section 3.2.2
ta_p	the amount of past transaction	Section 3.2.2
B	a buyer	Section 3.3.1
C-value	unique id of product category	Section 3.2.1.1
C-hrchy	the path in product category hierarchy	Section 3.3.1
CTT	Contextual Transaction Trust	Section 3.3.2
$d(p, p')$	the depth of the deepest common ancestor between two products p and p'	Section 3.2.1.2
D_{ta}	difference value of transaction amount	Section 3.2.2
p	the product purchased in the transaction	Section 3.3.1
PCT	Product Category based Trust	Section 3.3.2
r	the rating for transaction quality	Section 3.3.1
R_{ta}	relative value of transaction amount	Section 3.2.2
S	a seller	Section 3.3.1
S_{TA}	the similarity of two transaction amounts	Section 3.2.2
S_{TC}	transaction content similarity	Section 3.3.3.1
S_{TI}	the similarity of two transaction items	Section 3.2.1.2
S_{TT}	the transaction time similarity between a forthcoming transaction and a past one	Section 3.2.3
STAT	Similar Transaction Amount based Trust	Section 3.3.2
TIST	Transaction Item Specific Trust	Section 3.3.2

Table A.2: Notations Used in Chapter 3 (continued)

Notation	Representation	First occurrence
$Trans$	the set of past transactions	Section 3.3.1
TR	the transaction between a seller and a buyer at time t	Section 3.3.1
u, v	two parameters to compute ω	Eq. 3.11
λ_R	The threshold of relative value	Eq. 3.4
ω	the weight of "direct reference" ratings	Section 3.3.3.1
θ	thresholds of sufficient number of "direct reference" ratings	Section 3.3.3.1
θ_{TI}	threshold for transaction item similarity	Section 3.3.3.2
θ_{TA}	threshold for transaction amount similarity	Section 3.3.3.2

Table A.3: Notations used in Chapter 4 (continued)

Notation	Representation	First occurrence
B_1, B_2	two buyers	Section 4.3.2.2
$count\_r$	the number of ratings	Section 4.1
I_i	a record in I-node	Section 4.2.2.2
In	an I-node	Section 4.2.2.2
k	the number of brand-based product category	Section 4.2.2.4
L_i	a record in L-node	Section 4.2.2.2
Ln	an L-node	Section 4.2.2.2
RA	Range Aggregate	Section 4.1.1
R_i	a record in R-node	Section 4.2.2.1
Rn	an R-node	Section 4.2.2.1
sum_r	the sum of ratings	Section 4.1
S_1, S_2	two sellers	Section 4.3.1.1
TPT	Transaction Proportion based Trust	Section 4.2.1

Table A.4: Notations used in Chapter 6 (continued)

Notation	Representation	First occurrence
$gran$	the time granularity	Section 6.1.1
HTA	Hierarchical Temporal Aggregation	Section 6.1.1
k	the number of segments for time space	Section 6.1.1
m	the number of brand-based product categories	Section 6.1.2
S	the total storage size allocated to a seller	Section 6.1.3.1
seg	a specific time segment	Section 6.1.1
t_{begin}	the starting time	Section 6.2.1
t_{first}	the time of the first day	Section 6.3.2.2
t_{newdiv}	the time point where the ratings and transaction data are aggregated at different time granularities	Section 6.1.3.1
t_{now}	the current time	Section 6.2.1

Bibliography

[1] D. J. Abadi, S. R. Madden, and N. Hachem. Column-Stores vs. Row-Stores: How different are they really. In *ACM SIGMOD*, pages 967–980, 2008.

[2] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Hawaii International Conference on System Sciences*, pages 1–9, 2000.

[3] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *International Conference on Information and Knowledge Management*, pages 310–317, 2001.

[4] S. Adali, R. Escriva, M. Goldberg, M. Hayvanovych, M. Magdon-Ismail, B. Szymanski, W. Wallace, and G. Williams. Measuring behavioral trust in social networks. In *IEEE International Conference on Intelligence and Security Informatics*, pages 150–152, 2010.

[5] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[6] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[7] P. K. Agarwala, L. Argeb, S. Govindarajanc, J. Yanga, and K. Yi. Efficient external memory structures for range-aggregate queries. *Computational Geometry*, 46(3):358–370, 2012.

[8] G. A. Akerlof. The market for ”lemons”: Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.

[9] Alibaba. *http://resources.alibaba.com/article/232530/Protect_yourself_from_fraudsters_pretending_to_be_Gold_Suppliers.htm*.

