Mining Frequent Patterns of Crime using FP-Growth with Multiple Minimum Supports based on Shannon Entropy

George Matto
School of Computational and Communication Sciences and Engineering, Nelson Mandela African Institution of Science and Technology
Arusha, Tanzania

Joseph Mwangoka
School of Computational and Communication Sciences and Engineering, Nelson Mandela African Institution of Science and Technology
Arusha, Tanzania

ABSTRACT
FP-Growth is one of the most effective and widely used association rules mining algorithm for discovering interesting relations between items in large datasets. Unfortunately, classical FP-Growth mines frequent patterns by using single user-defined minimum support threshold. This is not adequate for real life applications such as crime patterns mining. On one side, if minimum support is set too low, huge amount of crime patterns (including uninteresting patterns) may be generated, and on the other side, if it is set too high lots of interesting patterns (including seasonal patterns) may be lost. This paper proposes the use of Multiple Item Support (MIS) thresholds instead of single minimum support to tackle the challenge. We employ Shannon entropy method to develop an algorithm that obtains MIS values from crime datasets. The proposed approach is tested on different sizes of input data via a developed working prototype. Experimental results show that our suggested approach outperforms classical FP-Growth in terms of running time and memory use.

Keywords
FP-Growth, Crime Pattern, Multiple Minimum Supports, Shannon Entropy.

1. INTRODUCTION
Mining association rules is one of the key data mining tasks. It discovers interesting relations amongst items in large databases. An association rule, according to Jiawei *et al.* [1], is an implication of the form $A \rightarrow B$, where, A is the antecedent while B is the consequent. It gives information of the form; when A appears then B will possibly appear too. The idea of discovering association rules begun from the investigation of market-basket data. It is on such investigations that rules like "*if a customer buy bread he is 85% likely to purchase butter also*" are generated. Today, association rule mining has become a powerful technique with huge potential and wide applications in several domains. One of such domains is crime patterns analysis, in which crime analysts mine association rules from crime datasets. Such rules help to discover patterns in criminal behaviour that can help to predict crime, anticipate criminal activities, and prevent further crimes [2], [3] and [26].

There are several association rules mining algorithms. According to Kumbhare and Chobe [4] Apriori, FP-Growth and Eclat are the most widely used. Comparative studies among such algorithms (see for example [4] and [5]) indicate FP-Growth as more efficient in terms of number of database scans, execution time and memory consumption. FP-Growth has also been used intensively in crime patterns mining [5], [6], and [26]. Classical FP-Growth mines frequent patterns by using a single user-specified minimum support (abbreviated as *minsup*) [7]. However, using single minimum support for crime patterns mining is not adequate since it does not reflect the nature of each crime item in the dataset. If, for instance, such a minimum support is very low, huge amount of crime patterns (including uninteresting patterns) will be generated and thus provides misleading results. On the other side, if it is set too high many interesting patterns may be lost since some of crimes (e.g. killing of people with albinism in some countries like Tanzania) occur seasonally and thus rarely found in the dataset.

To tackle the challenge, this paper proposes a method that replaces the minimum support value defined by user with an aggregate function that computes multiple minimum supports basing on empirical analysis of the dataset. The proposed function is based on the Shannon entropy equation. The proposed solution is tested on four clusters of training data with different sizes and compare the run time and memory utilization of our algorithm versus classical FP-Growth. Experimental results revealed that the suggested solution is more effective in terms of run time and memory consumption.

The remainder of this paper is organized as follows; Section 2 presents a survey of related literature. In this section various association rule-mining concepts are discussed, the FP-Growth algorithm is revisited, and previous studies that attempted to improve the FP-Growth are reviewed. Section 3 describes the proposed approach. Section 4 presents experimental results, and the paper is concluded in Section 5.

