# Application of Tree Mining to Matching of Knowledge Structures of Decision Tree Type

Fedja Hadzic[1] and Tharam S. Dillon[1]

[1]Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology, Perth, Australia
fedja.hadzic@postgrad.curtin.edu.au
tharam.dillon@cbs.curtin.edu.au

**Abstract.** Matching of knowledge structures is generally important for scientific knowledge management, e-commerce, enterprise application integration, etc. With the desire of knowledge sharing and reuse in these fields, matching commonly occurs among different organizations on the knowledge describing the same domain. In this paper we propose a knowledge matching method which makes use of our previously developed tree mining algorithms for extracting frequent subtrees from a tree structured database. Example decision trees obtained from real world domains are used for experimentation purposes whereby some important issues that arise when extracting shared knowledge through tree mining are discussed. The potential of applying tree mining algorithms for automatic discovery of common knowledge structures is demonstrated.

## 1 Introduction

Knowledge discovery task in general can be hard and time consuming, and hence sharing the already developed knowledge representations is desirable. This particularly occurs when a number of organizations coming from the same knowledge domain would like to have a knowledge basis on which they can integrate their organization specific knowledge. Having a shared knowledge model would save time and costs associated with acquiring general knowledge about the domain at hand.
Matching of knowledge structures has been of interest for a long time and many useful applications can be found in e-commerce, enterprise application integration and the general management of scientific knowledge. The emergence of semi-structured data sources (such as XML) which are commonly used for describing domain knowledge has called for the development of methods for matching of tree structured documents. The TreeDiff software package [1] takes two documents as input, represents them as ordered labeled trees and finds sequence of edit operations to transform one document tree into another. Finding common structures among semi-structured documents is useful for document clustering methods, since the structure is usually ignored by traditional clustering methods [2]. An algorithm which finds largest approximate common substructures between ordered labeled trees has been presented in [3]. A tree inclusion algorithm presented in [4] checks for the inclusion

of a query tree that can be of an embedded subtree type. Another related area where matching of tree structured knowledge models will be useful is in a classification ensemble which is a multi-classifier system where each classifier is developed for the same domain problem [5, 6]. Hence, if the classifiers are of decision tree type the shared knowledge structure would indicate the general knowledge of the domain. The new classifier is then expected to have better generalization capability and hence be more accurate in classifying future unseen data objects.

The method presented in this paper is based on the use of our previously developed tree mining algorithms in order to automatically extract common knowledge structures. While in this paper we focus on matching of decision tree, the method is generally applicable to matching of tree structured knowledge representations. By using the tree mining approach to knowledge matching many of the structural differences among the knowledge representations can be efficiently detected and the largest common structure is automatically extracted. The implications of mining different subtree types are discussed and for knowledge matching task the most suitable subtree type within the current tree mining framework is indicated. Decision trees obtained from real world datasets are used for experimental purposes. We have previously applied our algorithms on large and complex tree structures and experimentally demonstrated their scalability [7, 8]. In this paper we consider smaller decision trees in order to illustrate the underlying concept in a more comprehensible manner.

## 2 Frequent Subtree Mining

This section starts by providing the tree mining related definitions necessary for the purpose of describing the present work. We then proceed with an overview of our contributions in the area of frequent subtree mining. Please note that due to space and scope limitations we do not explain the details of our algorithms and the various issues involved when developing tree mining algorithms. For a more detailed overview of the area including related works and algorithm comparisons one may refer to our previous works [7, 8, 9], where such information has been provided

A tree can be denoted as $T(v_0,V,L,E)$, where (1) $v_0 \in V$ is the root vertex; (2) $V$ is the set of vertices or nodes; (3) $L$ is the set of labels of vertices, for any vertex $v \in V$, $L(v)$ is the label of $v$; and (4) $E = \{(x,y) | x,y \in V\}$ is the set of edges in the tree. A *root* is the topmost node in the tree. The *Parent* of node $v$ is defined as the predecessor of node $v$. A node $v$ can only have one parent while it can have one or more *children*. A node without any child is a *leaf* node; otherwise, it is an *internal* node. If for each internal node, all the children are ordered, then the tree is an *ordered tree*. The number of children of a node is commonly termed as *fan-out/degree* of the node. A path from vertex $v_i$ to $v_j$, is defined as the finite sequence of edges that connects $v_i$ to $v_j$. The length of a path $p$ is the number of edges in $p$. If $p$ is an *ancestor* of $q$, then there exists a path from $p$ to $q$.