[10] Amazon. *http://askville.amazon.com/customers-amazon-serve-year/AnswerViewer.do?requestId=88355172*.

[11] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111:1036–1060, 2004.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[13] R. Ashri, S. D. Ramchurn, J. Sabater, M. Luck, and N. R. Jennings. Trust evaluation through relationship analysis. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1005–1011, 2005.

[14] S. Ba and P. A. Pavlou. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly*, 26(3):243–268, 2002.

[15] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indices. *Acta Informatica*, 1:173–189, 1972.

[16] P. Beatty, I. Reay, S. Dick, and J. Miller. Consumer trust in e-commerce web sites: A meta-study. *ACM Computing Surveys*, 43(3):1–46, 2011.

[17] B. Becker, S. Gschwind, T. Ohler, B. Seeger, and P. Widmayer. An asymptotically optimal multiversion B-tree. *Very Large Databases*, 5(4):264–275, 1996.

[18] D. Beneventano, F. Guerra, S. Magnani, and M. Vincini. A web service based framework for the semantic mapping amongst product classification schemas. *Electronic Commerce Research*, 5(2):114–127, 2004.

[19] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. Secure internet programming. chapter The Role of Trust Management in Distributed Systems Security, pages 185–210. Springer-Verlag, 1999.

[20] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[21] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: From the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, 1997.

[22] S. Buchegger and J.-Y. L. Boudec. A robust reputation system for mobile ad-hoc networks. Technical report, P2PEcon, 2003.

[23] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the confidant protocol. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '02, pages 226–236, New York, NY, USA, 2002. ACM.

[24] A. Caballero, J. Botia, and A. Gomez-Skarmeta. On the behaviour of the trsim model for trust and reputation. In *German Conference on Multi-Agent System Technologies*, pages 13–24, 2007.

[25] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.

[26] comScore. *www.comscore.com/companyinfo*.

[27] E. Damiani, S. di Vimercati, P. Samarati, and M. Viviani. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM conference on Computer and Communications Security*, pages 207–216, 2002.

[28] E. Damiani, S. di Vimercati, P. Samarati, and M. Viviani. A wowa-based aggregation technique on trust values connected to metadata. *Electronic Notes in*

Theoretical Computer Science, 157(3):131–142, 2006.

[29] C. Dellarocas. Goodwill hunting: An economically efficient online feedback mechanism for environments with variable product quality. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 238–252, 2002.

[30] M. Deutsh. *Cooperation and Trust: Some theoretical notes*. Nebraska University Press, 1962.

[31] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[32] T. Dimitriou, G. Karame, and I. Christou. Supertrust-a secure and efficient framework for handling trust in super peer networks. In *International Conference on Distributed Computing and Networking*, pages 350–362, 2008.

[33] eBay UK. *http://pages.ebay.co.uk/aboutebay/thecompany/companyover/view.html*.

[34] E. ElSalamouny, V. Sassone, and M. Nielsen. HMM-based trust model. In *International Workshop on Formal Aspects in Security and Trust*, volume 5983, pages 21–35. Springer Berlin Heidelberg, 2010.

[35] M. Gerlach. Trust for vehicular applications. In *International Symposium on Autonomous Decentralized Systems*, pages 295–304, 2007.

[36] M. Gibas, G. Canahuate, and H. Ferhatosmanoglu. Online index recommendations for high-dimensional databases using query workloads. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):246–260, 2008.

[37] J. Golbeck and J. Hendler. Inferring binary trust relationships in web-based social networks. *ACM Transactions on Internet Technology*, 6(4):497–529, 2006.

[38] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in vanets. In *ACM International Workshop on Vehicular Ad Hoc Networks*, pages 29–37, 2004.

[39] S. Govindarajan, P. K. Agarwal, and L. Arge. CRB-Tree: An efficient indexing scheme for range-aggregate queries. In *International Conference on Database Theory*, pages 143–157.

[40] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3(4):2–16, 2000.

[41] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, and M. Venkatrao. Data Cube: A relational aggregation operator generalizing Group-by, Cross-tab, and Subtotals. In *International Conference on Data Engineering*, pages 152–159, 1996.

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

[43] N. Griffiths. Trust: Challenges and opportunities. *AgentLink News*, 19:9–11, 2005.

[44] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *International Conference on World Wide Web*, pages 403–412, 2004.

[45] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux. Design and analysis of a lightweight certificate revocation mechanism for vanet. In *ACM International Workshop on VehiculAr InterNETworking*, pages 89–98, 2009.

