AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Bioinformatics Data Compression and Retrieval Based on XML Structured Indexed Tree

[1]Baydaa Al-Hamadan and [2]Raad Alwan

[1]*Assistant Prof., Zarqa University, Department of Computer Science, Zarqa, Jordan.*
[2]*Associate Prof., Philadelphia University, Department of Computer Science, Amman, Jordan.*

ARTICLE INFO

Article history:
Received 27 January 2014
Received in revised form 5 March 2014
Accepted 8 March 2014
Available online 5 April 2014

Keywords:
XML, Bioinformatics, information retrieval, XML compression, XQuery language.

ABSTRACT

Background: Bioinformatics data are the description of all the chemical interactions or the protein structure to form the gene in the living bodies. To make these data easy to be stored, transmit, retrieved, and unified the best way is to represent these data as XML representation. However, XML documents suffer from high redundancy in its structure. **Objective:** This paper produces a new XML compressor (BioXComp) to compress the Bioinformatics XML documents (Bio-XML) and to retrieve information from the compressed XML documents without the need to decompress them. The proposed algorithm depends on generating the Structure Indexed Tree (SIT) for the Bio-XML document to use it for the retrieval purposes according to different types of queries. **Results:** BioXComp achieves 68.7% compression ratio and retrieves information based on different kinds of XQuery queries. **Conclusion:** As the importance of using XML documents in representing the Bioinformatics data increases, the need to compress these data is increasing as well to transmit these documents using less bandwidth. Instead of decompressing these files each time the user needs to retrieve a specific portion from them, BioXComp provides a way to retrieve the information from the compressed data using different types of XQuery query language.

To Cite This Article: Baydaa Al-Hamadani, Raad Alwan. Bioinformatics Data Compression and Retrieval Based on XML Structured indexed tree. ***Aust. J. Basic & Appl. Sci.,*** *8(3): 35-42, 2014*

INTRODUCTION

Nowadays, the most important issue dealing with data-intensive science is the way of storing, transmitting, and retrieving these data efficiently rather than collecting them. This problem magnifies when the collected data come from different sources and exchange these data among different platforms and different research teams to unify their experiments. One such field is the use of bioinformatics data.

Bioinformatics data are the description of all the chemical interactions or the protein structure to form the gene in the living bodies. These data are heterogeneous in their structures which mean that they are represented in many different kinds. They need to be unified in one type for transmission and retrieving purposes.

The cure to all these problems is the storing of these data in XML (eXtensible Markup Language) format. Several features of XML make it to be considered nowadays as the most important platform for representing and transferring data on the web. The ability to extend XML documents, their readability by the user, the interoperability and being read by different platforms, and their ability to represent different types of data representation are some of the important features for XML documents.

The features of XML just fit the properties of the bioinformatics data. These data are growing exponentially and they need to be represented, manipulated and retrieved in an efficient and easy way. Moreover, the bioinformatics data are disseminated in different data representations such as text and images. It would be very difficult for the users of these data if each type is represented in different way and needs different applications to be processed.

However, some of the bioinformatics data is one of the *regular documents* where the document ratios occupy between 40% and 60% of their size. While most of these documents are considered to be structural documents, which have less than 30% of data ratio. This means that the data itself could takes terabytes to be stored but the elements and tags of XML structure will enlarge these documents significantly. The ratio of the structure of the XML document varies depending on the type of the Bioinformatics document but it could take 28% to 57% of the file size. The problem occurs when the users of XML-bioinformatics data needs to transfer

Corresponding Author: Baydaa Al-Hamadani, Zarqa University, Department of Computer Science, Faculty of Sciences and Information Technology, Box.3030. Zarqa. Jordan.
Ph: (+962) 775663578. P.O.Box 132222. 13132 Jordan

these data trough networking. The need to compress these files and retrieves information from the compressed version with the need to no or just little decompression is an important issue when dealing with this type of data.

This paper proposes BioXComp as s new compression technique that abridges the bio-XML document to reduce the size used by the its structure. The compressed version is queriable using any kind of queries, starting from exact-match queries to range queries, using XQuery language.

Background:

Compressing XML documents is a hot field during the last decade and till now. All the existing XML compressors were designed to compress different types of these documents and there is no specialized compressor for bioinformatics data. However, several attempts have been made to compress XML documents.

There are two types of XML compressors, (1) queriable compressors which have the ability to query the compressed documents without the need to decompress them, and (2) the non-queriable compressors which concentrate on getting higher compression ratio and to query the compressed XML document; they have to completely decompress them first. Since the focus in this paper is to design a queriable XML compressor, this section listing the well-known queriable XML compressors.