2. RELATED LITERATURE
2.1 Association Rules Mining
Association rule mining problem is stated by [8] and [9] like this. Suppose $I = \{i1, i2, i3, \dots, in\}$ and T are sets of finite items and transactions respectively. Suppose DB is a transaction database that comprises set of items in I. $DB = T1, T2, T3, \dots, Tn$ where $Ti(i \in [1 \dots n])$. Suppose X and Y are itemsets, an itemset is a set of items in I i.e. $X \subseteq I$ and $Y \subseteq I$. An association rule of itemsets X and Y is denoted as $X \rightarrow Y$. This is the relationship between X and Y, given that X and Y are disjoint itemsets.

Support of an itemset X, represented as $\sup(X)$, is the fraction of transactions T, comprising X. Similarly, support of itemset Y, represented as $\sup(Y)$, is the fraction of transactions T, comprising Y.

$$\sup(X) = \frac{Number\ of\ Transactions\ comprising\ X}{Total\ Number\ of\ Transactions}$$

$$\sup(Y) = \frac{Number\ of\ Transactions\ comprising\ Y}{Total\ Number\ of\ Transactions}$$

Support of an association rule $X \rightarrow Y$, denoted as sup $(X \rightarrow Y)$, is the fraction of transactions comprising of both X and Y.

$$\text{sup }(X \rightarrow Y) = \frac{Number\ of\ Trans'\ comprising\ X, Y}{Total\ Number\ of\ Transactions}$$

Confidence of an association rule $X \rightarrow Y$, denoted as $conf\ (X \rightarrow Y)$, is an indication of how often the rule have been found to be true. It is given as

$$\text{conf }(X \rightarrow Y) = \frac{sup\ (X \cup Y)}{sup\ (X)}$$

Zeng *et al.* [10] pointed out that for a given user-specified minimum support, *minsup*, if the itemset meets the condition $\text{sup}(X) \geq minsup$, then itemset X is regarded as frequent itemset and conversely itemset X is regarded as infrequent itemset.

2.2 FP-Growth Algorithm

FP-Growth is one of the most efficient association rule mining algorithms. The algorithm mines frequent itemsets without generating the candidates. FP-Growth algorithm was proposed by Han et al. [11] to overcome the multiple database scans and candidate generations of the Apriori algorithm [12]. According to Han et al. [11] FP-Growth uses a divide-and-conquer strategy to mine frequent itemsets. It uses two steps. First, build an FP-Tree by condensing a transaction database into a compressed structure. And second, extract itemsets directly from the FP-Tree that was built in the first step.

Step 1: Building an FP-Tree

According to Han et al. [11], the algorithm for FP-Tree construction requires two sorts of inputs; the transaction database (DB) or the dataset and the minimum support threshold. The general steps for the FP-Tree construction are as shown in algorithm 1.

Table 1. FP-Tree construction

Algorithm 1: FP-Tree Construction
Input: DB, *minsup* (ξ) *Output*: FP-tree *Process*: 1. Scan DB once. Discard infrequent items and collect the set of frequent items, F, (and their supports). Sort F in support-descending order 2. Scan DB again to construct FP-Tree.

The way this algorithm works can be described by considering a transaction database, DB, in Table 2. DB has eight different crime items, i.e. crime1, crime2, crime3, crime4, crime5, crime6, crime7 and crime8 represented as C1, C2, C3, C4, C5, C6, C7 and C8 respectively. Let's consider *minsup* threshold for this case to be 2 (i.e. ξ = 2).

Table 2. Transaction Database (DB)

TID	Items	TID	Items
T001	C1, C2	T011	C1, C2
T002	C1, C5, C6	T012	C1, C3
T003	C3, C4	T013	C1, C2
T004	C1, C2, C8	T014	C2, C5, C6, C7
T005	C3, C4	T015	C3, C4
T006	C1, C3	T016	C1, C2, C4
T007	C1, C2	T017	C3, C4
T008	C5, C6	T018	C1, C3
T009	C3, C4, C7	T019	C1, C2, C5
T010	C1, C2	T020	C3, C4

According to Algorithm 1 first, a scan of DB collects F, the set of frequent items and the support of each of those frequent items. F = {C1:12, C2:9, C3:9, C4:7, C5:4, C6:3, C7:2, C8:1}. Then sort F in support-descending order as FList., while discarding those items whose support is less than *minsup* threshold. FList is the list of frequent items. Since C6, C7 and C8 has support less than 3 they will be discarded and thus FList = {C1:12, C2:9, C3:9, C4:7, C5:4, C6:3, C7:2}. Second, this order (i.e. FList) and the methods indicated in the FP-Tree construction algorithm are used to build an FP-Tree. Figure 1 is the constructed FP-Tree. Link is added to speed lookup and easy matching of the pointer to FP-Tree.