[46] S. M. Habib, S. Ries, and M. Muhlhauser. Towards a trust management system for cloud computing. In *International Conference on Trust, Security and Privacy in Computing and Communications*, pages 933–939, 2011.

[47] F. Ham, E. Imana, A. Ondi, R. Ford, W. Allen, and M. Reedy. Reputation prediction in mobile ad hoc networks using RBF neural networks. In *International Conference on Engineering Applications of Neural Networks*, volume 43, pages 485–494, 2009.

[48] C.-W. Hang, Y. Wang, and M. P. Singh. Operators for propagating trust and their evaluation in social networks. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1025–1032, 2009.

[49] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *ACM SIGMOD*, pages 205–216, 1996.

[50] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 42(1):1–31, 2009.

[51] S. Holtmanns and Z. Yan. Context-aware adaptive trust. In *Developing Ambient Intelligence*, pages 137–146, 2006.

[52] K. Hwang and D. Li. Trusted cloud computing with secure resources and data coloring. *IEEE Internet Computing*, 14(5):14–22, 2010.

[53] R. Z. Iris Bohnet. Trust, risk and betrayal. *Economic Behavior & Organization*, 55:467–484, 2004.

[54] S. Jones. TRUST-EC*: requirements for trust and confidence in e-commerce*. European Commission, Joint Research Center, 1999.

[55] A. Jøsang. Artificial reasoning with subjective logic. *Australian Workshop on Commonsense Reasoning*, 1997.

[56] A. Jøsang. A logic for uncertain probabilities. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–212, 2001.

[57] A. Jøsang. Robustness of trust and reputation systems: Does it matter? In *IFIP International Conference on Trust Management*, pages 253–262, 2012.

[58] A. Jøsang and J. Golbeck. Challenges for robust of trust and reputation systems. In *International Workshop on Security and Trust Management*, 2009.

[59] A. Jøsang, E. Gray, and M. Kinateder. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.

[60] A. Jøsang and R. Ismail. The beta reputation system. In *Bled Electronic Commerce Conference*, 2002.

[61] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[62] A. Jøsang and S. Lo Presti. Analysing the relationship between risk and trust. In *IEEE Conference on Trust Management*, pages 135–145, 2004.

[63] M. Jurgens and H.-J. Lenz. The Ra*-tree: An improved R-tree with materialized data for supporting range queries on OLAP-data. In *International Workshop on Database and Expert Systems Applications*, pages 186–191, 1998.

[64] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *International World Wide Web Conference*, pages 640–651, 2003.

[65] S. T. Kang, Y. D. Chung, and M. H. Kim. An efficient method for temporal aggregation with range-condition attributes. *Information Sciences*, 168(1-4):243–265, 2004.

[66] R. Kerr and R. Cohen. Modelling trust using transactional, numerical units. In *ACM International Conference on Privacy, Security and Trust*, pages 21:1–21:11, 2006.

[67] R. Kerr and R. Cohen. Smart cheaters do prosper: Defeating trust and reputation systems. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 993–1000, 2009.

[68] D. J. Kim, D. L. Ferrin, and H. R. Rao. A trust-based consumer decision-making model in electronic commerce: The role of trust, perceived risk, and

their antecedents. *Decision Support Systems*, 44(2):342–353, 2008.

[69] R. Kingston. The social implications of e-commerce: a review of policy and research. Technical report, University of Leeds, 2001.

[70] N. Kline and R. Snodgrass. Computing temporal aggregates. In *International Conference on Data Engineering*, pages 222–231, 1995.

[71] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*. RFC Editor, 1993.

[72] K. Konrad, G. Fuchs, and J. Barthel. Trust and Electronic Commerce-more than a technical problem. In *IEEE Symposium on Reliable Distributed Systems*, pages 360–365, 1999.

[73] J. S. Lener and D. Keltner. Fear, anger, and risk. *Personality and Social Psychology*, 81(1):146–159, 2001.

[74] L. Li and Y. Wang. Context based trust normalization in service-oriented environments. In *IEEE International Conference on Autonomic and Trusted Computing*, pages 122–138, 2010.

[75] L. Li and Y. Wang. Subjective trust inference in composite services. In *AAAI Conference on Artificial Intelligence*, pages 1377–1384, 2010.

[76] L. Li and Y. Wang. The study of trust vector based trust rating aggregation in service-oriented environments. *World Wide Web*, 15(5-6):547–579, 2012.