The main goal of queriable XML compressors is to provide the ability to the compressed version of the XML document to be queried without complete decompression them. The compression ratio for these compression techniques is lower than the non-queriable techniques. Some of these techniques are homomorphic compressors, which mean that the compressed file is a semi-structured file.

The first queriable compressor is *XGrind* by (Tolani and Haritsa, 2000). This technique replaces the elements and attributes names with the letters followed by a unique identifier which represents the substituted element or attribute name. The data part of the document is encoded using Huffman encoding. For the purpose of querying the compressed document, XGrind’s query processor finds the simple path to check whether it satisfies the path in the given query. The main drawback with XGrind is its ability to process exact-match and prefix-match queries only.

Xpress (Min *et al*., 2003) uses the *reverse arithmetic encoding* method to encode the label paths of the XML document as a distinct interval in [0.0, 1.0) . Using the relationships between these intervals will allow for the ability to evaluate path expressions more efficiently on the compressed XML document. To encode the data part of the XML document, *XPress* uses different compression techniques depending on the type of the data and without the need to the human interference.

Because *XGrind* and *Xpress* are homomorphic, the relationship between the size of the compressed document and the size of the original one is linear. To solve this problem (Cheng and NG, 2004) proposed a new technique (*XQzip*) that depends on extracting the *Structure Index Tree (SIT)* from the tree structure of the original document. The SIT depth is non-linear to the structure tree which makes this technique accomplishes higher compression ratio and faster query evaluation.

Instead of using (SIT), *XQueC* (Arion *et al*., 2007) uses the *structure summary tree* in order to efficiently stores the XML documents. The space needed to store the structure summary (*SS*) is:

$$CSaux = \sum_{n \in SS} (|tag(n)| + \log_2(|SS|))$$

(Müldner *et al*., 2009) created an annotation tree to succinctly store the structure of the XML document and use the containers to store the data part of the document. Their compressor, named *XSAQCT*, has two versions; the first was dependent on the XML Schema and the second was schema-free. They showed that the first version is better than the second from the standpoint of compression ratio even though it was slower.

(Arroyuelo *et al*., 2010) proved in their proposed *SXSI* compressor that the XPath queries can be performed better when using an indexing technique to compress the XML document. This technique is based on producing a labelled tree from the XML Tree structure and then indexing this tree into a bit array and compressing the data part of the document using a general back-end compressor.

Lately, (Qian *et al*., 2012) suggested a new compression technique by separating the structure of the XML document from its context and compress them respectively during their arrival time.

Meanwhile, (Szalapski *et al*., 2012) proposed a new XML compressor that reduce latency and bandwidth usage further in real time wireless applications. It shows significant results comparing with previous compressors.

BioXCompDesign:

The design of BioXComp depends on abridging the structured tree of the XML document to eliminate the structure part of the document which could take up to 57% of the original size of the document.

As illustrated in Figure 1, the first step on compressing the XML document is to parse these documents using the *Document Object Model* (DOM) parser. This type of parsing technique represents the whole XML document as a tree-like structure to make it easier to traverse this tree and collect the path from the root to the specific element. The result of the DOM parsing is the index structured tree (IST) and the ID of all the tags stored in a hash table to reduce the time required to retrieve them.

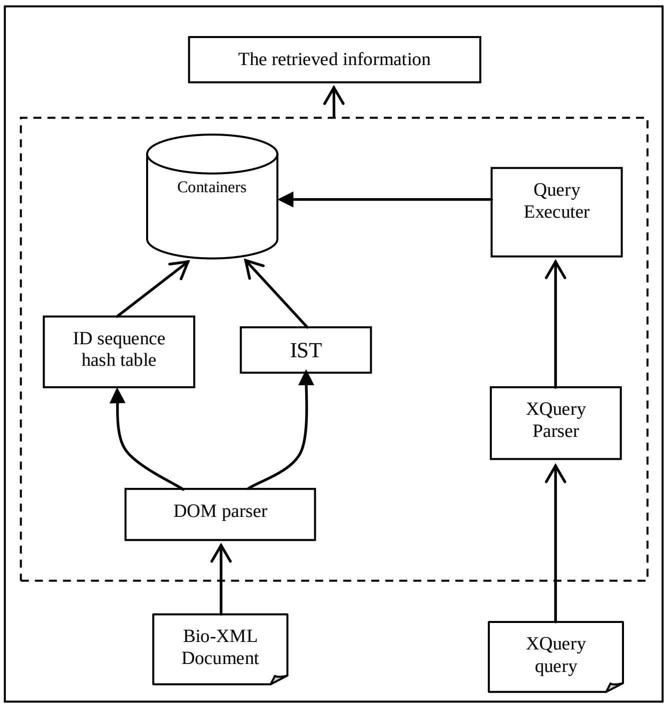

Fig. 1: The design of BioXComp.

