VOL. 10, NO. 6, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ANALYSIS OF DENSE AND SPARSE PATTERNS TO IMPROVE MINING EFFICIENCY

A. Veeramuthu
Department of Information Technology, Sathyabama University, Chennai India
E-Mail: aveeramuthu@gmail.com

ABSTRACT

Generally, data mining concept is used to gather information from various data repository. Frequent pattern mining is to be designed for displaying repetitions in the transactional database. Patterns are defined by predefined format. In this evolutionary work, the proposed concept to mine the transactional database using the combination of recommendations and prediction by the help of software simulation. In hardware side, this process is explained in pattern mining using systolic tree creation. This will handle pattern mining to configure the frequent pattern while generating systolic tree structure. But it can handle certain size of dataset only, but also generate more candidate item set when implementing the item set matching by tree projection algorithm. This will occupy more and more memory, each time reevaluation done from the scratch of dataset in hardware side. It is required more time to process. To overcome this problem, in software side to implement HI-Growth tree technique to analyze large scale of dataset. The new concept is introduced based on recommendation approach to avoid candidate set generation. This method is achieved to reduce the internal memory and mining time is by dividing the frequent pattern into the dense and the sparse patterns. In this paper, investigate the mining speed of the HI-Growth tree is fast-paced than original software side algorithm of FP-Growth tree, and also it consumes less amount of memory for analyzing dense and sparse pattern through recommendation technique to improve the mining efficiency, while achieving the higher throughput to overcome the defect of hardware approach.

Keywords: frequent pattern mining, systolic tree, recommendation, prediction, dense pattern, sparse pattern.

INTRODUCTION

Main goal of this paper is to achieve the mining efficiency based on HI-Growth Tree approach. It also analyzes the patterns to improve the speed of frequent mining. Achieving the efficiency based on recommendations and predictions. These are mentioned by user's requirements. This approach can consume less capacity of memory space at the time of analyzing the patterns. Frequent pattern mining is designed for displaying commonly occurring items in transactional database. Here suggest a reconfigurable architecture for frequent pattern mining based HI-GROWTH TREE structure. The purpose of this architecture (shown in Figure 2) is to reduce the mining time and to improve mining efficiency. In this, frequent patterns are divided into dense and sparse patterns.

In Hardware approach, it can achieve this by Hybrid Systolic Tree method. This method is going to show that the mining speed of the HI-Growth Tree is number of times faster for any long frequent patterns. To improve mining efficiency, need to use some possible combinations (recommendations), to analyze the frequent patterns in transactional databases.

In this approach, it uses parallel FPGA implementation to analyze the frequent patterns. Insist of using database Projection in FPGA [1, 4, 9] it can Apply HAPPI algorithm [5, 6] to keep only index values for reducing the memory space to store candidate item set, through this it will effectively mimic the internal memory layout for the frequency of loading database into the hardware.

This paper tells about pattern mining using recommendation to analyze the frequent patterns. It can achieve the mining efficiency through predicting the patterns via users behavioral, like, in on online shopping, it helps to recommend the products which will expect by the customer, and this paper also dealt with security mechanisms.

The remaining content of this article is arranged as follows: In section 2 mentioned related works in the field of data mining applications with the existing model. The problem description of proposed model is given in Section 3. In this section, the definition and creation of the HI -Growth tree structure are presented. Section 4 discussed the simulations of pattern mining over HI-Growth tree algorithm flow steps. In Section 5, shown the experimental results. In Section 6, evaluate the performance of the work. Finally, in Section 7, the conclusion of the paper is included.

RELATED WORK

Existing system

The design of Systolic tree creation in hardware approach to define the transactional database has the similar structure in FP tree given same transactional datasets. Pattern mining is used to define item set matching to analyze the frequent patterns for entire systolic tree.

Existing system uses binary search mechanism for generating recommendations from the entire transaction. As the transaction goes high, the existing uses caching mechanism for storing the frequent transactions and every time as the transaction changes it re-evaluates the cache or in-validates the cache.

In Figure-1, shown the hardware mechanism [3] [10], is using pattern mining with data projection to implement item set matching. This will generate more and more candidate item set to occupy more memory space.

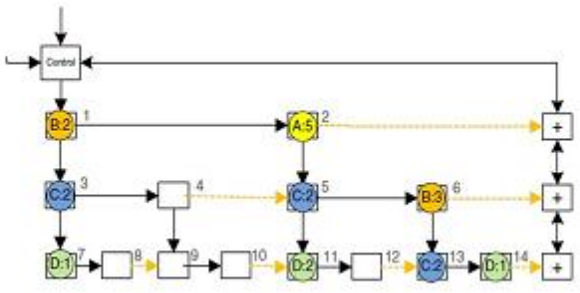


Figure-1. Systolic Tree Architecture.

