(12) INNOVATION PATENT (11) Application No. **AU 2021107061 A4**
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Suffix Forest: A novel in-memory data structure for analyzing time-series data

(51) International Patent Classification(s)
G06N 5/00 (2006.01) *G06N 3/08* (2006.01)
G06F 17/10 (2006.01) *G06N 20/20* (2019.01)

(21) Application No: **2021107061** (22) Date of Filing: **2021.08.24**

(45) Publication Date: **2021.12.02**
(45) Publication Journal Date: **2021.12.02**
(45) Granted Journal Date: **2021.12.02**

(71) Applicant(s)
Kartick Chandra Mondal;Somnath Mukhopadhyay;Sunita Sarkar;Samiran Chattopadhyay;Moumita Ghosh;Anindita Sarkar Mondal;Rohmatul Fajriyah

(72) Inventor(s)
Mondal, Kartick Chandra;Mukhopadhyay, Somnath;Sarkar, Sunita;Chattopadhyay, Samiran;Ghosh, Moumita;Mondal, Anindita Sarkar;Fajriyah, Rohmatul

(74) Agent / Attorney
Dr. Kartick Chandra Mondal, 2/454 Scarborough Beach Road Suite 709, Osborne Park, WA, 6017, AU

2021107061 24 Aug 2021

ABSTRACT

The present invention generally relates to a method for introducing suffix forest for mining tri-clusters from time series data comprises scanning all rows one by one and creating suffixes by deleting one front item in each rotation and appended in the tree to build a suffix-tree from SFD; showing data that satisfies each element in a particular branch by leaf node thereby gives clustered results; performing a cumulative addition of new suffixes over the suffix-tree built from the first SFD to merge multiple SFDs, wherein cumulative addition follows the match and buildEdge function to generate the Suffix-forest; and removing redundancy by pruning and merging the dataset of TM thereby extracting tri-clusters from leaf of the data structure.

202111070061 24 Aug 2021

Suffix Forest: A novel in-memory data structure for analyzing time-series data

FIELD OF THE INVENTION

The present disclosure relates to a method for introducing suffix forest for mining tri-clusters from time series data.

BACKGROUND OF THE INVENTION

Three-dimensional data, also known as triadic data, is a collection of multiple two-dimensional data matrices that are useful for representing and understanding the relationship between data in a variety of domains, including business analysis (product category-sales measure-time), medical data analysis (patient-record-time), bioinformatics (gene-sample-time), biodiversity (species-location-time), and so on.

Time-series data is a collection of observed data over a long period of time that can be used to answer analytics questions such as the progress or deterioration of a product's sale over time, patient response over time for a specific drug, gene expression changing under certain conditions over time, or changes in species presence records in different locations over time, and so on. As a result, modelling an unsupervised learning technique on these types of data in order to attribute the underlying structure or distribution of data is a difficult issue that is still being researched.

In the view of the forgoing discussion, it is clearly portrayed that there is a need to have a method for introducing suffix forest for mining tri-clusters from time series data.

SUMMARY OF THE INVENTION

The present disclosure seeks to provide a method to implement the tri-clustering concept with an informative structure where changes in forest cover, mangrove cover along time are monitored in different states and union territories.

In an embodiment, a method for introducing suffix forest for mining tri-clusters from time series data is disclosed. The method includes scanning all rows one by one and creating suffixes by deleting one front item in each rotation and appended in the tree to build a suffix-tree from SFD. The method further includes showing data that satisfies each element in a particular branch by leaf node thereby gives clustered results. The method further includes performing a cumulative addition of new suffixes over the suffix-tree built from the first SFD to merge multiple SFDs, wherein cumulative addition follows the match and buildEdge function to generate the Suffix-forest. The method further includes removing redundancy by pruning and merging the dataset of TM thereby extracting tri-clusters from leaf of the data structure.

In an embodiment, SFD dataset contains multiple rows in ascending order according to their frequency of occurrence.

In an embodiment, for a single itemset, corresponds to a particular row, the number of branches created, or merged, or updated equals the number of items of that itemset each branch represents a set of numbers in ascending order.

In an embodiment, if suffix-tree generated for MDM is considered, the internal HNodes are for the status of year-wise forest presence and leaf HNode are for the states having similar forest statistics, which is treated as result of bi-clusters.

In an embodiment, suffix-Forest data structure is consisting of HTree and HNode, wherein HTree holds all the unique items present in the SFD in separate cells having a copy in the very first HNode pointed by the HTree cells.

In an embodiment, suffix forest generation is user driven which allows merging of datasets based on the dataset requirement as per user query.

An object of the present disclosure is to introduce a data structure named suffix-forest and design a novel tri-clustering approach.

Another object of the present disclosure is to implement the tri-clustering concept with an informative structure where changes in forest cover, mangrove cover along time are monitored in different states and union territories.