The problem of frequent subtree mining can be generally stated as: given a tree database $T_{db}$ and minimum support threshold ($\sigma$), find all subtrees that occur at least $\sigma$ times in $T_{db}$. Within this framework the two most commonly mined types of subtrees

are induced and embedded. An induced subtree preserves the parent-child relationships of each node in the original tree. In addition to this, an embedded subtree allows a parent in the subtree to be an ancestor in the original tree and hence ancestor-descendant relationships are preserved over several levels. Formal definitions follow.

**Induced subtree.** A tree $T'(r', V', L', E')$ is an *induced subtree* of a tree T $(r, V, L, E)$ *iff* (1) $V' \subseteq V$, (2) $E' \subseteq E$, (3) $L' \subseteq L$ and $L'(v)=L(v)$, (4) $\forall v' \in V'$, $\forall v \in V$, $v'$ is not the root node, and $v'$ has a parent in $T'$, then parent($v'$)=parent($v$),

**Embedded subtree.** A tree $T'(r', V', L', E')$ is an embedded subtree of a tree $T(r, V, L, E)$ iff: (1) $V' \subseteq V$; (2) *if* $(v_1, v_2) \in E'$ *then parent*$(v_2) = v_1$ *in* $T'$ *and* $v_1$ *is ancestor of* $v_2$ *in* $T$; if $v' \in V'$, $v \in V$ and $v'$ is not the root node, set *ancestor*$(v')$ $\subseteq$ set *ancestor*$(v)$ and set *ancestor*$(v') \neq \emptyset$; $L' \subseteq L$ and $L'(v)=L(v)$.

**Level of embedding.** If $T'(r', V', L', E')$ is an embedded subtree of $T$, and there is a path between two nodes $p$ and $q$, the *level of embedding* ($\delta$) is defined as the length of the path between $p$ and $q$, where $p \in V'$ and $q \in V'$, and $p$ and $q$ form an ancestor-descendant relationship. A *maximum level of embedding* ($\Phi$) is the limit on the level of embedding between any $p$ and $q$. In other words, given a tree database $T_{db}$ and $\Phi$, then any embedded subtree to be generated will have the maximum length of a path between any two ancestor-descendant nodes equal to $\Phi$.

In addition to the previous definitions, the subtrees can be further distinguished based upon the ordering of sibling nodes. An ordered subtree preserves the left-to-right ordering among the sibling nodes in the original tree while in an unordered subtree this ordering is not preserved. In other words, for an unordered subtree the order of the siblings (and the subtrees rooted at sibling nodes) can be exchanged and the resulting subtree would be considered the same. Examples of different subtree types are given in Fig. 1 below.

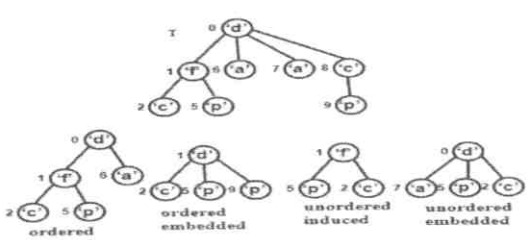



**Fig. 1.** Example of a tree T and its different subtree types

To determine the frequency of a subtree, most commonly used support definitions are transaction based and occurrence match support [7] and the choice is application dependant. Transaction based support (TS) is used when only the existence of items within a transaction is considered important, whereas occurrence match (OC) support takes the repetition of items in a transaction into account and counts the subtree occurrences in the database as a whole. Recently, we have proposed the hybrid support definition [10]. Using hybrid support threshold of x|y, a subtree is considered frequent iff it occurs in x transactions and it occurs at least y times in each of those x

transactions. Hence, it keeps the extra information about the intra-transactional occurrences of a subtree.