[77] W.-Y. Lin and I.-C. Kuo. A genetic selection algorithm for OLAP data cubes. *Knowledge and Information Systems*, 6(1):83–102, 2004.

[78] G. Liu, Y. Wang, and M. Orgun. Trust inference in complex trust-oriented social networks. In *International Conference on Computational Science and Engineering*, pages 996–1001, 2009.

[79] G. Liu, Y. Wang, and M. A. Orgun. Optimal social trust path selection in complex social networks. In *AAAI Conference on Artificial Intelligence*, pages 1391–1398, 2010.

[80] G. Liu, Y. Wang, M. A. Orgun, and H. Liu. Discovering trust networks for the selection of trustworthy service providers in complex contextual social networks. In *IEEE International Conference on Web Services*, pages 384–391, 2012.

[81] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile Ad Hoc networks. In *International Conference on Trust Management*, pages 48–62, 2004.

[82] X. Liu and A. Datta. Contextual trust aided enhancement of data availability in peer-to-peer backup storage systems. *Network and Systems Management*, 20(2):200–225, 2012.

[83] X. Liu and A. Datta. Modeling context aware dynamic trust using Hidden Markov Model. In *AAAI Conference on Artificial Intelligence*, pages 1938–1944, 2012.

[84] X. Liu, A. Datta, H. Fang, and J. Zhang. Detecting imprudence of 'reliable' sellers in online auction sites. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 246–253, 2012.

[85] Y. Liu, A. H. Ngu, and L. Z. Zeng. Qos computation and policing in dynamic web service selection. In *International World Wide Web Conference*, pages 66–73, 2004.

[86] N. Luhmann. *Trust and Power*. John Wiley & Sons, Inc, 1979.

[87] H. Ma, T. C. Zhou, M. R. Lyu, and I. King. Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems*, 29(2):9:1–9:23, 2011.

[88] Z. Malik and A. Bouguettaya. RATEWeb: Reputation assessment for trust establishment among web services. *Very Large Databases*, 18(4):885–911, 2009.

[89] Z. Malik and A. Bouguettaya. *Trust Management for Service-Oriented Environments*. Springer-Verlag, 2009.

[90] D. W. Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *IEEE International Conference on Distributed Computing Systems*, pages 312–321, 1998.

[91] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[92] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[93] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Record*, 31(4):36–41, 2002.

[94] D. H. Mcknight and N. L. Chervany. The meanings of trust. Technical report, University of Minnesota, 1996.

[95] W. W. Moe and P. S. Fader. Dynamic conversion behavior at e-commerce sites. *Management Science*, 50(3):326–335, 2004.

[96] B. Moon, I. F. V. Lopez, and V. Immanuel. Efficient algorithms for large-scale temporal aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):744–759, 2003.

[97] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[98] L. Mui, M. Mohtashemi, and C. Ang. A probabilistic rating framework for pervasive computing environments. In *MIT Student Oxygen Workshop*, 2001.

[99] J. J. Murphy. *Technical Analysis of the Financial Markets*. Prentice Hall Press, 1999.

[100] T. H. Noor, Q. Z. Sheng, S. Zeadally, and J. Yu. Trust management of services in cloud environments: Obstacles and solutions. *ACM Computing Surveys*, 46(1):1–30, 2013.

[101] M. H. Overmars. The design of dynamic data structures. *Lecture Notes in Computer Science*, 156, 1983.

[102] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[103] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1535–1549, 2008.

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

[105] D. Papadias, Y. F. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *International Conference on Data Engineering*, pages 166–175, 2002.

[106] A. S. Patrick. Building trustworthy software agents. *IEEE Internet Computing*, 6(6):46–53, 2002.

[107] I. Ray and S. Chakraborty. A vector model of trust for developing trustworthy systems. In *European Symposium on Research Computer Security*, pages 260–275, 2004.

[108] M. Rehak, M. Pechoucek, and J. M. Bradshaw. Representing context for multiagent trust modeling. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 737–746, 2006.

[109] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. *Advances in Applied Microeconomics*, 11:127–157, 2002.

[110] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on eBay: A controlled experiment. *Experimental Economics*, 9:79–101, 2003.

[111] A. Rettinger, M. Nickles, and V. Tresp. Statistical relational learning of trust. *Machine Learning*, 82(2):191–209, 2011.

[112] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *International Semantic Web Conference*, pages 351–368, 2003.

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

[114] J. Robinson. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In *ACM SIGMOD*, pages 10–18, 1981.