Problem definition

- Reading, sorting and collecting frequency count, it takes long time to scan database.
- Creating a Linked-List system to represent FP-Tree.
- Handling large size of transactions as challenging task by tree projection algorithm [8].
- Generating more item set to occupy more memory.

Limitations of an existing system

- Exhaustive search is not feasible for typical modern database
- Implementation of recursive processing directly in hardware is difficult.
- A threads increasing, it leads to take more time to analyze the frequent patterns.
- The systolic-tree approach depends on the size of transaction, as the transaction goes high the approach takes time in generating frequent patterns.
- Recommendations every time retrieved from FP-Tree [2] which takes sample time if the FP-Tree records cross certain limit.

Proposed system

Frequent pattern/sub-pattern mining plays a vital role in mining process of associations, max patterns, and multi dimensional patterns and other approaches in the data mining task.

In this study, proposed the Hybrid Intelligence mining approach, to handle large scale of frequent pattern, which is an advanced recommendation based tree data structure for storing highly compressed and critical contents about frequent patterns, and develop an HI-Growth tree based pattern mining method by using pattern recommendation approach, then it can configure complete set of frequent patterns. In this way it can avoid unnecessary candidate item set generations.
Efficiency of mining is attained by following techniques:

- A large database can tune into highly condensed and compressed format. This avoids repetitive database scans.
- The HI-Growth tree based pattern mining approach, adopts a pattern Recommendation fragment Growth techniques, to follow and avoid large number of candidate item sets generations.
- It can analyze the frequent patterns based on their threshold values to avoid wasting memory space to search more and more times to get result accuracy.
- It helps to split up as knowledgeable recommendations, and noisy recommendations.
- E.g. using regime identification techniques, it can predict the probability value of recommending item set.

In this hardware approach it can suggest pattern mining along with HAPPI algorithm to avoid unnecessary generation of candidate item set. Instead of giving attributes it can take only the index term as an item set.

The performance analysis shows that HI-Growth Pattern is best in consuming less memory space, avoiding costly database scans, and efficient in configuring large scale of transactional databases. Here, the size of the data bases not an issue.

System architecture

In Figure-2 shown the architecture, tells about how the transactional databases was implementing by HI-Growth Tree. It also handles the datasets to identifying the frequent patterns. Take the frequent pattern to analyze and giving recommendations based on it. Here, the efficiency was achieved through handling of frequent patterns. It can consume less memory to improve storage space of system capability.

HI-GROWTH TREE DESIGN CREATION

HI-Growth tree approach has some special methods:

- Identification of Frequent patterns.
- HI-Growth Tree Creation.
- Recommendations based Frequent Pattern implementation.

Identification of frequent patterns

This module is used to find frequent items by selecting a Dataset containing list of Transactions (see Table-1). It is used to calculate the minimum support count using minimum support value and transactions. Every frequent Item set should satisfy the min_support threshold value. Once it got the minimum support value this module counts frequency of occurrence for each and every item in the transaction list.

$$\text{Support} = \frac{(\text{X U Y}).\text{count}}{\text{N (No. of Transactions in dataset)}}$$

Min_support_frequency = Sup_frequency x N

VOL. 10, NO. 6, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

For generating Frequent Patterns, it travels from bottom to top of the HI-Growth Tree. After selecting a node it helps to find the possible transactions for the selected node, then for each and every node of transaction must support the minimum support count. If the path or transaction supports the minimum support counts then it need to be added as Frequent Pattern for the Transaction.

Then, the tree representation get created, using HI-Growth Tree, it takes *the Root Node as (Null).* However, FP algorithms have bottlenecks because they are not able to scan large size database.

HI-Growth tree creation

By frequent patterns, it re-shuffles the transactions example listed below in the Table-1.

Table-1. Sample transactional database.

T_ID	Transactions
1	B, D, A, E
2	B, D, A, E
3	B, A, E, C
4	B, D, A
5	D
6	B, D
7	D, A, E
8	B, C

Implementation of frequent patterns

In order to show the performance of the FP Matching algorithm, it needs the implementation of HI-Growth tree approach where Frequent-Patterns can increase the performance of the Application e.g.; online shopping. In this application, frequent patterns can be referred through possible combinations based on user requirements, to achieve the mining efficiency. In this module, it has implemented three possible combinations of Pattern matching for the users.