Yet another object of the present invention is to deliver an expeditious and cost-effective method for introducing suffix forest for mining tri-clusters from time series data.

To further clarify advantages and features of the present disclosure, a more particular description of the invention will be rendered by reference to specific embodiments thereof, which is illustrated in the appended drawings. It is appreciated that these drawings depict only typical embodiments of the invention and are therefore not to be considered limiting of its scope. The invention will be described and explained with additional specificity and detail with the accompanying drawings.

BRIEF DESCRIPTION OF FIGURES

These and other features, aspects, and advantages of the present disclosure will become better understood when the following detailed description is read with reference to the accompanying drawings in which like characters represent like parts throughout the drawings, wherein:

Figure 1 illustrates a flow chart of a method for introducing suffix forest for mining tri-clusters from time series data in accordance with an embodiment of the present disclosure;

Figure 2 illustrates structure of generalized suffix-tree for MDM in accordance with an embodiment of the present disclosure;

Figure 3 illustrates suffix forest by merging MDM and OM in accordance with an embodiment of the present disclosure;

Figure 4 illustrates updated suffix forest in accordance with an embodiment of the present disclosure;

Figure 5 illustrates pruned suffix forest in accordance with an embodiment of the present disclosure; and

Figure 6 illustrates pruned suffix forest of MDM, OM, TM in accordance with an embodiment of the present disclosure.

Further, skilled artisans will appreciate that elements in the drawings are illustrated for simplicity and may not have necessarily been drawn to scale. For example, the flow charts illustrate the method in terms of the most prominent steps involved to help to improve understanding of aspects of the present disclosure. Furthermore, in terms of the construction of the device, one or more components of the device may have been represented in the drawings by conventional symbols, and the drawings may show only those specific details that are pertinent to understanding the embodiments of the present disclosure so as not to obscure the drawings with details that will be readily apparent to those of ordinary skill in the art having benefit of the description herein.

DETAILED DESCRIPTION:

For the purpose of promoting an understanding of the principles of the invention, reference will now be made to the embodiment illustrated in the drawings and specific language will be used to describe the same. It will nevertheless be understood that no limitation of the scope of the invention is thereby intended, such alterations and further modifications in the illustrated system, and such further applications of the principles of the invention as illustrated therein being contemplated as would normally occur to one skilled in the art to which the invention relates.

It will be understood by those skilled in the art that the foregoing general description and the following detailed description are exemplary and explanatory of the invention and are not intended to be restrictive thereof.

Reference throughout this specification to "an aspect", "another aspect" or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present disclosure. Thus, appearances of the phrase "in an embodiment", "in another embodiment" and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

The terms "comprises", "comprising", or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a process or method that comprises a list of steps does not include only those steps but may include other steps not expressly listed or inherent to such process or method. Similarly, one or more devices or sub-systems or elements or structures or components proceeded by "comprises...a" does not, without more constraints, preclude the existence of other devices or other sub-systems or other elements or other structures or other components or additional devices or additional sub-systems or additional elements or additional structures or additional components.

The present invention method 100 as in Figure 1 is capable of being implemented by computer programs or modules can be executed in many exemplary ways, such as an application that is resident in the memory of a device or as a hosted application that is being executed on a server and communicating with the device application or browser via a number of standard protocols, such as TCP/IP, HTTP, XML, SOAP, REST, JSON and other sufficient protocols. The method is configured to be implemented programmable hardware devices such as processors, digital signal processors, central processing units, field programmable gate arrays, programmable array logic, programmable logic devices, cloud processing systems, or the like.

Unless otherwise defined, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this invention belongs. The system, methods, and examples provided herein are illustrative only and not intended to be limiting.

Embodiments of the present disclosure will be described below in detail with reference to the accompanying drawings.

Referring to **Figure 1**, a flow chart of a method for introducing suffix forest for mining tri-clusters from time series data is illustrated in accordance with an embodiment of the present disclosure. At step 102, the method 100 includes scanning all rows one by one and creating suffixes by deleting one front item in each rotation and appended in the tree to build a suffix-tree from SFD.

At step 104, the method 100 includes showing data that satisfies each element in a particular branch by leaf node thereby gives clustered results.

At step 106, the method 100 includes performing a cumulative addition of new suffixes over the suffix-tree built from the first SFD to merge multiple SFDs, wherein cumulative addition follows the match and buildEdge function to generate the Suffix-forest.

At step 108, the method 100 includes removing redundancy by pruning and merging the dataset of TM thereby extracting tri-clusters from leaf of the data structure.

In an embodiment, SFD dataset contains multiple rows in ascending order according to their frequency of occurrence.

In an embodiment, for a single itemset, corresponds to a particular row, the number of branches created, or merged, or updated equals the number of items of that itemset each branch represents a set of numbers in ascending order.

In an embodiment, if suffix-tree generated for MDM is considered, the internal HNodes are for the status of year-wise forest presence and leaf HNode are for the states having similar forest statistics, which is treated as result of bi-clusters.