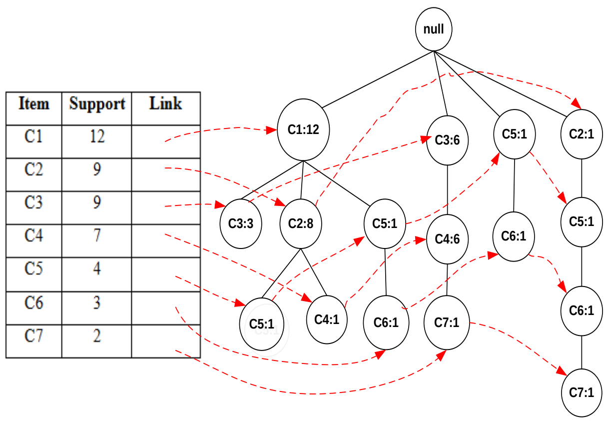

Fig 1: A complete FP-Tree for crime database (DB)

Step 2: Frequent Itemset Generation

After completing construction of the FP-Tree, the following step of the FP-Growth algorithm is to extract frequent itemsets from the FP-Tree. According to Tohidi and Ibrahim [13] the FP-tree is extracted by dividing the tree (or the compressed database) into sub-databases (conditional pattern base). And from those conditional pattern bases we find out pattern fragments (conditional FP-Tree) associated with each of the databases, and lastly do mining recursively on the tree. Table 3 shows the conditional pattern base, conditional FP-Tree and the mined frequent itemsets from the FP-Tree in Figure 1.

Table 3. Mined Patterns (with ξ=3)

Items	Conditional Pattern Base	Conditional FP-Tree	Frequent Itemsets
C7	{C3, C4:1}, {C3, C5, C6:1}	NULL	NULL
C6	{C1, C5:1}, {C5:1}, {C2, C5:1}	{<C5:3>}	{C5, C6:3}
C5	{C1, C2:1}, {C1:1}, {C2:1}	{<C1:2>, <C2:2>}	{C1, C5:2, C2,C5:2, C1,C2,C5:2}
C4	{C1, C2:1}, {C3:6}	{<C3:6>}	{C3,C4:6}
C2	{C1:8}	{<C1:8>}	{C1,C2:8}
C3	{C1:3}	{<C1:3>}	{C1, C2:7}

2.3 Studies to improve FP-Growth based on single minimum support

FP-Growth algorithm has been blamed to produce large number of conditional pattern base and consequently conditional FP-tree recursively in mining the frequent patterns [14]. Our illustration results in Table 2, which is based on a 20 transactions dataset, can be a good example on this. Studies have thus shown that the algorithm becomes less effective when the dataset size increases. Consequently, several methods have been suggested to improve efficiency of the algorithm. Some of such methods are; implementation of parallel FP-Growth ([15], [14] and [16]), mining only top-*k* frequent itemsets (Lee and Clifton [17] and Wang *et al.* [3], and the use of distributed computing for frequent patterns mining (Deng and Low [18] and Itkar and Kulkarni [19]). Although all of these, and other similar approaches, try to tackle the challenge of the algorithm especially with the increasing datasets, they ignore the mining for infrequent items. In fact, these techniques employ a single user specified *minsup*. The user specified minimum support threshold assumes that items in the dataset are of identical nature and occurrences. This is however a rare situation in real life applications especially in the crimes datasets where some crime items appear so regularly in the dataset while others appear rarely. It is on this same line that Isafiade *et al.* [2] used a quartile floor-ceiling functions of the descriptive statistics to propose a pruning step of the FP-Growth. This approach automatically identifies the *minsup* threshold for the fine-tuning of the algorithm's pruning step for identifying frequent crime pattern trends. Unfortunately this method works only for small datasets.

