ISSN (Online) : 2278-1021
ISSN (Print)   : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 8, October 2012

Efficient Handling of XML Tree Pattern Matching Queries – A Holistic Approach

Sravan Kumar K[1], Madhu P[2], Raghava Rao N[3]

M.Tech-SE, DRK Institute of Science and Technology, JNTUH, Hyderabad, India[1]

M.Tech-CS, DRK College of Engg. and Technology, JNTUH, Hyderabad, India[2]

HOD, Dept. of CSE, DRK Institute of Science and Technology, JNTUH, Hyderabad, India[3]

ABSTRACT:XML has become a defacto standard to store, share and exchange business data across homogenous and heterogeneous platforms. The interoperability is possible though XML. As enterprises are generating huge amount of data in XML format, there is a need for processing XML tree pattern queries. The existing holistic algorithms for XML tree pattern matching queries exhibit suboptimmality problem as they consider intermediate results before taking final results. This causes suboptimal performance. This suboptimality is overcome by using TreeMatch algorithm. This paper implements a prototype application that makes use of dewey labeling scheme to overcome suboptimality. The experimental results revealed that the proposed algorithm is better than the existing algorithms.

Index Terms –XML, parser, XSL, algorithms, dewey labeling, xml tree pattern query.

I. INTRODUCTION

Due to the business collaborations and for the purpose of portability enterprises are storing data in XML format. This has become a common practice as XML is portable and irrespective of platforms in which applications were developed, they can share information through XML file format. Such XML files are also validated using DTD or Schema. XML parsers are available in all languages that facilitate the usage of XML programmatically. Moreover XML is tree based and it is convenient to manipulate easily using DOM (Document Object Model) API. XML tree pattern queries are to be processed efficiently as that is the core operation of XML data. Recently many researchers developed various methods or algorithms [2], [3], [4], [5], [6], [7] for processing XML tree queries. A stack based algorithm [1] was proposed by Khalifa et al. that matches relationships such as A-D, and P-C. TwigStack is another algorithm proposed by Bruno et al. [8] for the purpose of XML tree pattern queries. However, the drawback of these algorithms is that they take unnecessary intermediary nodes while processing the query thus causing more time to process. To overcome the drawbacks indexing concept is used by algorithms provided in [9] and [10]. Some other algorithms use labeling schemes [11]. In industrial and academic applications these algorithms have proven to be highly promising [12]. However, from the study ofliterature we observed the fact that there is less research with respect to XML tree pattern queries with wildcards, order restrictions, negation and functions.

To address all these problems, we implement a TreeMatch algorithm that avoids suboptimality of those algorithms. This algorithm is based on the extended dewey labeling. As per the labeling scheme, the root node, children, grand children etc. a number or label is associated. For instance 0 is assigned to root node. The children of root gets labeling such as 0.0, 0.1 etc. The grand children of first parent node start with 0.0.0 and continue like 0.0.1 etc.
The rest of the paper is organized as follows:
Section II reviews the literature that gives insights into the research topic. Section III explains the system modeling. Section IV shows proposed technique. Section V provides the experimental results while section VI concludes the paper which is followed by the references.

II. RELATED WORK

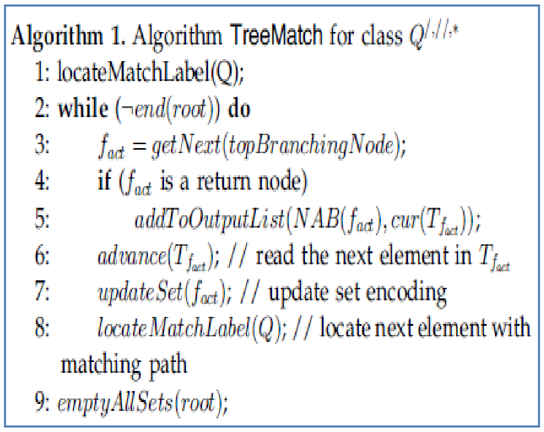
As XML databases are growing in quantity and usage by enterprises, it is essential to have an efficient mechanism to answer such queries. In literature many researchers have proposed algorithms for XML tree pattern query processing. Many have handled such queries efficiently. However, most of them did not focus on wildcards, functions, order criteria and negation. In this paper we overcome this entire problem by implementing a