In an embodiment, suffix-Forest data structure is consisting of HTree and HNode, wherein HTree holds all the unique items present in the SFD in separate cells having a copy in the very first HNode pointed by the HTree cells.

In an embodiment, suffix forest generation is user driven which allows merging of datasets based on the dataset requirement as per user query.

Suffix-Forest Building: Suffix-Forest data structure is consisting of HTree and HNode. HTree holds all the unique items present in the SFD in separate cells. All of these items have a copy in the very first HNode pointed by the HTree cells. Following the itemset, HNodes are created successively. HTree and HNode (step 2) is initialized. If the total number of items is x, count for each row, genSuffix() is called x times (step 3-8) and returns all possible suffixes for that row. Next, the tree is built (step 9 - 16) either by updating (step 10-11) the present branch or by creating a new branch (step 13 - 16). For updating, the technique finds whether an item is already present in HTree or not. If it finds that the item is already present, then subsequent HNodes are checked for finding similarity. This procedure is explained by match()(step 11). The match() takes the current HNode and the suffix. Otherwise, new data item is inserted in HTree and a branch is constructed in between HTree and HNode. The upcoming HNodes are connected by buildEdge()(step 16) where two parameters are passing, one for the new HNode to be created and the second one is for the remaining suffix after deleting the front.

genSuffix(): It describes how all the suffixes are generated for a particular row of SFD. Itemset-list denoting a row in SFD and an object is passed through this function. For loop in step 3 corresponds to number of suffixes to be generated which is equal to the length of the list (calculated in step 2). Second for loop (in step 5) generates m^{th} suffix starting from mth location by appending successive terms up to last. Step 9 adds the object. Thus all the suffixes are gathered in suffix variable and retuned at the end.

match(): Use of match() is for the suffixes containing initially similar items in the itemsets. During insertion of the second itemset, this function is used for checking the next unmatched item and according to that new branch in created using buildEdge(). Once a matched node is found, it deletes the first item from the suffix (step 2). Then recursively (step 3-6) checks subsequent nodes with the first item of the suffix until an unmatched node is found.

When an unmatched node, buildEdge() is called to create a sub-branch from the current HNode and remaining suffix (step 8). buildEdge(): This function is called twice. First time it is called (step 16) to create a new branch. Sub branches are generated after finding first unmatched items from two different suffixes. At the point of generating sub branch, second time buildEdge() is called from Function 4 (step 8).

HNode and suffix list are passed as argument through this function. Variable len stores the length of the list, which includes itemsets along a tree branch and terminator object as well. For the last item or branch with a single item to be processed, the function follows step 9 to 11 where data item and object lists both are inserted into the HNode. For all other HNodes buildEdge() is called recursively (step 7) where front item from the list is inserted, an edge is created from the current to the child node and inserted data is deleted from the list (step 4 - 6).

Execution of the disclosed method on the selected data comprises:

Construction of frequent generalized suffix-tree

Suffix-tree generated from MDM is depicted in Figure 2. After merging the suffix-tree of MDM with the suffixes generated from OM, observed in Figure 3. After that, updating is done. The updated suffix-forest is shown in Figure 4. Next, pruning is done to remove the redundancy. The derived forest is shown in Figure 5. Now, merging the dataset of TM, followed by update and prune, generate the suffix forest that is depicted in Figure 6.

Thus, the resulted suffix forest will continue to grow as more datasets will be added. This suffix forest generation can be user driven. That is merging of datasets can be processed based on the dataset requirement as per user query. From the pruned suffix-forest obtained in Figure 6, the tri-clusters are extracted from the leaf of the data structure.

The drawings and the forgoing description give examples of embodiments. Those skilled in the art will appreciate that one or more of the described elements may well be combined into a single functional element. Alternatively, certain elements may be split into multiple functional elements. Elements from one embodiment may be added to another embodiment. For example, orders of processes described herein may be changed and are not limited to the manner described herein. Moreover, the actions of any flow diagram need not be implemented in the order shown; nor do all of the acts necessarily need to be performed. Also, those acts that are not dependent on other acts may be performed in parallel with the other acts. The scope of embodiments is by no means limited by these specific examples. Numerous variations, whether explicitly given in the specification or not, such as differences in structure, dimension, and use of material, are possible. The scope of embodiments is at least as broad as given by the following claims.

Benefits, other advantages, and solutions to problems have been described above with regard to specific embodiments. However, the benefits, advantages, solutions to problems, and any component(s) that may cause any benefit, advantage, or solution to occur or become more pronounced are not to be construed as a critical, required, or essential feature or component of any or all the claims.