2.4 Approaches that use multiple minimum supports

To improve extraction of frequent itemsets studies have proposed the use of multiple minimum supports approach (see for example [26] and [28]). Liu *et al.* [20] used this approach to mine rare itemsets through an Apriori-like algorithm called Multiple Support Apriori (MSApriori). According to the author, the approach assigns each item with a minimum support value known as "Minimum Item Support" (MIS). Frequent itemsets are produced under the condition that they satisfy the lowest MIS value amongst the corresponding items.

In this multiple minimum support approach, association rules definition remains the same as presented in section 2.1 above, but the rule's *minsup* is defined in terms of MIS of items occurred in the rule. In other words, each item in the database can have MIS value that is calculated using a formula or stated by the user. The provision of different MIS values for different items helps the user to efficiently define distinctive support needs for distinctive rules. For instance, if a dataset consists of four crime items, e.g. *murder*, *robbery*, *killing_of_albino*, and *rape*, then MIS values could vary as follows: $MIS(murder) = 3\%$, $MIS(robbery) = 5\%$, $MIS\ (killing\_of\_albino) = 0.1\%$, $MIS(rape) = 0.5\%$. In addition, the minimum support for any itemset $X = \{i_1, i_2, \ldots, i_k\}, 1 \leq k \leq n,$ is given as

$$minsup(X) = minimum(MIS(i_1), MIS(i_2), \ldots, MIS(i_k))$$

According to Liu *et al.* [20] MIS for every 1-itemset (in the MSApriori), expressed as $MIS(i)$, is calculated using the following percentage-based formula

$$\text{MIS(i)} = \begin{cases} M(i) & M(i) > LS \\ LS & Otherwise \end{cases}$$

$$M(i) = \beta \, . f(i)$$

Where, LS is the least support. This is stated by the user to express the lowest allowed minimum item support, $f(i)$ is the frequency of occurrence of an item in the dataset, and β is the value that governs how MIS values should be associated to their occurrences.

Unfortunately, MSApriori undergoes the same performance drawbacks as the classical Apriori algorithm [11]. FP-Growth-like algorithms that use multiple *minsup* were then proposed. Specifically, Ya-Han and Yen-Liang [21] proposed CFPGrowth algorithm, and later Kiran *et al*. [22] proposed the CFPGrowth++ algorithm. According to Kiran [9], the main idea of CFPGrowth++ was the use of the notion of Support Difference (SD) instead of a percentage-based methodology, to specify items' MIS values as follows.

$$MIS(i) = maximum(Sup(i) - SD, LS)$$

SD can be either user-specified or calculated from the formula $SD = \boldsymbol{\lambda}(1 - \beta)$, where, λ is a parameters such as mean, median, and mode of the item, and β and LS is the same as for MSApriori.

Although CFPGrowth++ have shown improvements as compared to its predecessors, studies identified that its main weakness is on stating "good" MIS value for each item. According to Chen *et al*. [23], for example, the algorithm requires users to identify a minimum support value for each item and continuously tune it to obtain the best value. This is a costly in terms of time and efforts.

3. THE PROPOSED APPROACH

The approach for multiple minimum supports FP-Growth in this paper is based on Shannon entropy (also known as Information Entropy). Entropy is simply the average (expected) amount of the information from the event. According to Lesne [24] the Shannon entropy of X is given as

$$E(X) = -\sum_{i=1}^{n} P_i * log_2 P_i \qquad (1)$$

Where, $E(X)$ (sometimes denoted as $H(X)$) is the entropy of a random variable/item X, n is the number of different outcomes, and P_i is the probability of a given item.