- Knowledgeable recommendation.
- Noisy recommendation.
- General recommendation

Property 1: Knowledgeable recommendation

Only recommend the most relevant items in the frequent Pattern. Most relevant items are called dense pattern.

$$\text{Dense} = \text{Compactness, Thick}$$

Property 2: Noisy recommendation

Based on threshold value it may recommend items in frequent pattern. This irrelevant pattern may call sparse pattern.

$$\text{Sparse} = \text{Scattered, Thin}$$

Property 3: General recommendation

More combined frequent items based on pattern generation, it will explained by Customer-Customer (Category wise recommendation).

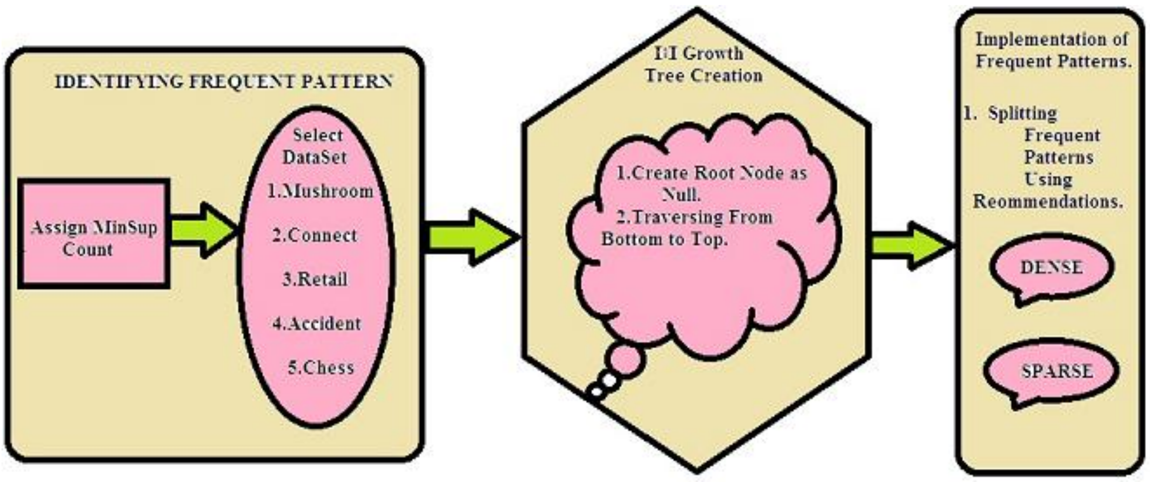

IDENTIFYING FREQUENT PATTERN
Assign MinSup Count
Select DataSet
1.Mushroom
2.Connect
3.Retail
4.Accident
5.Chess
HI Growth Tree Creation
1.Create Root Node as Null.
2.Traversing From Bottom to Top.
Implementation of Frequent Patterns.
1. Splitting Frequent Patterns Using Reommendations.
DENSE
SPARSE

Figure-2. Architecture.