WE CLAIM:

1. A method for introducing suffix forest for mining tri-clusters from time series data, the method comprises:

scanning all rows one by one and creating suffixes by deleting one front item in each rotation and appended in the tree to build a suffix-tree from SFD;

showing data that satisfies each element in a particular branch by leaf node thereby gives clustered results;

performing a cumulative addition of new suffixes over the suffix-tree built from the first SFD to merge multiple SFDs, wherein cumulative addition follows the match and buildEdge function to generate the Suffix-forest; and

removing redundancy by pruning and merging the dataset of TM thereby extracting tri-clusters from leaf of the data structure.

2. The method as claimed in claim 1, wherein SFD dataset contains multiple rows in ascending order according to their frequency of occurrence.

3. The method as claimed in claim 1, wherein for a single itemset, corresponds to a particular row, the number of branches created, or merged, or updated equals the number of items of that itemset each branch represents a set of numbers in ascending order.

4. The method as claimed in claim 1, wherein if suffix-tree generated for MDM is considered, the internal HNodes are for the status of year-wise forest presence and leaf HNode are for the states having similar forest statistics, which is treated as result of bi-clusters.

5. The method as claimed in claim 4, wherein suffix-Forest data structure is consisting of HTree and HNode, wherein HTree holds all the unique items present in the SFD in separate cells having a copy in the very first HNode pointed by the HTree cells.

6. The method as claimed in claim 1, wherein suffix forest generation is user driven which allows merging of datasets based on the dataset requirement as per user query.

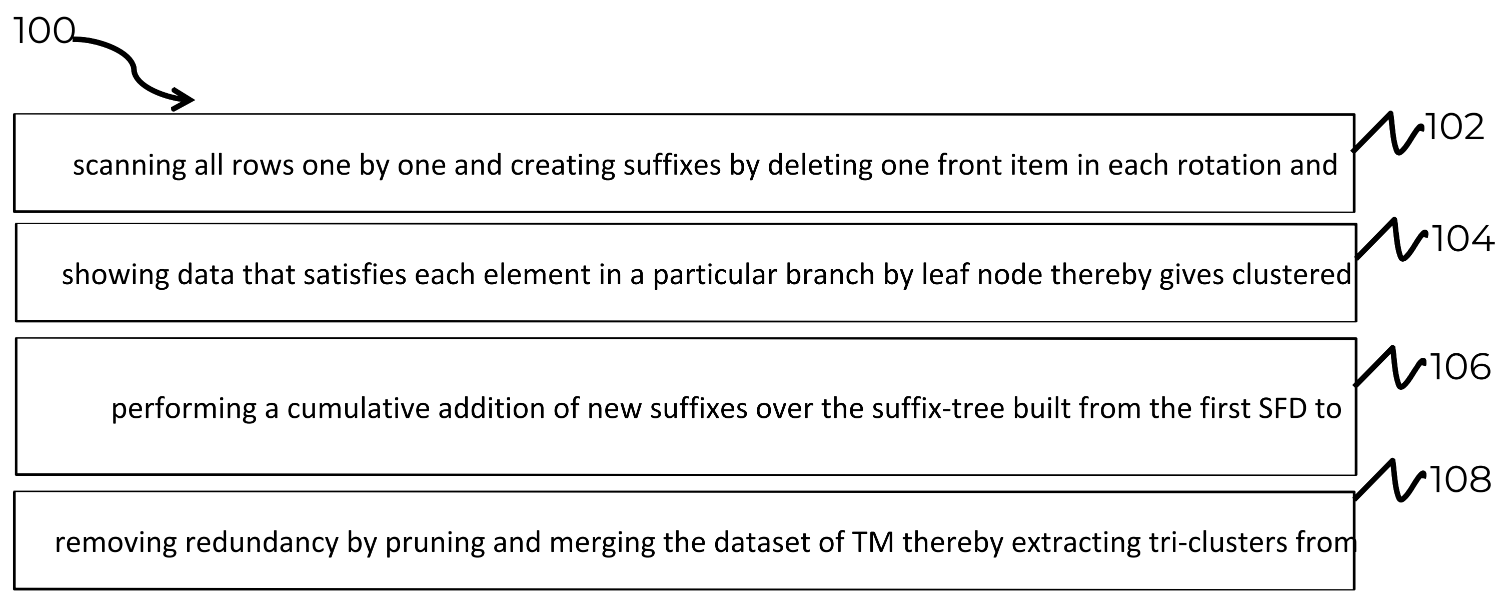
100
scanning all rows one by one and creating suffixes by deleting one front item in each rotation and
102
showing data that satisfies each element in a particular branch by leaf node thereby gives clustered
104
performing a cumulative addition of new suffixes over the suffix-tree built from the first SFD to
106
removing redundancy by pruning and merging the dataset of TM thereby extracting tri-clusters from
108