A. BioXComp compressor:

From (swissprot.xml), which represent the description of all the protein, the description of only one protein takes (110) nodes when it changed to tree-like structure. Figure 3 illustrated the description of the same protein in SIT and it takes (27) nodes only. This process abridges the original XML tree by more than 75%.

Meanwhile, each node in the generated SIT is going to have a specific index according to the nodes entry in a hash table. This process is useful to abridge more nodes when there is more than one node with the same name. As an example, the SIT in Figure 3 has more than one node with the name “from” and “to”. Depending on the generated SIT and its relevant hash table, the second step in compressing the XML document is to generate the containers. Each container has an index that represents the complete path starting from the root to a specific leaf. Inside the containers, all the data part that lies under the same root of the document is stored. Figure 2 shows examples of the containers that are generated from the SIT in Figure 3.

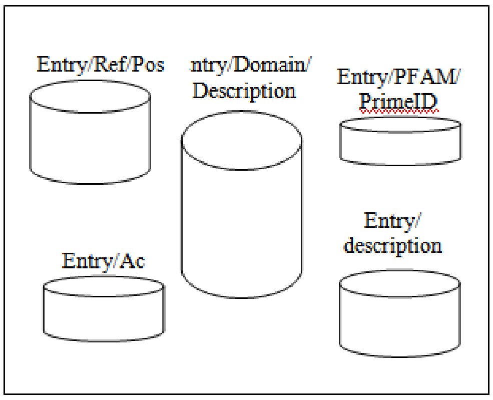

Fig. 2: Samples of the containers from the SIT of swissprot.xml.

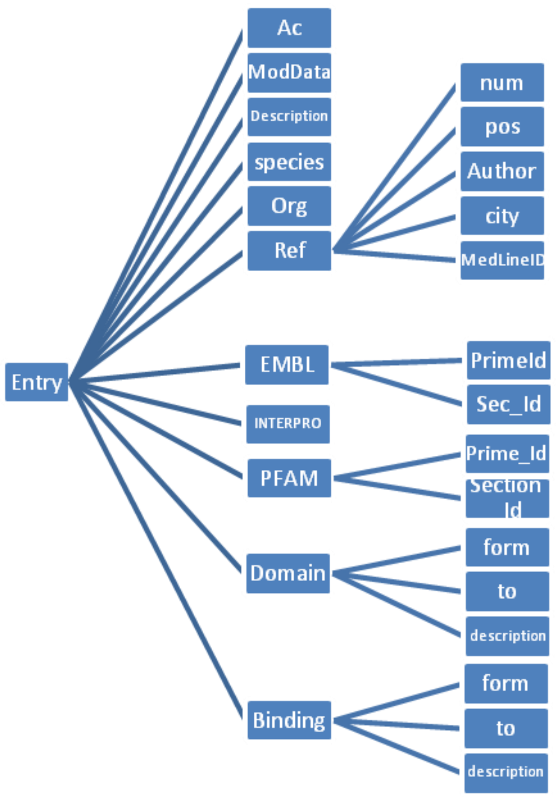

Fig. 3: The Structured Indexed Tree for the SwissProt.xml.

B. Bio X Comp decompressor:

The design of the decompressor is asymmetric. This means that the decompressor technique is not following the same steps of the compressor. To rebuild the same XML document, the decompressor takes the SIT for the corresponding document and its hash table. To check the consistency of the order, *the decompressor* uses the following steps:

1. If *XCVQ-D* has a piece of data **[(σ)d]**, where σ denotes the **ID$_{order}$** accompanied with the data in a container, σ' is the **ID$'_{order}$** which denotes the order of the data written so far in (D'), then the new order of D should be calculated by getting the difference between **(σ)** and the **(σ')** taking into consideration the number of the elements and attribute names still not written in (D'), such that:

$$C_{order} = \sigma - (\sigma' + [\mu_P - \mu_C])$$

Where:

μ_P: The number of elements and attribute names written in D'.

μ_C: The number of elements and attribute names in the index of the container having this data.