Our work in the area of frequent subtree mining is characterized by adopting a Tree Model Guided (TMG) [11, 7] candidate generation. This non-redundant systematic enumeration model ensures only valid candidates are generated which conform to the actual tree structure of the data. Embedding List [11] representation of the tree structure has allowed for an efficient implementation of the TMG approach which resulted in efficient algorithm MB3-Miner [11] for mining of ordered embedded subtrees, which was accompanied with the TMG mathematical model for estimating the worst case complexity of enumerating all embedded subtrees. The large complexity of mining embedded subtrees motivated our Level of Embedding [7] constraint so that one can decrease the level of embedding constraint gradually down to 1, from which all the obtained subtrees are induced. Razor algorithm [12] was a further extension developed for mining embedded subtrees where the distance of nodes relative to the root of the subtree needs to be considered. Using the general TMG framework the UNI3 [9] and U3 [8] algorithms mine unordered induced and embedded subtrees, respectively. From the application perspective, in [13] we have indicated the potential of the tree mining algorithms in providing interesting biological information when applied to tree structured biological data.

## 3 Overview of the approach

In this section we provide a brief overview of our approach to knowledge matching. It is motivated by the real world scenario of different organizations from the same domain obtaining their knowledge models separately and then in future wanting to match their models in order to obtain a shared understanding of the domain. In real world, it is common that different organizations use different names for same concepts but this is a different problem of concept matching which is out of the scope of the current work. Hence, the current assumption is that all concept (attributes and constraints) names are the same or if not some concept matching algorithm has already been applied to find the corresponding mappings.

Fig. 2 illustrates the experimental setup in this paper used for demonstrating how knowledge matching can be achieved through the application of tree mining. Suppose that there are four different organizations in possession of database (DB) for a particular domain. They all use the data considered relevant for the domain to build a knowledge model (KM) by using some existing data mining tools. Each KM is most likely to differ in the way the knowledge is structured and the amount of concept granularity. However since they are describing the same domain there are usually parts of knowledge equivalent among different KMs. This is where the tree mining algorithms will prove useful since they are capable of efficiently extracting sub-structures from large tree databases. More specifically, the shared knowledge model will be obtained by extracting the largest common sub-structure among the available knowledge models.

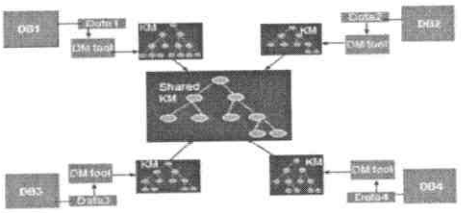



**Fig. 2.** Illustration of the knowledge matching approach.

The obtained knowledge model could be less specific than the knowledge model from a particular data source, but it is therefore valid for all organizations. Furthermore, each of the different organizations could have their own specific part of knowledge which is only valid from their perspective, and which can be added to the shared knowledge model so that every aspect for that organization is covered. Hence, the shared knowledge model can be used as the basis for structuring the knowledge for that particular domain and different communities of users can extend this model when required for their own organization specific purposes.
In this section we have described a real world scenario which we aim to mimic in this paper and details are provided in the sections that follow. However, the core of our proposed method is the application of tree mining for extracting the common knowledge structures from knowledge models provided by different organizations.