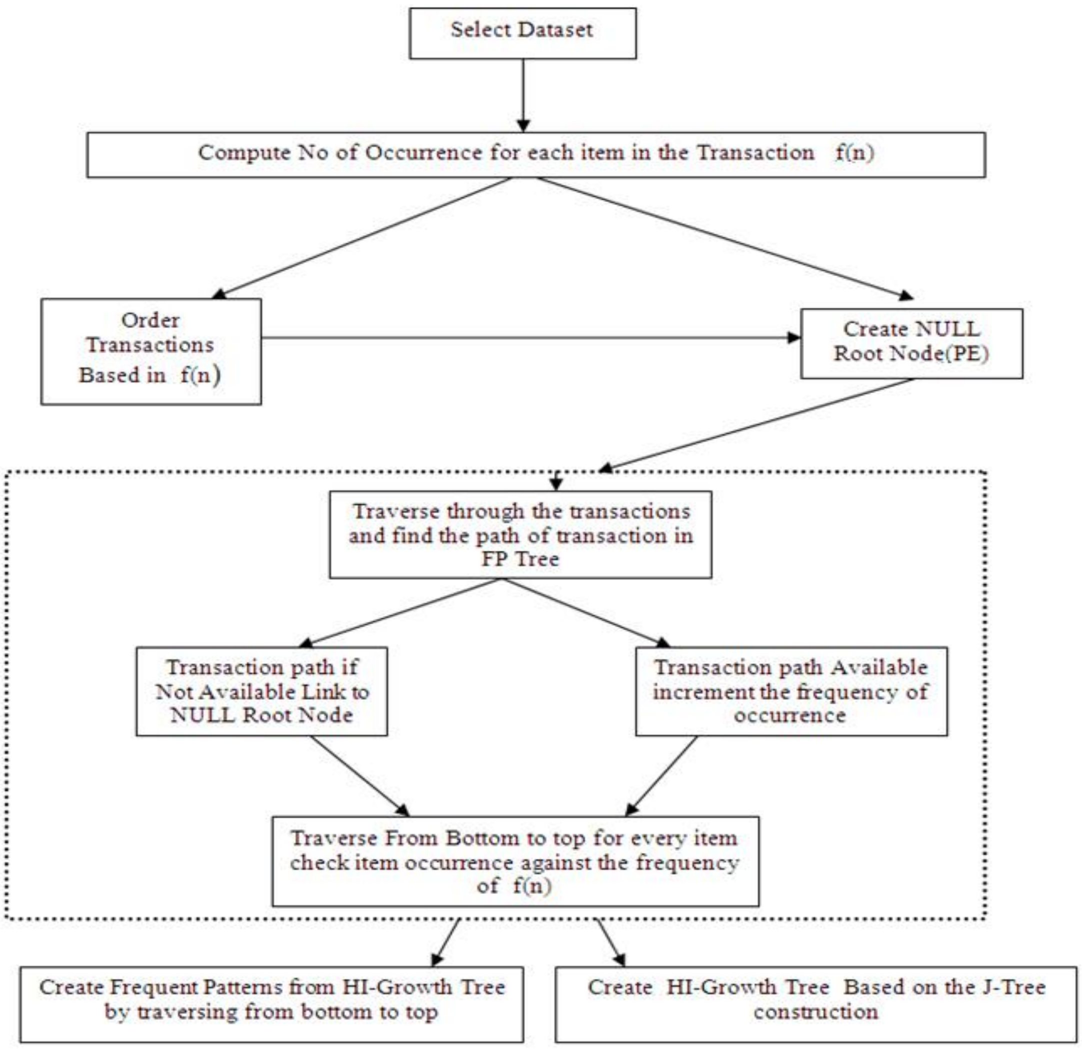

Select Dataset
Compute No of Occurrence for each item in the Transaction f(n)
Order Transactions Based in f(n)
Create NULL Root Node(PE)
Traverse through the transactions and find the path of transaction in FP Tree
Transaction path if Not Available Link to NULL Root Node
Transaction path Available increment the frequency of occurrence
Traverse From Bottom to top for every item check item occurrence against the frequency of f(n)
Create Frequent Patterns from HI-Growth Tree by traversing from bottom to top
Create HI-Growth Tree Based on the J-Tree construction

Figure-3. HI-Growth algorithm flow diagram.

Threads increases avoided by category recommendation. (Sup-Count increase).

HI-Growth algorithm complete flow

Step-1: Choose the dataset from the given application. Process it for getting minimum frequency and sup_count.

Step-2: Configuring the dataset to scan it and produce frequent pattern based on tuning the application.

Step-3: Displaying the frequent pattern in tree view. It consists of General PE and Control PE (Processing Elements).

Step-4: General PE is acting as a root element of the tree, and control PE act as a child element of the tree, based on their frequency count.

Step-5: Here, it process only the frequent patterns which crossing the threshold value of the support count.

Step-6: Based on the frequent patterns recommendations can be viewed.

Step-7: Suggest the data to know which one having maximum no .of frequency count.

Step-8: Data get recommended based on their behavioral sense of repetitions.

Step-9: It will show similar and non similar data together, based on frequency count.

Step-10: Frequent patterns are splitting as knowledgeable recommendations, General recommendations, and Noisy recommendations (Shown in Figure-3).

EXPERIMENTAL RESULTS

Here, selecting the dataset for processing (shown in Figure-4). Choosing the chess data set as a given datasets. Need to assign minimum frequency to find minimum support count to find frequent pattern from the required dataset.

After finding Min_supportCount, configure the dataset based on their threshold value (shown in Figure-5) and specify the chosen dataset transaction size.

Based on frequency count it can display the transaction item from chosen dataset. It's totally based on user's expectation and their predictive query requirements.

Recommendations (shown in Figure-6) are mentioned by users view. General recommendation and Noisy recommendation explaining dense and sparse patterns.

PERFORMANCE EVALUATION

For representing the graph (shown in Figure 7) and the input of frequent pattern count is taken in x axis, and Dense and sparse patterns Count in y axis value. Now analyzed the frequent patterns and producing the efficiency of this approach.