[115] R. J. Robles and T. hoon Kim. Review: Context aware tools for smart home development. *International Journal of Smart Home*, 4(1):187–200, 2010.

[116] J. B. Rotter. A new scale for the measurement of interpersonal trust. *Personality*, 35(4):651–665, 1967.

[117] D. M. Rousseau, S. B. Sitkin, R. S. Burt, and C. Camerer. Not so different after all: A cross-discipline view of trust. *Academy of Management Review*, 23(3):393–404, 1998.

[118] S. Ruohomaa and L. Kutvonen. Trust management survey. In *International Conference on Trust Management*, pages 77–92, 2005.

[119] S. Ruohomaa, L. Kutvonen, and E. Koutrouli. Reputation management survey. In *International Conference on Availability, Reliability and Security*, pages 103–111, 2007.

[120] J. Sabater and C. Sierra. REGRET: reputation in gregarious societies. In *ACM AGENTS*, pages 194–195, 2001.

[121] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.

[122] J. Samek and F. Zboril. Hierarchical model of trust in contexts. In *Networked Digital Technologies. Communications in Computer and Information Science*, pages 356–365, 2010.

[123] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.

[124] W. Sherchan, S. Nepal, and C. Paris. A survey of trust in social networks. *ACM Computing Surveys*, 45(4):1–33, 2013.

[125] A. Sieg, B. Mobasher, and R. Burke. Ontological user profiles for representing context in web search. In *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, pages 91–94, 2007.

[126] C. Sierra and J. Debenham. Information-based agency. In *International Joint Conference on Artificial Intelligence*, pages 1513–1518, 2007.

[127] C. Sierra and J. Debenham. The logic negotiation model. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1026–1033, 2007.

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

[129] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, 2001.

[130] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *International Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.

[131] G. Suryanarayana and R. N. Taylor. A survey of trust management and resource discovery technologies in peer-to-peer applications. Technical report, University of California, Irvine, 2004.

[132] A. Swaminathan, R. G. Cattelan, Y. Wexler, C. V. Mathew, and D. Kirovski. Relating reputation and money in online markets. *ACM Transactions on the Web*, 4(4):1–31, 2010.

[133] P. Sztompka. *Trust A Sociological Theory*. Cambridge University Press, 1999.

[134] Y. Tao and D. Papadias. Historical spatio-temporal aggregation. *ACM Transactions on Information Systems*, 23(1):61–102, 2005.

[135] 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.

[136] M. Tavakolifard, S. J. Knapskog, and P. Herrmann. Trust transferability among similar contexts. In *ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pages 91–97, 2008.

[137] R. K. Taylor. Marketing strategies: Gaining a competitive advantage through the use of emotion. *Competitiveness Review*, 10(2):146–152, 2000.

[138] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. TRAVOS: trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

[139] R. Tian and X. Lan. E-commerce concerns: Cross-cultural factors in internet marketing. In *International Conference on Electronic Commerce and Business Intelligence*, pages 83–86, 2009.

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

[141] T. Tran and R. Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modelling and adjustable product quality. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 828–835, 2004.

[142] 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, 2008.

[143] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11:93–136, 1996.

[144] S. d. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati. Integrating trust management and access control in data-intensive web applications. *ACM Transactions on the Web*, 6(2):6:1–6:43, 2012.

[145] L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-based service selection and ranking with trust and reputation management. In *International Conference on Cooperative information system*, pages 466–483, 2005.

[146] X. F. Wang, L. Liu, and J. Su. RLM: a general model for trust representation and aggregation. *IEEE Transaction on Service Computing*, 5(1):131–143, 2012.

[147] Y. Wang and L. Li. Two-dimensional trust rating aggregations in service-oriented applications. *IEEE Transactions on Service Computing*, 4(4):257–271, 2011.

[148] Y. Wang, L. Li, and G. Liu. Social context-aware trust inference for trust enhancement in social network based recommendations on service providers. *World Wide Web*, pages 1–26, 2013.

[149] Y. Wang and E.-P. Lim. The evaluation of situational transaction trust in e-service environments. In *IEEE International Conference of Engineering and Business Education*, pages 265–272, 2008.

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

[151] Y. Wang, K.-J. Lin, D. S. Wong, and V. Varadharajan. Trust management towards service-oriented applications. *Service Oriented Computing and Applications*, 3(2):129–146, 2009.

[152] Y. Wang and M. P. Singh. Formal trust model for multiagent systems. In *International Joint Conference on Artificial Intelligence*, pages 1551–1556, 2007.