Figure 1

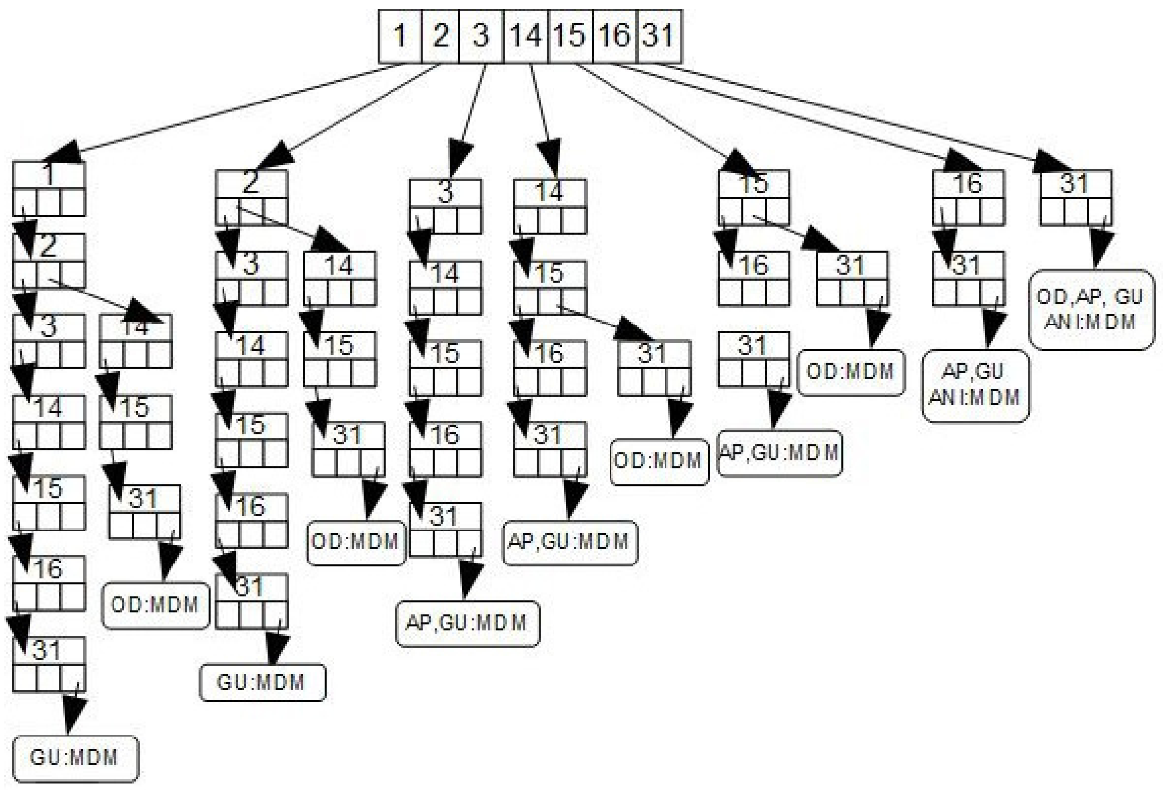
1 2 3 14 15 16 31
1
2
3
14
15
16
31
GU:MDM
14
15
31
OD:MDM
2
3
14
15
16
31
GU:MDM
14
15
31
OD:MDM
3
14
15
16
31
AP,GU:MDM
14
15
16
31
AP,GU:MDM
31
OD:MDM
15
16
31
AP,GU:MDM
31
OD:MDM
16
31
AP,GU AN I:MDM
31
OD,AP, GU AN I:MDM