TreeMatch algorithm [1]. XML query processing was considered by Timber [13] and Lore DBMS [9]. More focused research on XML storage, and relational databases were done in [14], [15], [16], [17]. These techniques can be leveraged by our holistic algorithm TreeMatch. Choi et al. [18] developed many theorems that are used to prove that the holistic algorithms for XML tree pattern matching can't solve the problem of suboptimality. XML twig queries and their space complexity has been studied in [19] by Shalem and BarYossef. They considered upper bound in parent – child and ancestor – descendent relationships in query processing. Many algorithms developed in existing works have focused on XML tree pattern matching queries in terms of P-C or A-D relationships or both. All these holistic algorithms have drawbacks in query processing. Their computational cost is more as they take some intermediary results in to consideration which is not necessary. That problem is considered matching cross and those results in suboptimiality of performance. Apart from holistic algorithms for XML tree pattern matching queries; there are many other approaches available. For instance ViST [20] and PRIX [21] perform processing by transforming XML tree pattern match into a sequence match. Those algorithms gave importance to only ordered queries and extending them to support unordered queries is non-trivial. In [14] a comprehensive research and experiments were made in order to compare various algorithms used to match XML tree pattern matching queries. Those algorithms are holistic and confirm that they are best available algorithms for XML tree pattern matching queries. They provide guarantee in performance and robust in nature. Nevertheless, they have drawback of using some unnecessary intermediary results that they can't prevent. The proposed algorithm TreeMatch thing in the lines of holistic and overcomes the problem of suboptimality in XML tree pattern matching queries. Moreover it supports 3 types of classes as described in [1] and the prototype application implemented demonstrates all three kinds of queries with negative predicates, wild cards, ordered and unordered restrictions. The tree match algorithm is based on the extended dewey labeling scheme.

III. EXTENDED DEWEY LABELING

It is an improved labeling scheme that considers the root label as zero. The children of root are given 0.1, 0.2, etc. The grand children of root are given as 0.1.0, 0.1.1, etc. This labeling scheme can describe the tree easily. Once a node is found, its ancestors and descendents and other relationships can be found easily. For instance, 0.0.1.5 indicates that 0.0.1 is the parent of current node and grand parent is 0.0 and the root is 0. This labeling makes XML tree pattern matching query processing easy.

IV. PROPOSED ALGORITHM



```
Algorithm 1. Algorithm TreeMatch for class $Q^{/,//,*}$
  1: locateMatchLabel(Q);
  2: while ($\neg end(root)$) do
  3:       $f_{act} = getNext(topBranchingNode)$;
  4:       if ($f_{act}$ is a return node)
  5:             $addToOutputList(NAB(f_{act}), cur(T_{f_{act}}))$;
  6:       $advance(T_{f_{act}})$; // read the next element in $T_{f_{act}}$
  7:       $updateSet(f_{act})$; // update set encoding
  8:       $locateMatchLabel(Q)$; // locate next element with
     matching path
  9: $emptyAllSets(root)$;
```

Fig. 1 – The main TreeMatch Algorithm [1]

This algorithm invokes functions provided in [1] to complete query processing. It makes use of locating matching extended dewey label for the given query and then completes processing. As discussed in the introduction, this algorithm is an improved holistic algorithm that makes use of labeling scheme and avoids taking useless intermediary results. Thus its processing time is less and solves the problem of suboptimality.

V. EXPERIMENTS and RESULTS

A. Environment

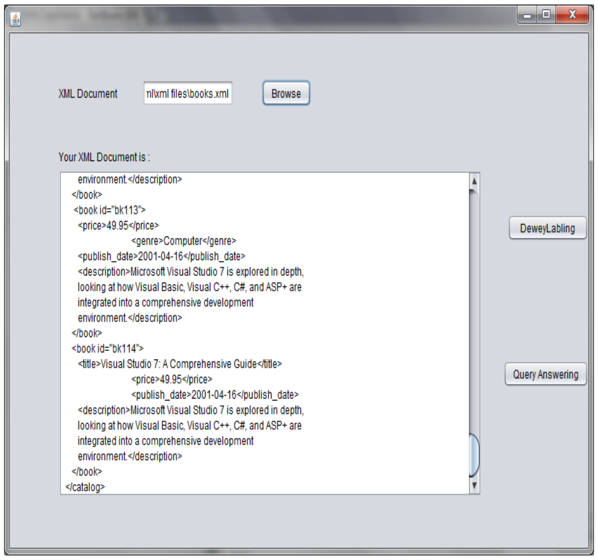
The environment used for the experiments include Java Programming Language (JSE 6.0), Net Beans IDE, a PC with 2GB RAM. The SWING API of Java is used to build graphical user interface while the IO and XML API of Java are used for actual functionality.