[153] Y. Wang and V. Varadharajan. Trust^2: Developing trust in peer-to-peer environments. In *International Conference on Services Computing*, pages 24–31, 2005.

[154] Y. Wang and V. Varadharajan. Two-phase peer evaluation in p2p e-commerce environments. In *International Conference on e-Technology, e-Commerce and e-Service*, pages 654–657, 2005.

[155] Y. Wang and J. Vassileva. Toward trust and reputation based web service selection: A survey. Technical report, University of Saskatchewan, 2007.

[156] Y. Wang, D. S. Wong, K.-J. Lin, and V. Varadharajan. Evaluating transaction trust and risk levels in peer-to-peer e-commerce environments. *Information Systems and e-Business Management*, 6(1):25–48, 2008.

[157] Y. Wang, D. S. Wong, K. J. Lin, and V. Varadharajan. Evaluating transaction trust and risk levels in peer-to-peer e-commerce environments. *Information Systems and e-Business Management*, 6(1):25–48, 2008.

[158] 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, 2003.

[159] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust in peer-to-peer communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[160] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. In *Nordic Workshop on Secure IT Systems*, pages 933–939, 2003.

[161] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. *Very Large Databases*, 12(3):262–283, 2003.

[162] B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In *International Workshop on Cooperative Information Agents IV*, pages 154–165, 2000.

[163] B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *ACM International Conference on Autonomous Agents and Multiagent Systems*, pages 294–301, 2002.

[164] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907, 2000.

[165] D. Zhang, D. Gunopulos, V. J. Tsotras, and B. Seeger. Temporal and spatio-temporal aggregations over data streams using multiple time granularities. *Information System*, 28(1-2):61–84, 2003.

[166] D. Zhang, A. Markowetz, V. J. Tsotras, D. Gunopulos, and B. Seeger. Efficient computation of temporal aggregates with range predicates. In *ACM International Symposium on Principles of Database Systems*, pages 237–245, 2001.

[167] D. Zhang, A. Markowetz, V. J. Tsotras, D. Gunopulos, and B. Seeger. Efficient aggregation over objects with extent. In *ACM International Symposium on Principles of Database Systems*, pages 121–132, 2002.

[168] D. Zhang, A. Markowetz, V. J. Tsotras, D. Gunopulos, and B. Seeger. On computing temporal aggregates with range predicates. *ACM Transactions on Database Systems*, 33(2):1–38, 2008.

[169] H. Zhang and Y. Wang. A novel model for contextual transaction trust computation with fixed storage space in e-commerce and e-service environments. In *IEEE International Conference on Services Computing*, pages 667–674, 2013.

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

[171] H. Zhang, Y. Wang, and X. Zhang. Efficient contextual transaction trust computation in e-commerce environments. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 318–325, 2012.

[172] H. Zhang, Y. Wang, and X. Zhang. A trust vector approach to transaction context-aware trust evaluation in e-commerce and e-service environments. In

IEEE International Conference on Service Oriented Computing and Applications, pages 1–8, 2012.

[173] H. Zhang, Y. Wang, and X. Zhang. The approaches to contextual transaction trust computation in e-commerce environments. *Security and Communication Networks*, doi:10.1002/sec.839, Preprint, 2014.

[174] H. Zhang, Y. Wang, X. Zhang, and E.-P. Lim. ReputationPro: The efficient approaches to contextual transaction trust computation in e-commerce environments. *ACM Transactions on the Web*, Submitted, 2014.

[175] H. Zhang and H. Zhang. A security enhancement scheme for image perceptual hashing. In *International Joint Conference on INC, IMS and IDC*, pages 1445–1448, 2009.

[176] H. Zhang, H. Zhang, Q. Li, and X. Niu. Predigest watson's visual model as perceptual hashing method. In *International Conference on Convergence and Hybrid Information Technology*, pages 617–620, 2008.

[177] J. Zhang. A survey on trust management for vanets. In *IEEE International Conference on Advanced Information Networking and Applications*, pages 105–112, 2011.

[178] H. Zhao and X. Li. Vectortrust: Trust vector aggregation scheme for trust management in peer-to-peer networks. In *IEEE International Conference on Computer Communications and Networks*, pages 1–6, 2009.

[179] X. Zheng, Y. Wang, and M. A. Orgun. Modeling the dynamic trust of online service providers using HMM. In *International Conference on Web Services*, pages 459–466, 2013.

[180] R. Zhou and K. Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[181] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. Robust cooperative trust establishment for manets. In *ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 23–34, 2006.