Figure 2

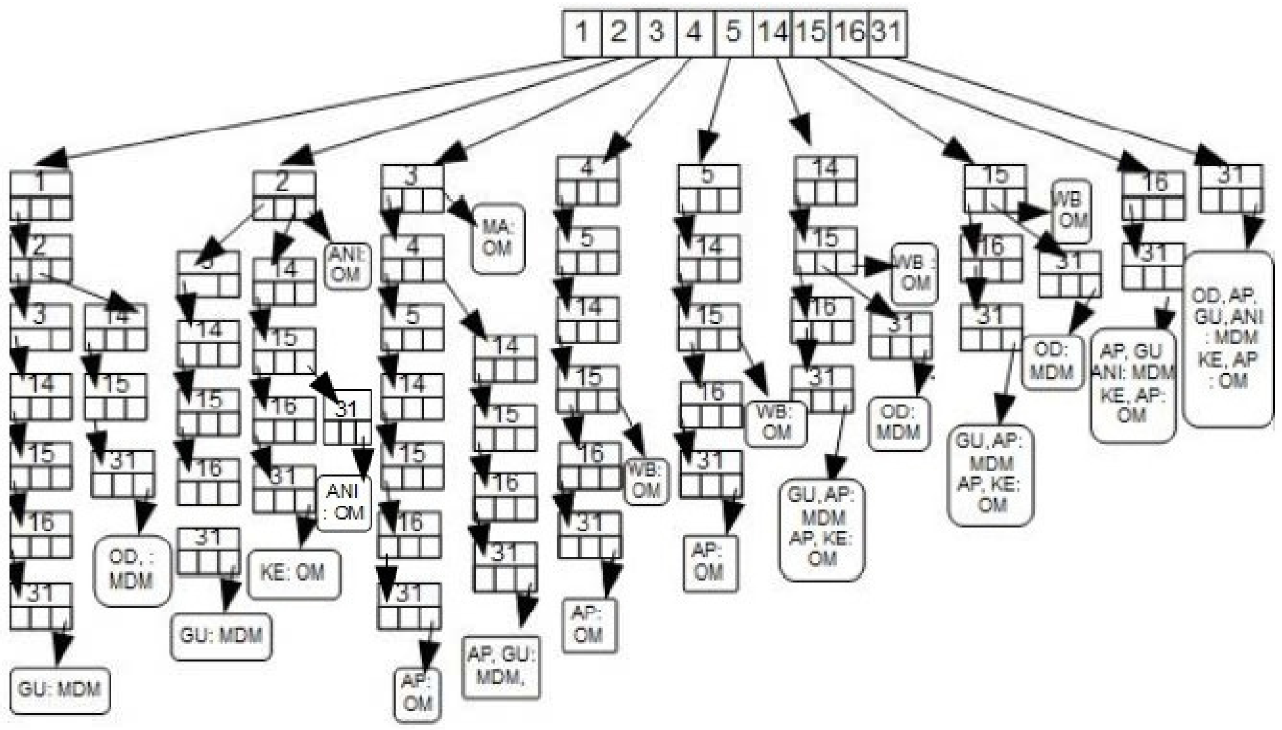
1 2 3 4 5 14 15 16 31
GU: MDM
OD, : MDM
GU: MDM
KE: OM
ANI: OM
ANI : OM
MA: OM
AP: OM
AP, GU: MDM,
AP: OM
WB: OM
AP: OM
WB: OM
WB : OM
GU, AP: MDM AP, KE: OM
OD: MDM
WB OM
GU, AP: MDM AP, KE: OM
OD: MDM
AP, GU ANI: MDM KE, AP: OM
OD, AP, GU, ANI : MDM KE, AP : OM

Figure 3

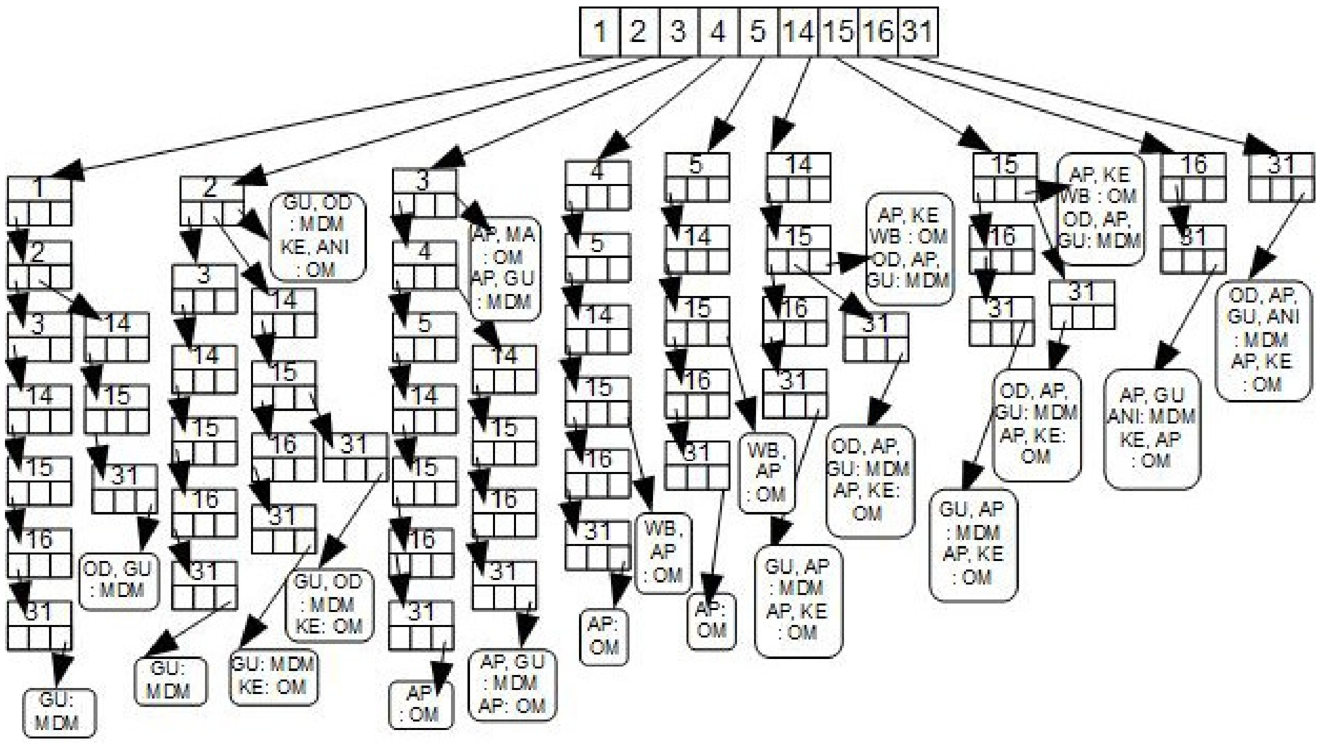
1 2 3 4 5 14 15 16 31
GU: MDM
OD, GU : MDM
GU: MDM
GU: MDM KE: OM
GU, OD : MDM KE: OM
GU, OD : MDM KE, ANI : OM
AP : OM
AP, GU : MDM AP: OM
AP, MA : OM AP, GU : MDM
AP: OM
WB, AP : OM
AP: OM
WB, AP : OM
GU, AP : MDM AP, KE : OM
OD, AP, GU: MDM AP, KE: OM
AP, KE WB : OM OD, AP, GU: MDM
GU, AP : MDM AP, KE : OM
OD, AP, GU: MDM AP, KE: OM
AP, KE WB : OM OD, AP, GU: MDM
AP, GU ANI: MDM KE, AP : OM
OD, AP, GU, ANI : MDM AP, KE : OM