2. The value of C_{order} is checked and a performance made as shown in the following equation.

$$C_{order} \begin{cases} = 0 & newID_{order} = \sigma \\ \neq 0 & D' = D' \cup ([\mu_P] - [\mu_C]) \end{cases}$$

3. If C_{order} equals to (0) this means that the current ID_{order} is consistent with the number of elements and attribute names in D'. Otherwise, the difference between the current path in $D'$and the index path for the current container should be added to D' before adding the required data.

C. BioXComp querying:

The most important feature of BioXComp is its ability to retrieve information from more than one XML documents that match the given query without the need to specify the required document in advance. This feature comes from collecting the compressed XML documents into a repository and each time a query is given, all the documents in the repository is going to be searched for relevant information. Only the compresses documents that have a pre-specified relevancy threshold will be chosen as relevant.

To check if a document is relevant to the given query, first, the XPath query should be decomposed first into two main parts, the path and the data. The part is used to select the relevant documents by matching it with the document's hash table. If there is similarity between them then the document is considered to be relevant to the given query and the relevancy value is given as follows:

$$sim(d,q) = \frac{number\ of\ matches\ between\ q\ and\ d.hash\_table}{total\ number\ of\ items\ in\ q}$$

The containers in the relevant XML documents are going to be searched for the data part of the XPath query to determine if the required data are exist. Only the relevant containers are decompressed and viewed to the user as an XML document.

BioXComp has the ability to retrieve information according to several types of queries such as:

1. Simple queries: used to retrieve part of the document according to general specifications.
2. Criteria queries: used to retrieve part of the document according to a specific criterion.
3. Conjunctive queries: used to retrieve part of the document according to conjunction of two or more criteria.
4. Range queries: used to retrieve information according to a range between given minimum and maximum values.

Testing:

For the purposes of testing BioXComp, several factors are used. The factors and the results are listed below:

1. *Compression Ratio (CR):* this factor is used to test the difference between the original XML file size and the compressed file size as illustrated in Eq.(3) (Salomon, 2007). It is used in two stages.

$$CR = 1 - \left(\frac{Size\ of\ compressed\ file}{Size\ of\ original\ file}\right)$$

As illustrated in Figure 4, the compression ratio of the tested XML documents depends on the structure ratio in these documents. The higher the structure the bigger compression ratio exists. The average compression ratio for the entire corpus is 66.12.

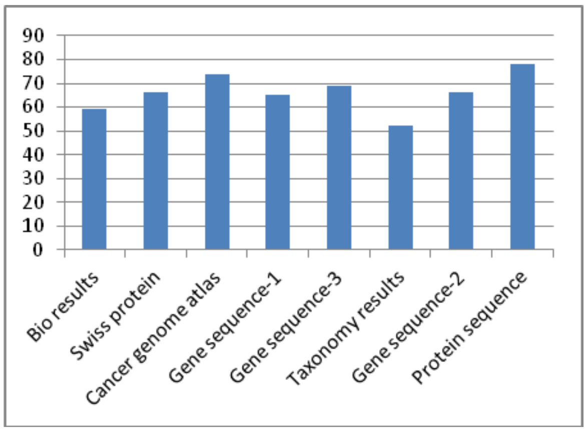

Fig. 4: Compression ratio for the Bio- XML corpus.

2. *Compression Time (CT):* This factor is used to determine the time required to compress each XML document in seconds (s) and to specify its relation with the file size. Fig.5 shows that the time needed to compress the XML document is directly proportion with the size of the document.
3. *Decompression Time (DT):* This is the measure of the time required to decompress the XML document in order to obtain the original one. The effect of the file size on DT was obtained and the results can be seen in Fig.6.
4. *Query Functional Test (QFT):* The purpose of this test is to determine the main types of queries that can be processed by *BioXComp query processor*. For this purpose, a query benchmark in Table 1 was tested.
5. *Query Performance Test (QPT):* This factor is used to determine the time required to process each of the XQuery query in the benchmark and retrieve the relevant results. Fig. 7 shows the detailed test for all the query types listed in Table 1.

It is clear that the (Axes) query type takes less time to be performed. This is due to the architecture of BioXComp that uses the path of each node as the index to the container that has this node. While (Function) query type need more time to be processes and retrieve the required information since they need more than one stage. The performance process of this type of queries starts by looking for the required data and then performs the specific mathematical or string manipulation functions.