B. Prototype Application

For extended XML tree pattern matching while making queries on XML database, the main screen of the application appears as shown in fig. 6. The application has three buttons. The browse button enables user to choose any XML file on which queries are to be made. The XML document tree is shown in the text area. The DeweyLabling button generates XML tree with Dewey Labeling scheme implemented.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

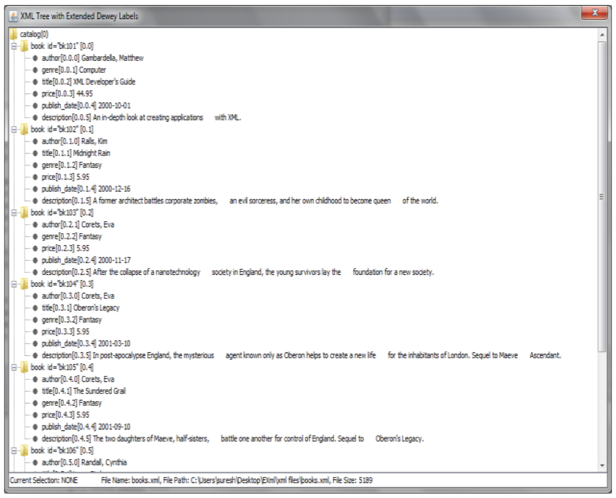
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 8, October 2012



```
    environment.</description>
  </book>
  <book id="bk113">
    <price>49.95</price>
                    <genre>Computer</genre>
    <publish_date>2001-04-16</publish_date>
    <description>Microsoft Visual Studio 7 is explored in depth,
    looking at how Visual Basic, Visual C++, C#, and ASP+ are
    integrated into a comprehensive development
    environment.</description>
  </book>
  <book id="bk114">
    <title>Visual Studio 7: A Comprehensive Guide</title>
                    <price>49.95</price>
                    <publish_date>2001-04-16</publish_date>
    <description>Microsoft Visual Studio 7 is explored in depth,
    looking at how Visual Basic, Visual C++, C#, and ASP+ are
    integrated into a comprehensive development
    environment.</description>
  </book>
</catalog>
```

Fig. 6 – The main GUI of the application

On choosing QueryAnswering button, the application takes user to a form where queries can be made. The application supports all kinds of XPATH queries. However, the queries are processed as per the algorithms presented in [1]. Especially dewey labeling made the query processing easy as it is simple to determine ancestor and descendent relationships with dewey labeling. In Fig. 6 books details are in XML format. The root element is catalog and it contains a collection of books. All queries later are made on this XML only. However, the application has been tested with a variety of XML files containing variations such as empty elements, elements only elements, mixed elements, elements with body etc. The tested XML files have both complex and simple elements. Dewey labeling for the selected XML file is visualized as shown in fig. 7.



```
catalog(0)
book id="bk101" [0.0]
    author[0.0.0] Gambardella, Matthew
    genre[0.0.1] Computer
    title[0.0.2] XML Developer's Guide
    price[0.0.3] 44.95
    publish_date[0.0.4] 2000-10-01
    description[0.0.5] An in-depth look at creating applications      with XML.
book id="bk102" [0.1]
    author[0.1.0] Ralls, Kim
    title[0.1.1] Midnight Rain
    genre[0.1.2] Fantasy
    price[0.1.3] 5.95
    publish_date[0.1.4] 2000-12-16
    description[0.1.5] A former architect battles corporate zombies,      an evil sorceress, and her own childhood to become queen      of the world.
book id="bk103" [0.2]
    author[0.2.1] Corets, Eva
    genre[0.2.2] Fantasy
    price[0.2.3] 5.95
    publish_date[0.2.4] 2000-11-17
    description[0.2.5] After the collapse of a nanotechnology      society in England, the young survivors lay the      foundation for a new society.
book id="bk104" [0.3]
    author[0.3.0] Corets, Eva
    title[0.3.1] Oberon's Legacy
    genre[0.3.2] Fantasy
    price[0.3.3] 5.95
    publish_date[0.3.4] 2001-03-10
    description[0.3.5] In post-apocalypse England, the mysterious      agent known only as Oberon helps to create a new life      for the inhabitants of London. Sequel to Maeve      Ascendant.
book id="bk105" [0.4]
    author[0.4.0] Corets, Eva
    title[0.4.1] The Sundered Grail
    genre[0.4.2] Fantasy
    price[0.4.3] 5.95
    publish_date[0.4.4] 2001-09-10
    description[0.4.5] The two daughters of Maeve, half-sisters,      battle one another for control of England. Sequel to      Oberon's Legacy.
book id="bk106" [0.5]
    author[0.5.0] Randall, Cynthia
Current Selection: NONE        File Name: books.xml, File Path: C:\Users\suresh\Desktop\EXml\xml files\books.xml, File Size: 5189
```