Figure 4

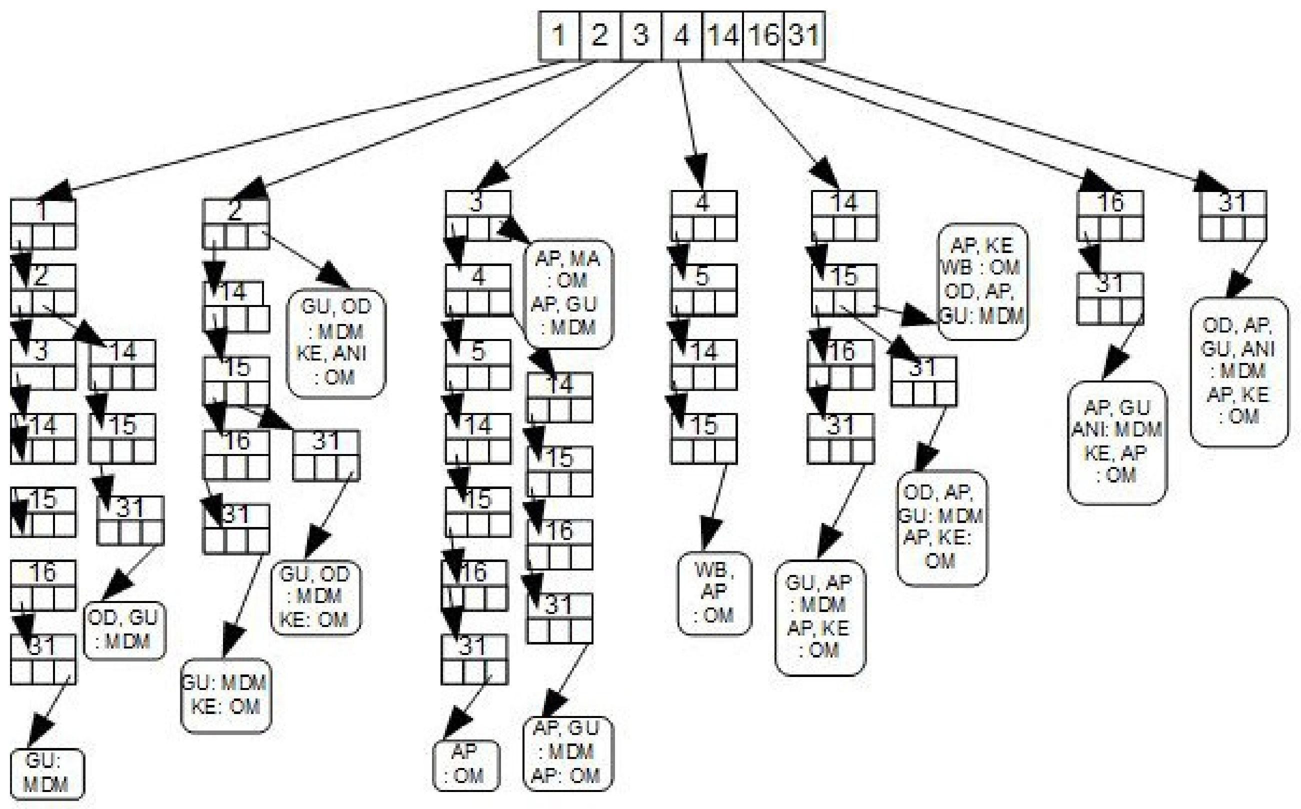
1 2 3 4 14 16 31
GU: MDM
OD, GU : MDM
GU: MDM KE: OM
GU, OD : MDM KE: OM
GU, OD : MDM KE, ANI : OM
AP : OM
AP, MA : OM AP, GU : MDM
AP, GU : MDM AP: OM
WB, AP : OM
GU, AP : MDM AP, KE : OM
AP, KE WB : OM OD, AP, GU: MDM
OD, AP, GU: MDM AP, KE: OM
AP, GU ANI: MDM KE, AP : OM
OD, AP, GU, ANI : MDM AP, KE : OM