## 4 Experimental Setup

The publicly available datasets from the UCI machine learning repository [14] were used. In real world scenarios, the data collected by different organizations can differ in various aspects. The instance collection may differ and certain attributes may be found important by one organization and irrelevant by another. To account for these possible differences in our experiments the subset of data used as input to the data mining tools was different in some cases, and at times we used a feature selection criterion [15] to include only a subset of domain attributes for learning. The end result is that certain models will be more general or specific than others. The described methods of adding variability among knowledge representations were performed only if the classification accuracy remained sufficiently high. In some cases the application of these methods did not have much effect on the obtained decision tree and hence in our domains the number of decision trees considered may differ.
Once the knowledge models are obtained they are all represented as a separate subtree in the tree database. In this regard each of the knowledge models can be viewed as a separate transaction within the tree database. This kind of representation is suitable for the application of tree mining algorithms for the analysis of independent knowledge structures. Since in this work we are not considering the problem of concept matching we are assuming that the concepts describing the same aspect (attribute or constraint) of a domain have the same label. Next, we describe each of the domains considered and show the different knowledge models obtained.

*Wisconsin Breast Cancer Database.* This dataset consists of 10 attributes describing the clinical cases. The task is to predict whether a clinical instance is classified as benign or malignant. For this domain we could only obtain two knowledge models that are different while still being sufficiently accurate. These are displayed in Fig. 3.

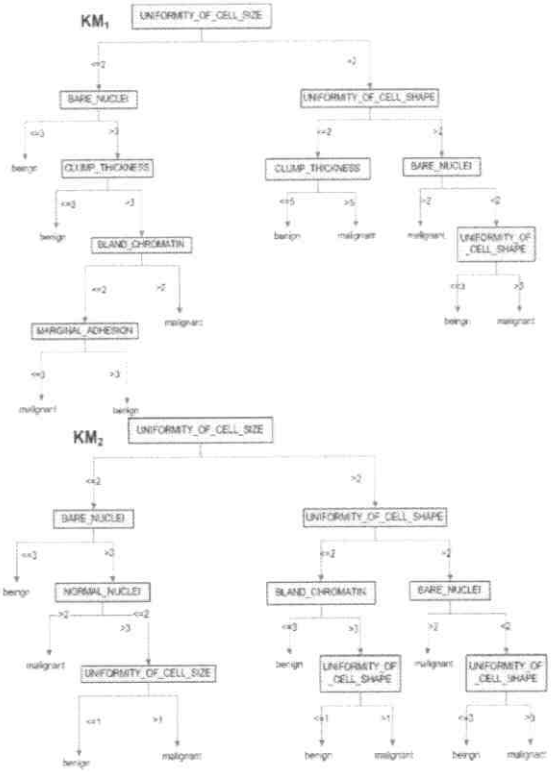
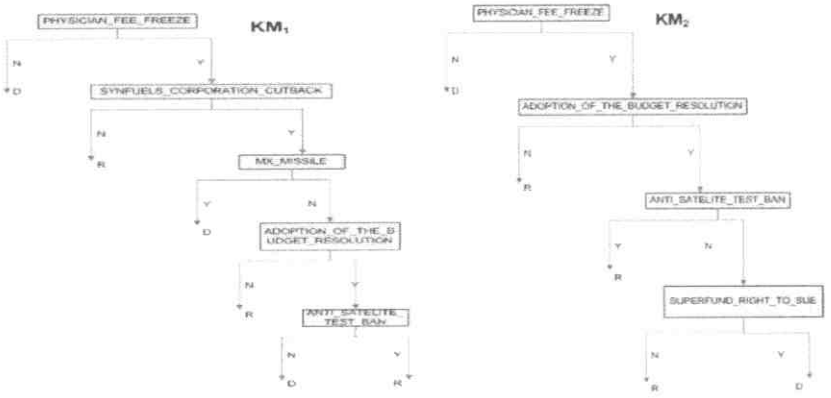