Survival

This study can compete with the real time approach of online shopping and web browsing eBay applications through Pattern recommendation approach it can recommending product based on user's perspective.

Use of Cache

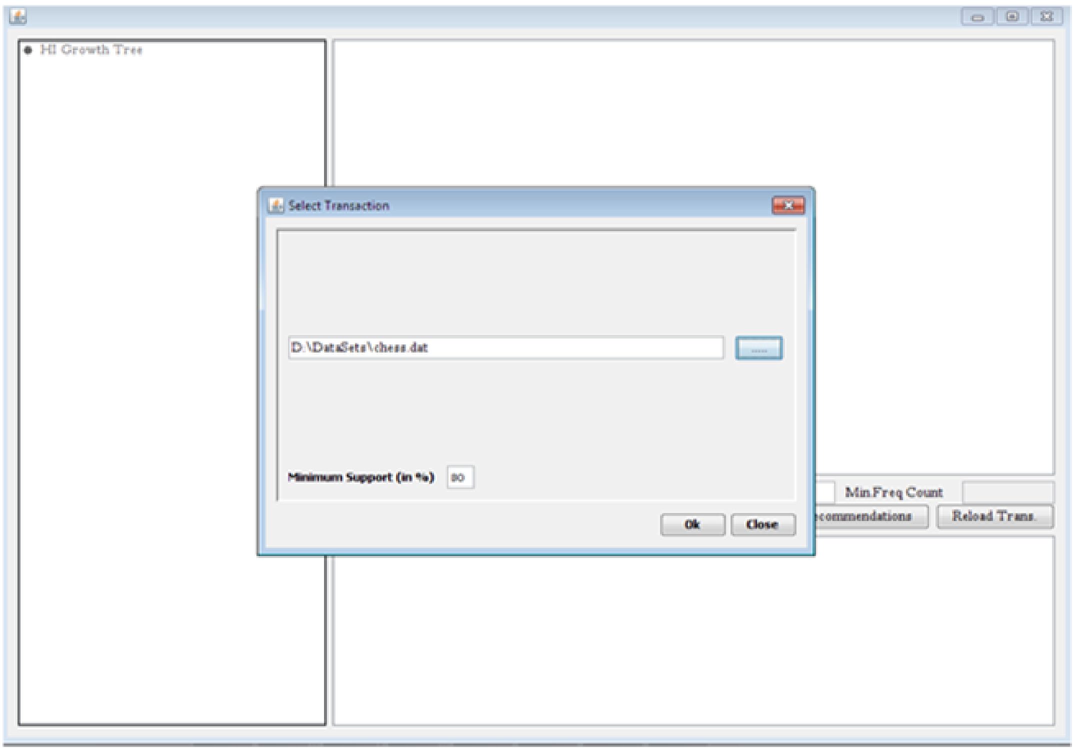
Frequent recommendations can be cached to further increase the efficiency of the application. So, no need to re-evaluate the cache data every time. This approach only mentioned in real time approach based on history of shopping mode.

Use of integrity

This performance can analyze in future works also. E.g. Signature verification process for authentication approach. Here it can generate pattern as trained dataset. Using pattern mining it can achieve the character reorganization. Tree based mining used to mine trained set, image, and symbols which are defined already.

CONCLUSIONS

In this paper, proposed novel tree based data structure, HI-Growth Tree for storing compact compressed and analyzed information about the frequent pattern, and developed pattern prediction method. HI-Growth approach implemented for the efficient mining of any size of transactional database. Here the size of transactional database not an issue. There are several advantages in HI-Growth tree approach over other approaches; it constructs highly compressed, HI-Growth frequent pattern tree consists of crucial information, which naturally smaller than original database. It can save costly database scan for more time to predict frequent patterns in pattern mining process. It reduce the memory consumption through avoiding long time scanning for implement item set matching for every cache mechanism. The performance studies shows, Hi-Growth Tree mining approach can analyze, thousands and ten thousands of large scale transaction in minimal of time requirements, based on current recommendation generation based approach. The following tasks can be taken as an enhancement for the proposed work. To predict the item based on users behavior. It can give some suggestion about their nature of work and category. Without getting value from users it recommend the item. E.g. Online shopping. It can implement tree based mining in training set and also implementing pattern mining in character reorganization; through