Figure 5

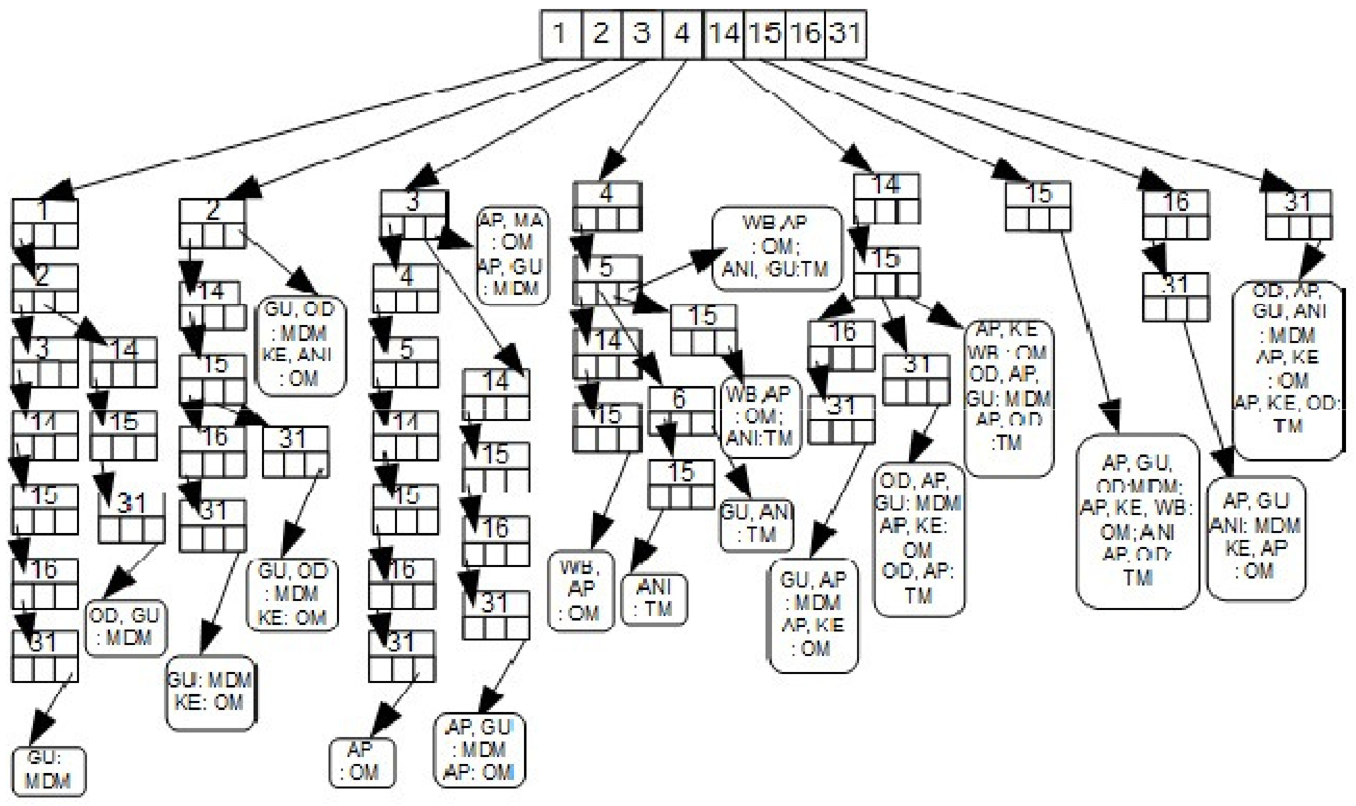
1 2 3 4 14 15 16 31
GU: MDM
OD, GU : MDM
GU, OD : MDM KE, ANI : OM
GU, OD : MDM KE: OM
GU: MDM KE: OM
AP, MA : OM AP, GU : MDM
AP : OM
AP, GU : MDM AP: OM
WB,AP : OM; ANI, GU:TM
WB, AP : OM
ANI : TM
WB,AP : OM; ANI:TM
GU, ANI : TM
GU, AP : MDM AP, KE : OM
OD, AP, GU: MDM AP, KE: OM OD, AP: TM
AP, KE WB : OM OD, AP, GU: MDM AP, OD :TM
AP, GU, OD:MDM; AP, KE, WB: OM;ANI AP, OD TM
AP, GU ANI: MDM KE, AP : OM
OD, AP, GU, ANI : MDM AP, KE : OM AP, KE, OD: TM

Figure 6