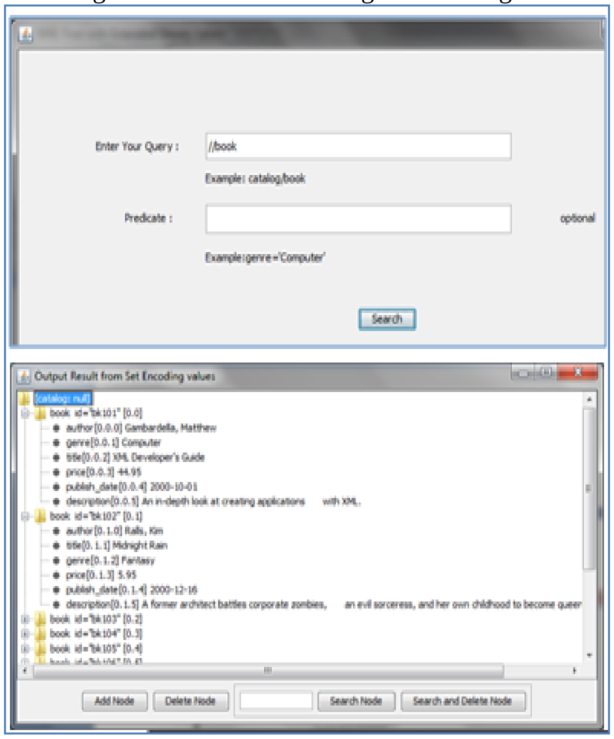
Fig. 7 – XML tree with Dewey Labeling

As can be seen in fig. 7, dewey labeling starts with root element with label zero. All children of root get labeling like 0.1, 0.2 and so on. The root element is "catalog" which has label 0. The first child "book" element has label 0.1. In turn the children of first book element have the labels like 0.1.0, 0.1.1, 0.1.2 and so on. This way labeling is applied to the entire tree. When label of any element dewey is known, it is possible to find its siblings, ancestor and descendents easily. Wild cards are also supported in the query processing of the application.

C. Tree Pattern Matching Queries

On clicking "Query Answering" button on the main screen of the application user is given a query window where tree pattern matching queries can be given. The proposed application supports the 3 types of queries as described in [1]. Fig. 8 shows both query and the corresponding answer.

The query here is //book. It does mean that all book elements irrespective of their position in the XML root element are to be presented. This is evident in the results. The results window also has provisions to further manipulate the XML tree visualized. It supports operations like adding new node, deleting node, searching for a node and searching and deleting a node.