**Fig. 3.** Knowledge models for breast-cancer domain (class values: benign, malignant)



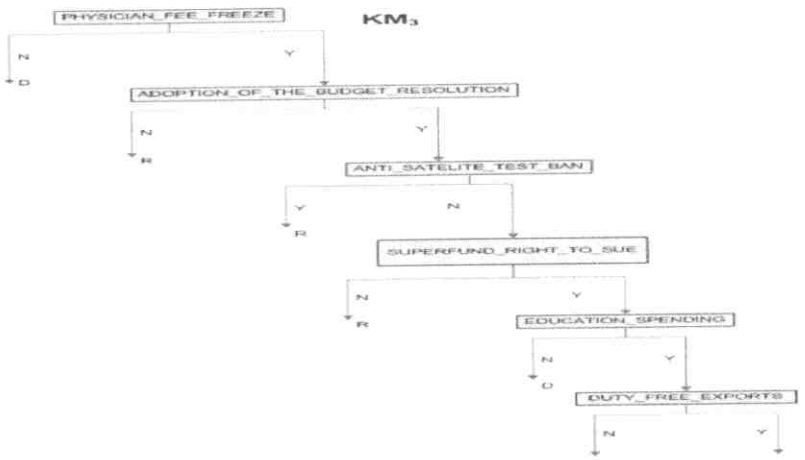



**Fig. 4.** Knowledge models for voting domain (class values: D-democrat, R-republican).

*Voting Records.* This data set describes the 1984 United States Congressional Voting Records Database [14]. It consists of 16 Boolean attributes indicating a certain policy and the classification task is to predict whether the instance describes a democrat or a republican. For this domain we obtained three different and accurate knowledge models, which are displayed in Fig.4.

Please note that for the ease of representation the edges in the displayed knowledge models are labeled, rather than being considered as separate nodes themselves. However, within the current tree mining framework the edge labels are usually not considered. Each of the labeled edges from the picture above becomes a node itself with its parent node being the node from which the edge is emanating, whereas its child node is the node to which the edge is connecting.

## 5 Results and Discussion

When comparing decision trees the order of sibling nodes is not considered important, and hence unordered subtree mining is more suitable. One further choice to make is whether we are interested in mining induced or embedded subtrees. The difference occurs in the fact that if embedded subtrees are mined we are allowing structures to be considered similar even if they occur at different levels in the tree. In an embedded subtree the relationship is not limited to parent-child and hence by allowing ancestor-descendant relationships enables the extraction of more sub-structures where the levels of embeddings between the nodes are not limited to one and can be different. For example if in knowledge representation 'A' the level of embedding between the nodes 'a' and 'b' representing some domain concepts is much larger than the level of embedding between the nodes representing the same concepts in knowledge

representation 'B', then 'A' stores more specific knowledge about the concept represented by node 'a'. In our examples the extra specific knowledge corresponds to the additional number of attribute constraints used for further separation of class values. Since it is common that knowledge representations could differ in the amount of specific knowledge stored, mining of embedded subtrees is more suitable for general knowledge analysis. Taking these observations into account our U3 [8] algorithm for mining unordered embedded subtrees will be used.

Each of the decision trees is represented as a separate subtree (transaction) within the tree database and transaction based support is used to extract the largest common structure that occurs in each transaction. If we have k different models then a subtree will only be considered frequent if it occurs in all k models. Hence, the transaction based support was used with the thresholds set to 2 and 3 for the 'breast-cancer' and 'voting records' domain, respectively.

In Fig.5 we display the common structure obtained by analyzing the largest frequent unordered embedded subtrees for the breast-cancer domain. Please note that since in this domain we are dealing with a binary classification problem and the knowledge model is represented by a binary decision tree, we filter out any of the subtree patterns where any of the nodes has a degree larger than 2. These patterns would be meaningless as there would be three class values distinguished by two attribute constraints. In other words one attribute constraint would lead to two possible class values which is in itself a contradiction since it defeats the purpose of that constraint.