The Shannon entropy equation (1) is used to obtain the entropy of each of the crime items in the crime dataset basing on the frequency of occurrence of each of those items. To reflect the present context, the Shannon entropy equation is rewritten as shown in equation (2) below. In this equation, C represents crime item.

$$E(C) = -\sum_{i=1}^{n} P(C_i) * log_2 P(C_i) \qquad (2)$$

This gives the probability of occurrence of a particular crime from a set of similar crime items. In this case, when the number of crime items increase the probability decreases, and thus the entropy. In other words, highly occurring crimes will have higher entropy than the low occurring crimes. To avoid this situation, reciprocal of the entropy is taken. Reciprocal of the entropy assigns entropy values that increase with the increase of frequency of occurrence of an item. The entropy equation for crime items thus becomes as shown in equation (3).

$$E(C) = (-\sum_{i=1}^{n} P(C_i) * log_2 P(C_i))^{-1} \qquad (3)$$

The entropy value obtained in equation (3) above gives us the MIS values of crime items in the dataset. Equation (3) is thus rewritten in terms of MIS. In fact, $E(C)$ is replaced by $MIS(C_i)$ to obtain equation (4).

$$MIS(C_i) = (-\sum_{i=1}^{n} P(C_i) * log_2 P(C_i))^{-1} \qquad (4)$$

Where $MIS(C_i)$ is the minimum item support of crime item i when $MIS(C_i)$ is greater than or equals to LS, otherwise $MIS(C_i) = LS$. As Liu *et al.* [20] defines, LS is the user-specified *Least Support.* The final MIS of an item will not entirely depend on the value obtained from our aggregate function in equation (4). Depending on the nature of dataset, calculated MIS value could even be one or less than one. The concept of LS, where user will set the least support value, is used to avoid the possibility of getting unreasonable MIS.

Algorithm for obtaining MIS by using this approach is shown in Table 4. Figure 2 shows the implementation of the proposed algorithm in Java.

Table 4. The proposed algorithm for specifying MIS values

Algorithm 2: Specifying MIS values using Shannon Entropy
Input: Transaction database (DB), Least Support (LS). *Output*: Complete set of MIS values *Process*: 1. Scan DB; Count the total number of available distinct crimes in each of the crime categories found in DB. Call the counts $N(C_i)$. 2. For every crime type (C_i) compute the probability of (C_i) (i.e. $P(C_i)$) as $1/N(C_i)$ 3. Compute the entropy of crime type (C_i) as $E(C_i) = -(P(C_i) * \ln(P(C_i)))$ 4. Compute reciprocal of the entropy obtained in 3 above as $E(C_i)^{-1}$ 5. If $E(C_i)^{-1} \geq LS$ then $MIS(C_i) = E(C_i)^{-1}$ else $MIS(C_i) = LS$

```java
private void Calculate_MIS_Values(){
    try{   //obtaining crime dataset from the database of reported crimes
        String sqls = "select crimes from crimes_reported";
        pst=conn.prepareStatement(sqls);
        rs=pst.executeQuery();
        while(rs.next()){
            search += rs.getString("crimes");
        }
           //Calculating MIS values from crime database using Entopy equation by George Matto & Joseph Mwangoka
        String[] split = search.split(" ");
        double ks = 1; int LS=2;
        Arrays.stream(split).collect(Collectors.groupingBy(s -> s))
        .forEach((k,v) ->
        crimes_reported += k+" "+ Math.round(ks/(-((ks/(double)v.size())*(Math.log(ks/(double)v.size()))))) +" ");

            //Checking if the calculated MIS is less than LS
        String[] splits = crimes_reported.split(" ");
        for(int i=1; i<splits.length; i=i+2){
            if(splits[i].length()>7){ splits[i]="2";
            }else if(Integer.valueOf(splits[i])>=LS){ splits[i]=splits[i];
            }else{ splits[i]="LS";
            }
            crimes_mis += splits[i-1]+" "+splits[i]+"\n";
        }
            //end of calculating MIS values
        ItemSets.setText(crimes_mis);
        crimes_mis = "";  crimes_reported = "";
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
    finally{
        try{
            rs.close(); pst.close();
        }
        catch(Exception e){}
    }
}
```

Fig 2: Java codes for obtaining MIS values using our proposed method

The calculated MIS values are then used to obtain the conditional pattern base and conditional FP-tree from the FP-tree. FP-tree construction uses the same procedures as shown in Algorithm 1 above, but in this case ξ is LS. For example, suppose $LS = 2$, basing on the transaction database in Table1 and the constructed FP-tree in Figure 1, the obtained MIS values, conditional pattern base, conditional FP-Tree and the mined frequent itemsets will be as shown in Table 5.