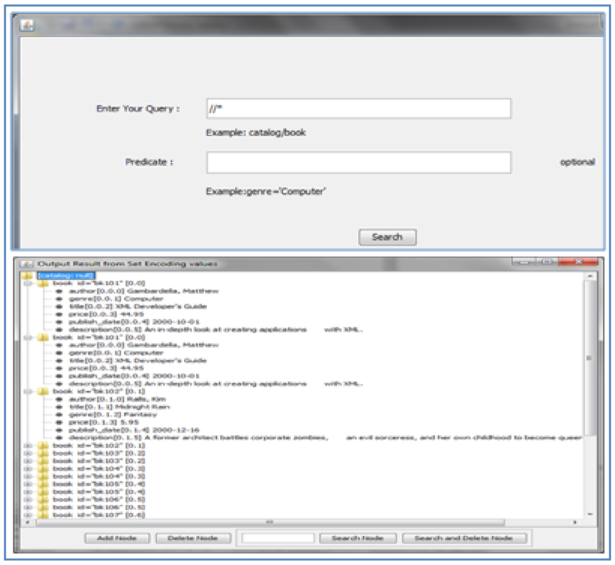


```
[catalog: null]
book id="bk101" [0.0]
    author[0.0.0] Gambardella, Matthew
    genre[0.0.1] Computer
    title[0.0.2] XML Developer's Guide
    price[0.0.3] 44.95
    publish_date[0.0.4] 2000-10-01
    description[0.0.5] An in-depth look at creating applications      with XML.
book id="bk102" [0.1]
    author[0.1.0] Ralls, Kim
    title[0.1.1] Midnight Rain
    genre[0.1.2] Fantasy
    price[0.1.3] 5.95
    publish_date[0.1.4] 2000-12-16
    description[0.1.5] A former architect battles corporate zombies,      an evil sorceress, and her own childhood to become queen
book id="bk103" [0.2]
book id="bk104" [0.3]
book id="bk105" [0.4]
```

Fig. 8 – Query and Results without wildcards

Fig. 9 shown query with wild card characters and results.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

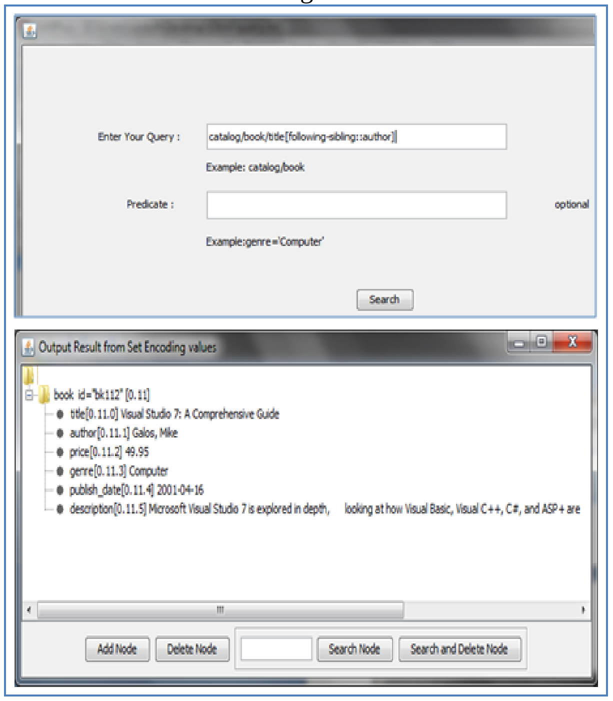


Enter Your Query : //*
Example: catalog/book
Predicate : optional
Example:genre='Computer'

```
Output Result from Set Encoding values
[catalog: null]
book id="bk101" [0.0]
    author[0.0.0] Gambardella, Matthew
    genre[0.0.1] Computer
    title[0.0.2] XML Developer's Guide
    price[0.0.3] 44.95
    publish_date[0.0.4] 2000-10-01
    description[0.0.5] An in-depth look at creating applications      with XML.
book id="bk101" [0.0]
    author[0.0.0] Gambardella, Matthew
    genre[0.0.1] Computer
    title[0.0.2] XML Developer's Guide
    price[0.0.3] 44.95
    publish_date[0.0.4] 2000-10-01
    description[0.0.5] An in-depth look at creating applications      with XML.
book id="bk102" [0.1]
    author[0.1.0] Ralls, Kim
    title[0.1.1] Midnight Rain
    genre[0.1.2] Fantasy
    price[0.1.3] 5.95
    publish_date[0.1.4] 2000-12-16
    description[0.1.5] A former architect battles corporate zombies,      an evil sorceress, and her own childhood to become queen
book id="bk102" [0.1]
book id="bk103" [0.2]
book id="bk103" [0.2]
book id="bk104" [0.3]
book id="bk104" [0.3]
book id="bk105" [0.4]
book id="bk105" [0.4]
book id="bk106" [0.5]
book id="bk106" [0.5]
book id="bk107" [0.6]
```

Fig. 9 – Query with wildcards and results

As can be seen in fig. 9, the query is given as "//*". This does mean that any element present anywhere in the document. The results reveal the entire XML file. The query results are presented in fig. 9 in the form of a tree. JTree class of Java SWING API is used to visualize XML tree. This tree is flexible and can be navigated and manipulated easily. Fig. 10 shows a query with restriction in terms of siblings.