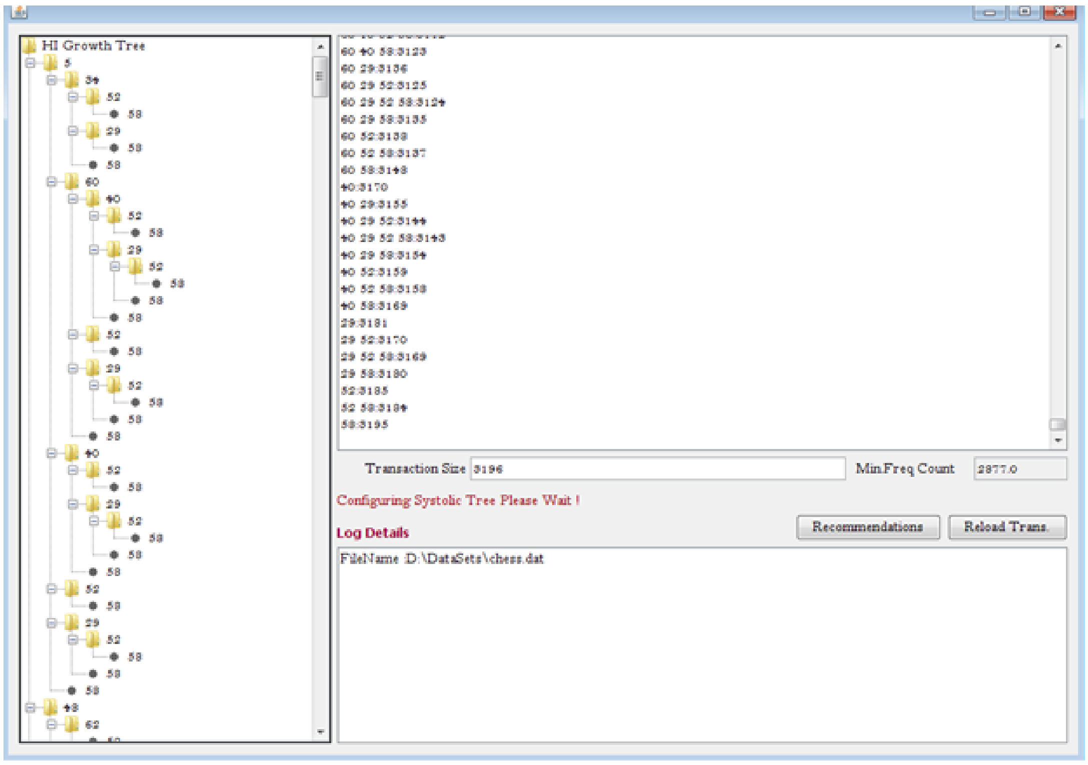


D:\DataSets\chess.dat

Minimum Support (in %) 80

Figure-4. Selecting dataset for processing.

VOL. 10, NO. 6, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences



```
HI Growth Tree
  5
    34
      52
        58
      29
        58
      58
    60
      40
        52
          58
        29
          52
            58
          58
        58
      52
        58
      29
        52
          58
        58
      58
    40
      52
        58
      29
        52
          58
        58
      58
    52
      58
    29
      52
        58
      58
    58
  48
    62
```

```
60 40 58:3123
60 29:3136
60 29 52:3125
60 29 52 58:3124
60 29 58:3135
60 52:3138
60 52 58:3137
60 58:3148
40:3170
40 29:3155
40 29 52:3144
40 29 52 58:3143
40 29 58:3154
40 52:3159
40 52 58:3158
40 58:3169
29:3181
29 52:3170
29 52 58:3169
29 58:3180
52:3185
52 58:3184
58:3195
```