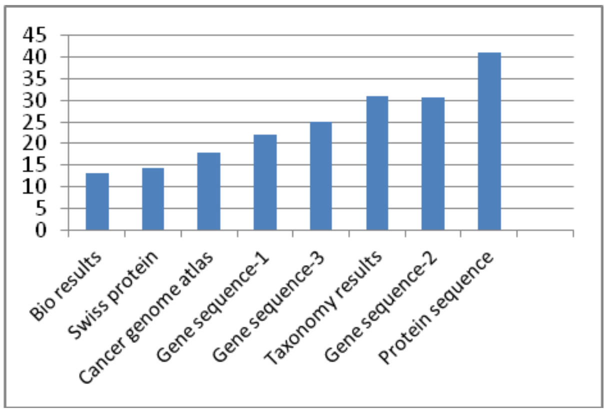

Fig. 5: Compression time for the Bio-XML corpus.

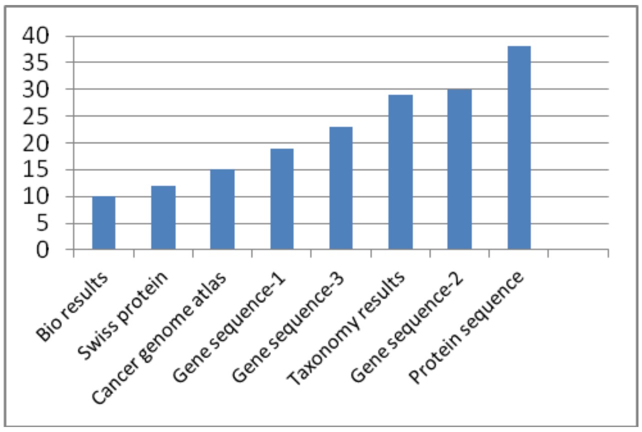

Fig. 6: Decompression time for the Bio-XML corpus.

Table 1: XQuery-FT query benchmark.

QFT concepts	Description
Axes	parent, descendant, preceding
Filters	predicates
Node Test	Comment(), text(), node()
Operators	Relational operators (<, =,…) and Boolean operators (and, or)
Functions	String manipulating functions and mathematical functions

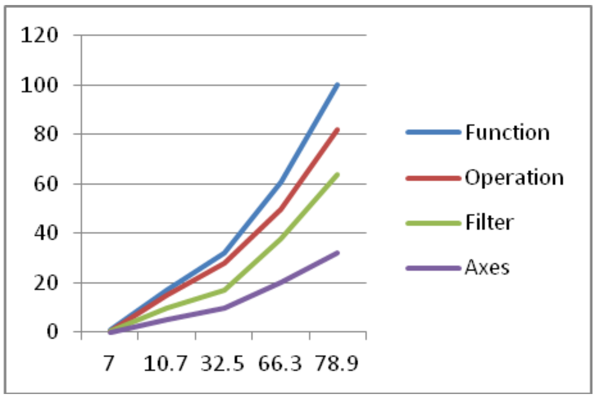

Fig. 7: XQuery-QP testing.

Conclusion and future work:

As the importance of using XML documents in representing the Bioinformatics data increases, the need to compress these data is increasing as well to transmit these documents using less bandwidth. Instead of decompressing these files each time the user needs to retrieve a specific portion from them, BioXComp provides a way to retrieve the information from the compressed data using different types of XQuery query language. This process will save the time needed to decompress these files for retrieving purposes and compressing them again to transfer them. Without the need to the Schema or DTD for the specific XML document, BioXComp has the ability to compress these documents with average compression ratio equals to 68.7%.

Our future work is to increase the average compression ratio by depending on the Schema or the DTD of the XML documents to group each part of data, which have the same type, in containers and using specific compression technique suitable for each data type. Another development will be made on BioXComp is to retrieve information according to vague queries, a query that does not match the syntax of the query language.

ACKNOWLEDGEMENT

The author acknowledge financial support from the Deanship of Research and Graduate studies in Zarqa University/ Jordan

REFERENCES

Achard, F., G. Vaysseix, E. Barillot, 2001. XML, bioinformatics and data integration. *Oxford Journals/ Bioinformatics,* 17: 115-125.

Arion, A., A. Bonifati, I. Manolescu, A. Pugliese, 2007. XQueC: A query-conscious compressed XML database. *ACM Trans. Internet Technol.,* 7: 10.

Arroyuelo, D., F. Claude, S. Maneth, V. M¨Akinen, G. Navarro, K. Nguyen, J. Sir´En, N. V¨Alim¨Aki, 2010. Fast In-Memory XPath Search using Compressed Indexes. In Proceedings of the IEEE Twenty-Sixth International Conference on Data Engineering (ICDE 2010), California, USA.