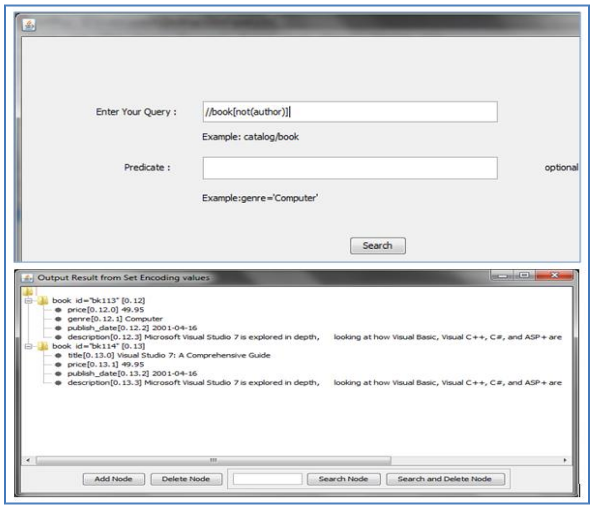
Enter Your Query : catalog/book/title[following-sibling::author]
Example: catalog/book
Predicate : optional
Example:genre='Computer'

```
Output Result from Set Encoding values
book id="bk112" [0.11]
    title[0.11.0] Visual Studio 7: A Comprehensive Guide
    author[0.11.1] Galos, Mike
    price[0.11.2] 49.95
    genre[0.11.3] Computer
    publish_date[0.11.4] 2001-04-16
    description[0.11.5] Microsoft Visual Studio 7 is explored in depth,      looking at how Visual Basic, Visual C++, C#, and ASP+ are
```

Fig. 10 – Query with Sibling and Results

As can be seen in fig. 10 the query given is "catalog/book/title[following-sibling::author]". It does mean that the root element must be catalog and its sub element book. The book's sub element must be title and the title must have a sibling followed by it. The results that match these criteria are shown in fig. 10. Fig. 11 shows query with negation.

As can be seen in fig. 11, the given query is "//book[not(author)]. It does mean that "book" element can be anywhere in the XML file without having "author" element in it. The query results revealed that two books are available without author.



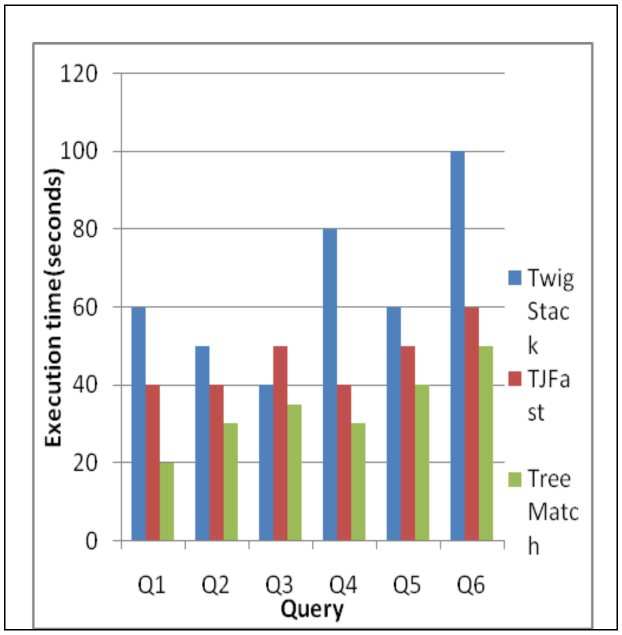
Enter Your Query : //book[not(author)]
Example: catalog/book
Predicate : optional
Example:genre='Computer'

```
Output Result from Set Encoding values
book id="bk113" [0.12]
    price[0.12.0] 49.95
    genre[0.12.1] Computer
    publish_date[0.12.2] 2001-04-16
    description[0.12.3] Microsoft Visual Studio 7 is explored in depth,      looking at how Visual Basic, Visual C++, C#, and ASP+ are
book id="bk114" [0.13]
    title[0.13.0] Visual Studio 7: A Comprehensive Guide
    price[0.13.1] 49.95
    publish_date[0.13.2] 2001-04-16
    description[0.13.3] Microsoft Visual Studio 7 is explored in depth,      looking at how Visual Basic, Visual C++, C#, and ASP+ are
```

Fig. 11 – Query with Negation and Results

D. Results

The experimental results of TreeMatch algorithm is compared with other holistic algorithms such as TwigStack, and TJFast.



ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 8, October 2012

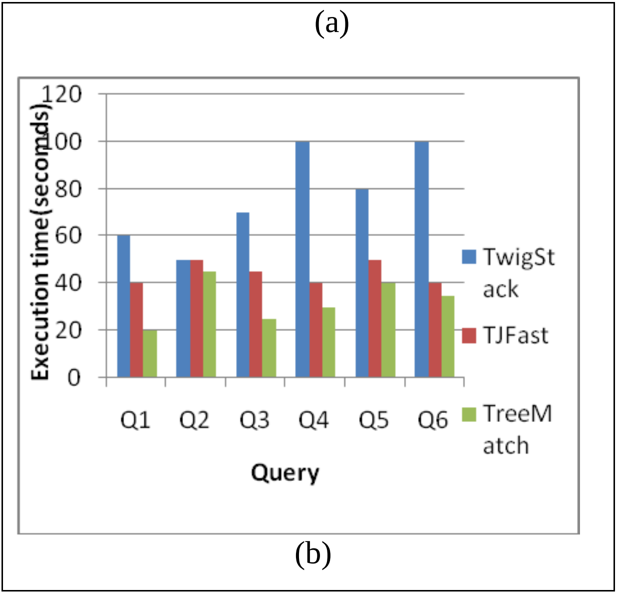


Fig. 12 – Execution time on random data (a) Small memory (b) Large memory

As seen in fig. 12, with respect to small and large memory with random data TreeMatch performance is always best.

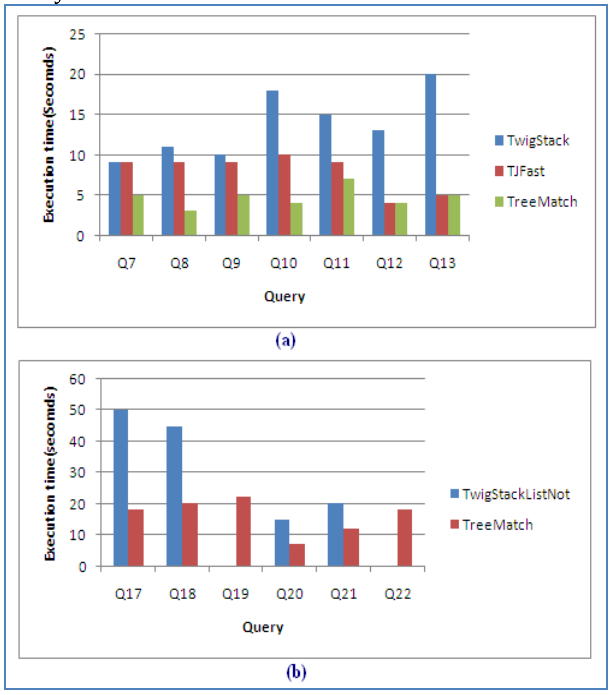


Fig. 13 – Execution time on TreeBank data(large memory).

As seen in fig. 12, with respect to large memory with TreeBank data TreeMatch performance is always best.

VI. CONCLUSION

This paper overcomes the suboptimality in holistic XML tree pattern matching algorithms. The TreeMatch algorithm as described in [1] is explored and the processing of all three types of XML tree pattern matching queries with the help of dewey labeling scheme. A prototype application is built to demonstrate the efficiency of TreeMatch algorithm and tested with extensively with all three kinds of queries. The experimental results reveal that the algorithm is capable of avoiding retrieval of intermediary results before obtaining final results. It does mean that it overcomes the problem of suboptimamlity that existed in the previous algorithms.