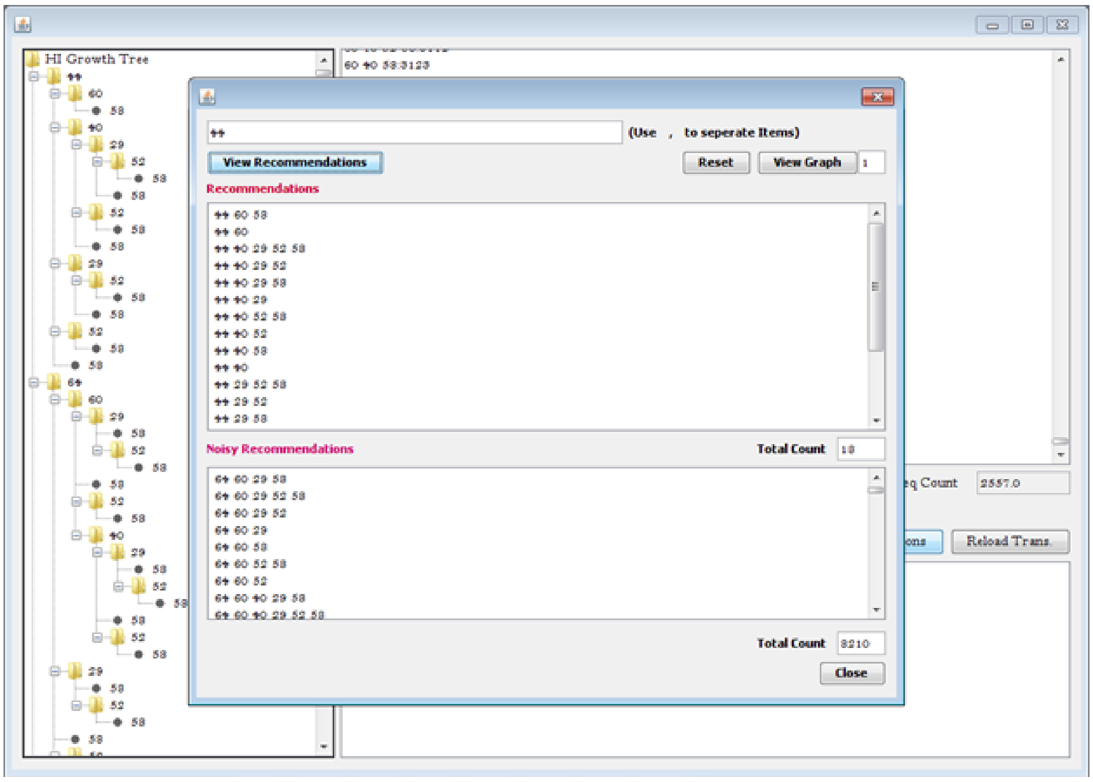
Transaction Size 3196 Min.Freq Count 2877.0

Configuring Systolic Tree Please Wait !

Log Details

```
FileName :D:\DataSets\chess.dat
```

Figure-5. Configuring HI-Growth tree.



```
HI Growth Tree
  44
    60
      58
    40
      29
        52
          58
        58
      52
        58
      58
    29
      52
        58
      58
    52
      58
    58
  64
    60
      29
        58
        52
          58
      58
      52
        58
      40
        29
          58
          52
            58
        58
        52
          58
    29
      58
      52
        58
    58
```

```
60 40 58:3123
```

44 (Use , to seperate Items)

1

Recommendations

```
44 60 58
44 60
44 40 29 52 58
44 40 29 52
44 40 29 58
44 40 29
44 40 52 58
44 40 52
44 40 58
44 40
44 29 52 58
44 29 52
44 29 58
```

Noisy Recommendations Total Count 18

```
64 60 29 58
64 60 29 52 58
64 60 29 52
64 60 29
64 60 58
64 60 52 58
64 60 52
64 60 40 29 58
64 60 40 29 52 58
```

Total Count 3210

eq Count 2557.0

Figure-6. Displaying recommendation.