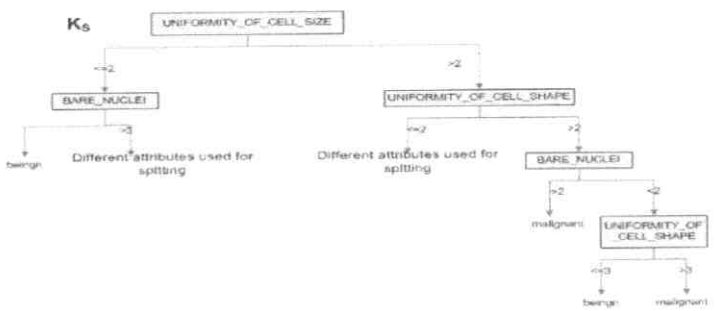



**Fig. 5.** Shared knowledge structure ($K_S$) of the 'breast-cancer' domain.

When analyzing the filtered results there were four largest common subtrees. At the points where both KMs from Fig.3 match the two variations occur with nodes 'benign' and 'malignant' (i.e. class values). This indicates that at these points in the knowledge structures different attributes are used for further class separation (see Fig. 5). In fact if induced subtrees are mined then there would be only one largest common subtree since no further match would be made past the point where concept nodes are different. This however may not always be desirable as will be shown for the 'voting records' domain.

The largest frequent unordered embedded subtree for the voting records domain is displayed in Fig. 6. In this case there was only one largest common subtree, $K_S$. By comparing $K_S$ with the knowledge models from Fig.4 we can see that it was necessary to mine embedded subtrees in order to detect the common knowledge structure.

Looking back at the knowledge models from Fig. 4 it could be said that the $KM_2$ most closely matches the shared structure Ks from Fig.6. $KM_2$ only differs to the Ks in the sense that it has one extra attribute after the last node which splits the class values further into 'D' and 'R', rather than generalizing it to 'R'. The knowledge model $KM_1$ stores two additional attributes after the first attribute node while $KM_3$ stores additional three attributes at the last node. We can see that $KM_2$ is more general than $KM_3$, but $KM_2$ is only more general than $KM_1$ between the first and second attribute while it is more specific at the last node.

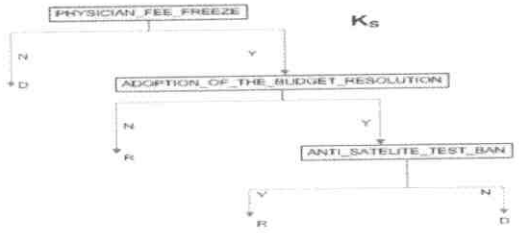



**Fig. 6.** Shared knowledge structure ($K_S$) of the 'voting records' domain.

The examples used indicate the differences between mining induced and embedded subtrees. In the example from Fig. 3 largest induced subtree (Fig. 5) was more suitable whereas for the second example in Fig.4 largest embedded subtree (Fig. 6) allowed for the similarity in structure to be detected besides the differences in specificity. As a guide, one could start with mining of embedded subtrees and if most of the large frequent patterns are meaningless in the sense described earlier, one could only concentrate on induced subtrees. However, contrasting to our examples, the level of embeddings at which the knowledge structures can differ could in reality be very large. To make the large change from mining embedded to induced subtrees runs the risk of missing many other common structures where the difference in the level of embedding is not so great. In these cases the maximum level of embedding ($\Phi$) [7] could be useful so that $\Phi$ is progressively decreased until some differences are resolved. In fact it probably will not provide only one matching KM as for the induced case, but larger common structures would be detected. Which method to adapt is again dependent on the type of knowledge that is being matched as for some applications induced subtrees may be sufficient and the level of embedding can be ignored while for others it is important as it indicates the extra specific knowledge stored in a different knowledge model. Either way even if the user is not a domain expert different options can be tried with respect to $\Phi$ and this should reveal some more detail about the similarities and differences among the knowledge structures.