REFERENCES

[1] Jiaheng Lu, Tok Wang Ling, Senior Member, IEEE, ZhifengBao, and Chen Wang.Extended XML Tree Pattern Matching: Theories and Algorithms.IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 3, MARCH 2011
[2] R. Goldman and J. Widom, “Dataguides: Enabling QueryFormulation and Optimization in Semistructured Databases,”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 436-445, 1997.
[3] Q. Li and B. Moon, “Indexing and Querying XML Data forRegular Path Expressions,” Proc. Int’l Conf. Very Large Data Bases(VLDB), pp. 361-370, 2001.
[4] N. Bruno, D. Srivastava, and N. Koudas, “Holistic Twig Joins:Optimal XML Pattern Matching,” Proc. ACM SIGMOD, pp. 310-321, 2002.
[5] H. Jiang et al., “Holistic Twig Joins on Indexed XML Documents,”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[6] C.Y. Chan, W. Fan, and Y. Zeng, “Taming Xpath Queries byMinimizing Wildcard Steps,” Proc. Int’l Conf. Very Large Data Bases(VLDB), pp. 156-167, 2004.
[7] W. Wang, H. Wang, H. Lu, H. Jiang, X. Lin, and J. Li, “EfficientProcessing of XML Path Queries Using the Disk-Based F&B Index,”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 145-156, 2005.
[8] N. Bruno, D. Srivastava, and N. Koudas, “Holistic Twig Joins:Optimal XML Pattern Matching,” Proc. ACM SIGMOD, pp. 310-321, 2002.
[9] S. Chen, H.-G.Li, J. Tatemura, W.-P.Hsiung, D. Agrawal, and K.S.Candan, “Twig2stack: Bottom-Up Processing of Generalized-Tree-Pattern Queries over XML Document,” Proc. Int’l Conf. Very LargeData Bases (VLDB), pp. 19-30, 2006.
[10] H. Jiang et al., “Holistic Twig Joins on Indexed XML Documents,”Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[11] J. Lu, T.W. Ling, C. Chany, and T. Chen, “From Region Encodingto Extended Dewey: On Efficient Processing of XML Twig PatternMatching,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 193-204, 2005.
[12] P. ONeil, E. O’Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury,“ORDPATHs: Insert-Friendly XML Node Labels,” Proc. ACMSIGMOD, pp. 903-908, 2004.
[13] H.V. Jagadish and S. AL-Khalifa, “Timber: A Native XMLDatabase,” technical report, Univ. of Michigan, 2002.

[14] M. Moro, Z. Vagena, and V.J. Tsotras, "Tree-Pattern Queries on aLightweight XML Processor," Proc. Int'l Conf. Very Large DataBases (VLDB), pp. 205-216, 2005.
[15] C. Zhang, J.F. Naughton, D.J. DeWitt, Q. Luo, and G.M. Lohman,"On Supporting Containment Queries in Relational DatabaseManagement Systems," Proc. ACM SIGMOD, pp. 425-436, 2001.
[16] I. Tatarinov, S. Viglas, K.S. Beyer, J. Shanmugasundaram, E.J.Shekita, and C. Zhang, "Storing and Querying Ordered XMLUsing a Relational Database System," Proc. ACM SIGMOD,pp. 204-215, 2002.
[17] P. ONeil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury,"ORDPATHs: Insert-Friendly XML Node Labels," Proc. ACMSIGMOD, pp. 903-908, 2004.
[18] B. Choi, M. Mahoui, and D. Wood, "On the Optimality of theHolistic Twig Join Algorithms," Proc. 21st Int'l Conf. Database andExpert Systems Applications (DEXA), pp. 28-37, 2003.
[19] M. Shalem and Z. Bar-Yossef, "The Space Complexity ofProcessing XML Twig Queries over Indexed Documents," Proc.24th Int'l Conf. Data Eng. (ICDE), 2008.
[20] H. Wang and X. Meng, "On the Sequencing of Tree Structures for XML Indexing," Proc. 21st Int'l Conf. Data Eng. (ICDE), pp. 372-383, 2005.
[21] P. Rao and B. Moon, "PRIX: Indexing and Querying XML UsingPrufer Sequences," Proc. 20th Int'l Conf. Data Eng. (ICDE), pp. 288-300, 2004.

Biography

Sravan Kumar K is a student of DRK Institute of science and Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech degree in Computer Science and Engineering and M.Tech Degree in Software Engineering. His main research interest includes Software Engineering, Enterprise Resource Planning.

Madhu P is a student of DRK College of Engineering & Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech Degree in Computer Science and Engineering and M.Tech Degree in Computer Science. His main research interest includes Software Engineering.

Raghava Rao N is working as Associate Professor at DRK Institute of Science & Technology, Ranga Reddy, and Andhra Pradesh, India. She has received M.Tech Degree in Computer Science. His Main Interest includes Cloud Computing, Software Engineering.