Table 5. The proposed algorithm for specifying MIS values

Items	MIS	Conditional Pattern Base	Conditional FP-Tree	Frequent Itemsets
C7	3	{C3, C4:1}, {C3, C5, C6:1}	NULL	NULL
C6	3	{C1, C5:1}, {C5:1}, {C2, C5:1}	{<C5:3>}	{C5, C6:3}
C5	3	{C1, C2:1}, {C1:1}, {C2:1}	NULL	NULL
C4	4	{C1, C2:1}, {C3:6}	{<C3:6>}	{C3,C4:6}
C2	4	{C1:8}	{<C1:8>}	{C1,C2:8}
C3	4	{C1:3}	NULL	NULL

4. PROTOTYPE IMPLEMENTATION

In order to examine effectiveness of the proposed solution a working prototype was developed basing on the suggested approach. The developed prototype allows reporting of crime as well as extraction of patterns from crime data. This prototype is named Crime Reporting and Pattern Extraction System (CRaPES). CRaPES is a desktop application developed by using Java. But since this prototype allows reporting and thus storing of crimes, it comprises of a crime database that was developed by using SQLite.

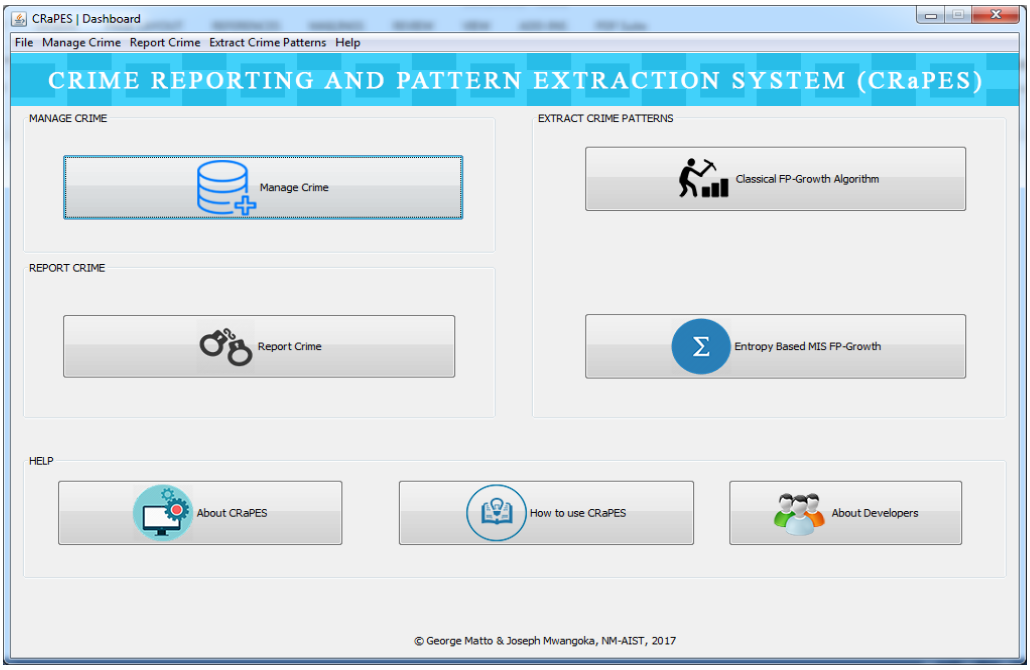

Fig 3: CRaPES Dashboard

In running the prototype, a MacBook Air (MacOS Sierra version 10.12.4), Intel Core i5 1.6GHz processor machine with 8GB of memory was used. The prototype allows extraction of crime patterns from crime data stored in CRaPES database as well as from external sources. Since CRaPES database did not have data enough for experimentations, external sources of data were used to evaluate this prototype. Specifically, these data were obtained from the link https://catalog.data.gov/dataset?tags=crime.