## 6. Conclusions and Future Work

In this paper we have described a way in which the tree mining algorithms can be effectively used for obtaining the shared knowledge from separate knowledge models of decision tree type. Implications of mining different subtree types were discussed

with the unordered embedded subtree type being the most appropriate for the task. This is our preliminary work in the area and as such the aim was to demonstrate that the application of tree mining to the problem is a promising method. It takes the knowledge matching task closer toward automation.

## 7. References


1. Wang, J.T.L., Shasha, D., Chang, G., Relihan, L., Zhang, K. and Patel, G.: Structural matching and discovery in document databases. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, pp. 560-563, (1997).
2. Wang, K. and Liu, H.: Discovering Structural Association of Semistructured Data. In: IEEE Transactions on Knowledge and Data Engineering, (1999).
3. Wang, J. T. L., Shapiro, B.A., Shasha, D., Zhang, K. and Currey, K.M.: An algorithm for finding the largest approximately common substructures of two trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 889–895, (1998).
4. Chen, Y., Shi, Y. and Chen, Y.: Tree inclusion algorithm, signatures and evaluation of path-oriented queries. In: ACM symposium on Applied computing, Dijon, France, pp. 1020 – 1025, (2006).
5. Zhang, Y., Street, W.N. and Burer, S.: Sharing Classifiers among Ensembles from Related Problem Domains. Proc. of the 5th IEEE Int'l Conf. on Data Mining, 2005.
6. Prodromidis, A., Chan, P. and Stolfo, S.: Metalearning in distributed data mining systems: Issues and approaches. In: KDD'07, (1997).
7. Tan, H., Dillon, T.S., Hadzic, F., Feng, L. and Chang, E.: IMB3-Miner: Mining Induced/Embedded subtrees by constraining the level of embedding. In: PAKDD'06, Singapore, (2006).
8. Hadzic, F., Tan, H., Dillon, T.S. and Chang, E.: U3 – mining unordered embedded subtrees using TMG candidate generation. Submitted to the 1st ACM Int'l Conf. on Web Search and Data Mining, San Francisco Bay Area, California, USA, (2008).
9. Hadzic, F., Tan, H. and Dillon, T.S.: UNI3 – Efficient Algorithm for Mining Unordered Induced Subtrees Using TMG Candidate Generation. IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007), Honolulu, Hawaii, (2007).
10. Hadzic, F., Tan, H., Dillon, T.S. and Chang, E.: Implications of frequent subtree mining using hybrid support definition. Data Mining & Information Engineering, 18-20 June, The New Forest, UK, (2007).
11. Tan, H., Dillon, T.S., Hadzic, F., Chang, E. and Feng, L.: MB3-Miner: mining eMBedded sub-TREEs using Tree Model Guided candidate generation. In: Proc. of the 1st Int'l Workshop on Mining Complex Data, in conj. with ICDM'05, Houston, Texas, USA, (2005).
12. Tan, H., Dillon, T.S., Hadzic, F. and Chang, E.: Razor: mining distance constrained embedded subtrees. IEEE ICDM 2006 Workshop on Ontology Mining and Knowledge Discovery from Semistructured documents, 28-22 December, Hong Kong, (2006).
13. Hadzic, F., Dillon, T.S., Sidhu, A., Chang, E. and Tan, H.: Mining Substructures in Protein Data. IEEE ICDM 2006 Workshop on Data Mining in Bioinformatics (DMB 2006), 18-22 December, Hong Kong, (2006).
14. Blake, C., Keogh, E. and Merz, C.J.: UCI Repository of Machine Learning Databases. Irvine, CA: University of California, Department of Information and Computer Science, (1998) [http://www.ics.uci.edu/~mlearn/MLRepository.html].
15. Hadzic, F. and Dillon, T.S.: Using the Symmetrical Tau (τ) Criterion for Feature Selection in Decision Tree and Neural Network Learning. In: Proc. of the 2nd FSDM Workshop, in conj. with 2006 SIAM Int'l Conf. on Data Mining, Bethesda, (2006).