VOL. 10, NO. 6, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

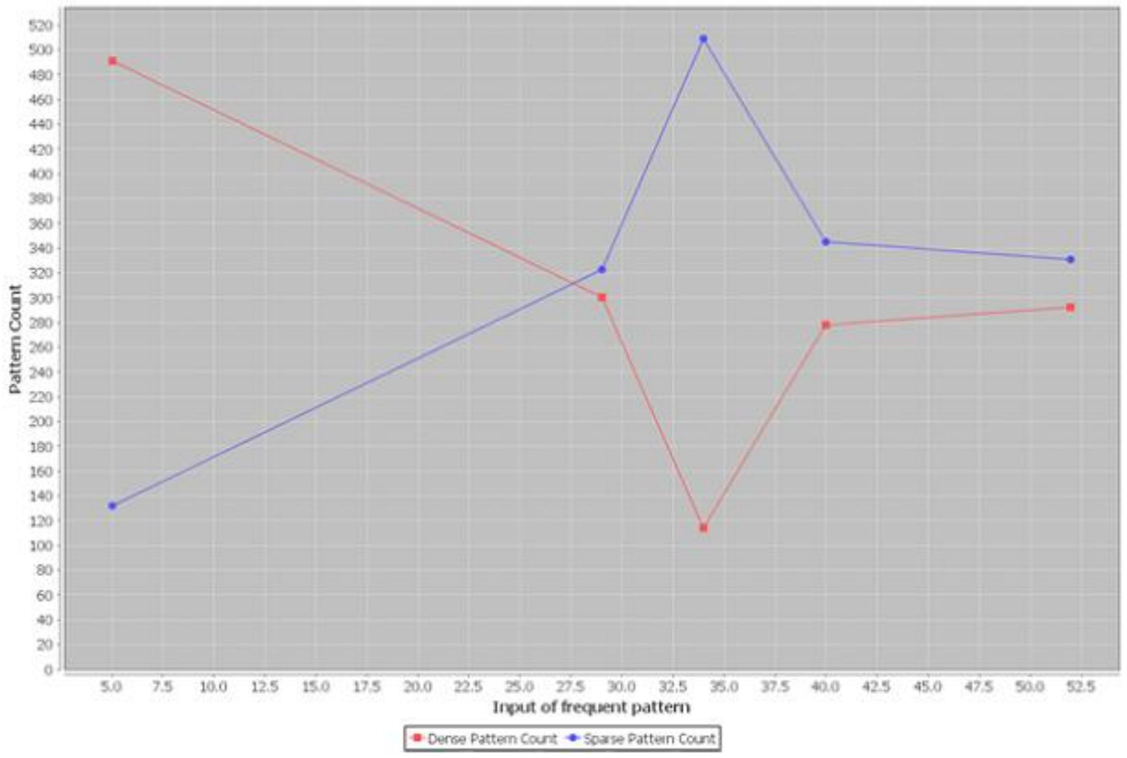


Figure-7. Graph representation for analyzing dense and sparse patterns.

REFERENCES

[1] Song Sun, Joseph Zambreno. 2011. Design and analysis of reconfigurable platform for frequent pattern mining. IEEE transactions on parallel and distributed systems. 22(9).

[2] J. Han, J. Pei, Y. Yin, and R. Mao. 2004. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery. 8(1): 53-87.

[3] S. Sun and J. Zambreno. 2008. Mining Association Rules with Systolic Trees. Proc. Int'l Conf. Field-Programmable Logic and Applications (FPL '08).

[4] R. Narayanan, D. Honbo, G. Memik, A. Choudhary and J. Zambreno. 2007. An FPGA Implementation of Decision Tree Classification. Proc. Conf. Design, Automation, and Test in Europe (DATE). pp. 189-194.

[5] Z. Baker and V. Prasanna. 2005. Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs. Proc. IEEE Symp. Field- Programmable Custom Computing Machines (FCCM). pp. 3-12.

[6] Y.-H. Wen, J.-W. Huang and M.-S. Chen. 2008. Hardware-Enhanced Association Rule Mining with Hashing and Pipelining. IEEE Trans. Knowledge and Data Eng. 20(6): 784-795.

[7] C. Lucchese, S. Orlando and R. Perego. 2004. kDCI: On Using Direct Count Up to the Third Iteration. Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations (FIMI).

[8] J. Bentley and H. Kung. 1979. A Tree Machine for Searching Problems. Proc. Int'l Conf. Parallel Processing (ICPP). pp. 265-266.

[9] Avrilia Floratou, Sandeep Tata, and Jignesh M. Patel. 2011. Efficient and Accurate Discovery of Patterns in Sequence Data Sets. IEEE transactions on parallel and distributed systems. 239(8): 1154-1168.

[10] Jun TAN and Yingyong BU and Bo YANG. 2009. An Efficient Frequent Pattern Mining Algorithm. Sixthe International Conference on Fuzzy Systems and Knowledge Discovery. pp. 48-50.

[11] Jin Qian, and Feiyue Ye. 2007. Mining Maximal Frequent Itemsets with Frequent Pattern List. pp. 628-632.

[12] Xiaolei Tan, Haiwei Pan, Qilong Han, and Jun Ni. 2009. Domain Knowledge-Driven Association Pattern Mining Algorithm on Medical Images. Fourth International Conference on Internet Computing for Science and Engineering, IEEE Computer Society. pp. 30-35.

[13] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010. Mining Frequent Subgraph Patterns from Uncertain Graph Data. IEEE transactions on parallel and distributed systems. 22(8): 1203-1218.

[14] Shingo Mabu, Ci Chen, Nannan Lu, Kaoru Shimada and Kotaro Hirasawa. 2011. An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining using Genetic Network Programming. IEEE transactions on systems, man, and cybernetics-part C: Applications and Reviews. 41(1): 130-139.

[15] Kuo-Cheng Yin, Yuh-Longh Hsieh, and Don-Lin Yang. 2010. GLFMiner: Global and Local Frequent Pattern Mining with Temporal Intervals. 5th IEEE Conference on Industrial Electronics and Applications. pp. 2248-2253.