CRaPES allows data file to be imported and then MIS values of each of the crime item in the dataset to be calculated based on the method proposed in this paper. Figure 4 is the CRaPES interface for MIS calculation. After calculating MIS values of the crime items, the file containing those values (i.e. MIS file) is exported.

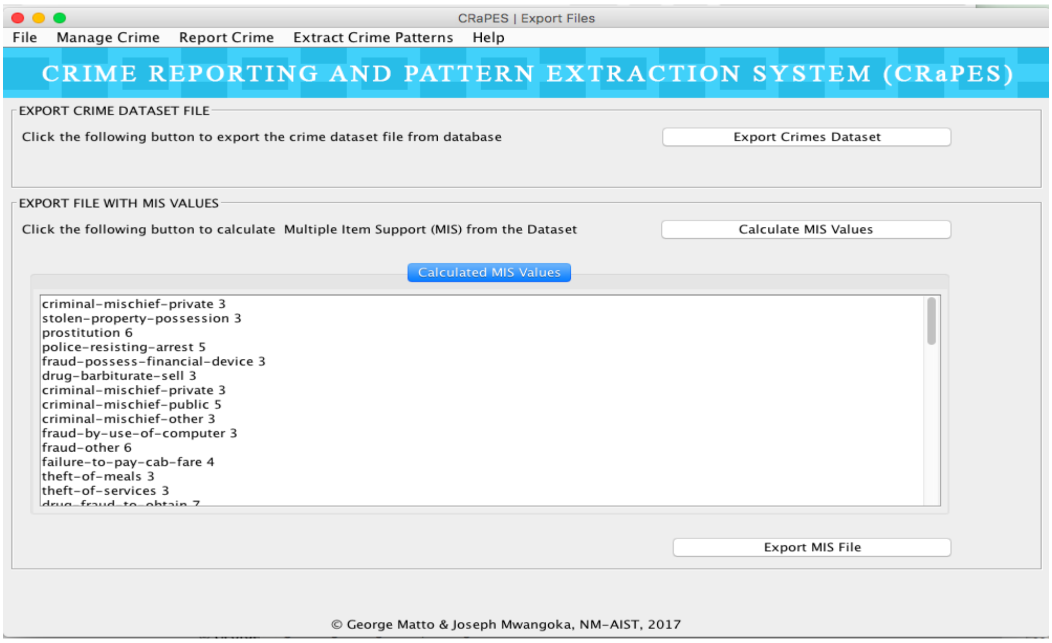

Fig 4: Calculated MIS values

After obtaining MIS file the next step is the actual pattern extraction. In this case, as shown in Figure 5, both data file (crime dataset) and its MIS file are imported to the system and the algorithm is run to obtain the patterns