Cheney, J., 2006. Tradeoffs in XML Database Compression. DCC '06 Proceedings of the Data Compression Conference, pp: 392-401.

Cheng, J., W. Ng, 2004. XQZip: Querying Compressed XML using Structural Indexing. International Conference on Extending Data Base Technology (EDBT).

Ganesan, N., B. Kalyanasundaram, M. Velauthapllai, 2007. Bioinformatics Data Profiling Tools: A Prelude to Metabolic Profiling. *Pacific Symposium on Biocomputing,* 12: 127-132.

Gulhane, V.S., M.S. Ali, 2012. Compressed XML Database and Query Evaluation over XML Databases. *International Journal of Advanced Research in Computer Science and Electronics Engineering*, 1.

He, Y., R.R. Vines, A.R. Wattam, G.V. Abramochkin, A.W. Dickerman, J.D. Eckart, B.W.S. Sobral, 2005. PIML: the Pathogen Information Markup Language, 21: 116-121.

Jupp, S., J. Klein, J. Schanstra, R. Stevens, 2011. Developing a kidney and urinary pathway knowledge base. *Journal of Biomedical Semantics,* 2.

Mesiti, M., E. Jiménez-Ruiz, I. Sanz, R. Berlanga-Llavori, P. Perlasca, G. Valentini, D. Manset, 2009. XML-based approaches for the integration of heterogeneous bio-molecular data. *BMC Bioinformatics,* 10.

Min, J.K., M.J. Park, C.W. Chung, 2003. XPRESS: a queriable compression for XML data. Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM, San Diego, California, 122-133.

Müldner, T., C. Fry, J.K. Miziołek, S. Durno, 2009. XSAQCT: XML Queryable Compressor. Balisage: The Markup Conference.

Pinho, A.J., D. Pratas, S.P. Garcia, 2011. GReEn: a tool for efficient compression of genome resequencing data.

Qian, B., H. Wang, J. Li, H. Gao, Z. Bao, Y. Gao, Y. Gu, L. Guo, Y. Li, J. Lu, Z. Ren, C. Wang, X. Zhang, 2012. Path-Based XML Stream Compression with XPath Query Support Web-Age Information Management. Springer Berlin / Heidelberg.

Raymond, W., A. Vo Ngoc, A. Kiyoshi, 2012. Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics,* 28: 628-635.

Saadawi, G.M., J.H. Harrison, 2006 Definition of an XML Markup Language for Clinical Laboratory Procedures and Comparison with Generic XML Markup. *Clinical Chemistry,* 52: 1943-1951.

Seibel, P.N., J. Krüger, S. Hartmeier, K. Schwarzer, K. Löwenthal, H. Mersch, T. Dandekar, R. Giegerich, 2006. XML schemas for common bioinformatic data types and their application in workflow systems. *BMC Bioinformatics,* 7.

Sroka, J., G. Kaczor, J. Tyszkiewicz, A.M. Kierzek, 2006. XQTav: an XQuery processor for Taverna environment. *BIOINFORMATICSAPPLICATIONS NOTE,* 22: 1280-1281.

Strömbäck, L., V. Ivanova, D. Hall, 2011a. Exploring Statistical Information for Applications-Specific Design and Evaluation of Hybrid XML storage. The Third International Conference on Advances in Databases, Knowledge, and Data Applications.

Strömbäck, L., V. Ivanova, D. Hall, 2011b. Using Statistical Information for Efficient Design and Evaluation of Hybrid XML Storage. *International Journal on Advances in Software,* 4: 389-400.

Szalapski, T., S. Madria, M. Linderman, 2012. TinyPack XML: Real time XML compression for wireless sensor networks. Wireless Communications and Networking Conference (WCNC), 2012 IEEE, 3165-3170.

Tolani, P.M., J.R. Haritsa, 2000. XGRIND: A Query-friendly XML Compressor. IEEE 18th international conference on Data Engineering.

Vyas, H., R. Summers, 2005. Interoperability of bioinformatics resources. *VINE,* 35: 132-139.

Weile, J., M. Pocock, S.J. Cockell, P. Lord, J.M. Dewar, E.M. Holstein, D. Wilkinson, D. Lydall, J. Hallinan, A. Wipat, 2011. Customizable Views on Semantically Integrated Networks for Systems Biology. *Bioinformatics,* 27: 1299-1306.