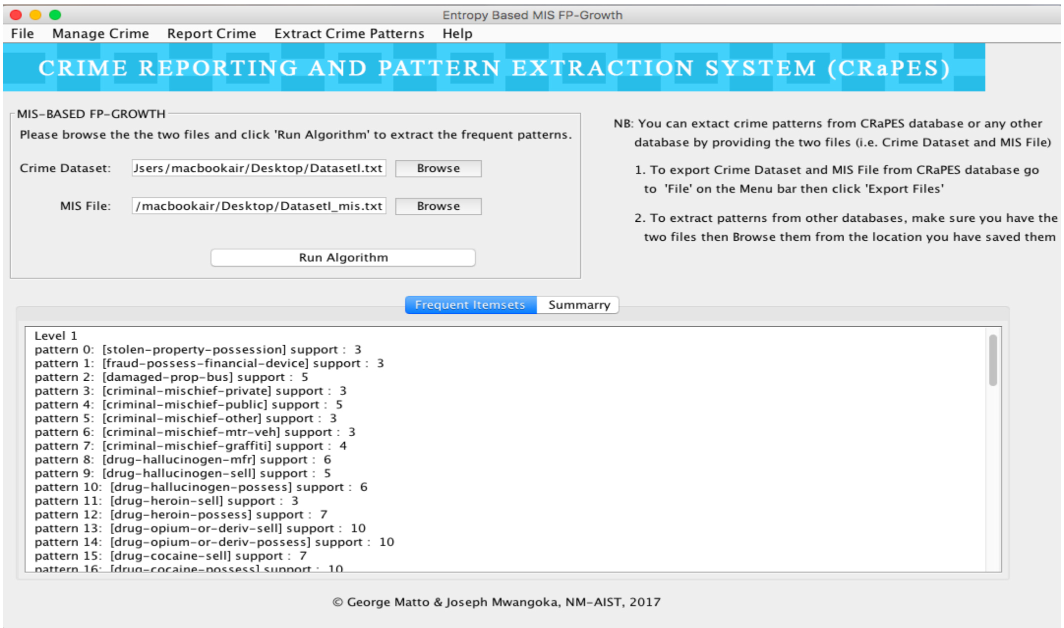

Fig 5: Extracted Patterns

We furthermore evaluated how our proposed solution behaves on the varying sizes of crime data in comparison with the existing approaches. For that to be achieved, we created four clusters of input data; the first cluster was 5KB with 847 records (we represent this dataset as DATASET-I), the second was 10KB with 1390 records (DATASET-II), third was 15KB with 2162 records (DATASET-III), and the fourth was 20KB with 2910 records (DATASET-IV). Our evaluation criteria were execution time and memory usage. Thus, we compared execution time and memory consumption on our proposed solution over FP-Growth with varying minimum support thresholds.

Table 6 shows memory consumption in the classical FP-Growth with minimum supports of 10, 20 and 30, and memory consumption with our proposed solution. It was observed that varying user-defined minimum supports did not affect memory consumption of the FP-Growth algorithm, but when the size of the dataset increased our proposed solution was more effective in terms of memory consumption.

Table 6. Memory Use

Datasets	Memory Use (in MB)			
	MinSup=10	MinSup=20	MinSup=30	Proposed Approach
DATASET-I	1.6	1.6	1.6	1.6
DATASET-II	2.24	2.24	2.24	2.24
DATASET-III	2.88	2.88	2.88	2.56
DATASET-IV	3.52	3.52	3.52	2.86

Concerning execution time, as shown in Figure 6 we observed an increase of time to complete algorithm's execution as the size of dataset increased. Our proposed solution, however, recorded a lower execution time as compared to FP-Growth algorithm with minimum support values of 10, 20 and 30.

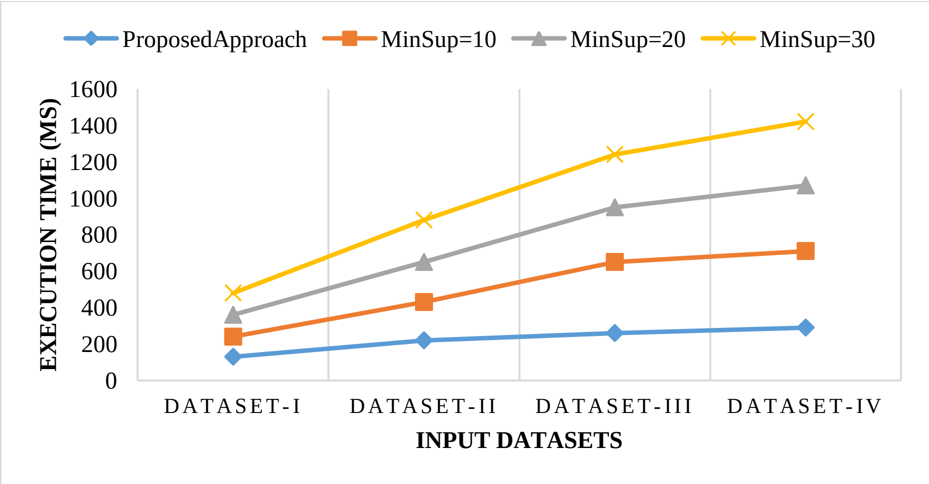

Fig 6: Execution Time

5. CONCLUSION

This research paper proposed an approach for extracting frequent patterns of crime by using the Multiple Itemset Support (*MIS*) approach to improve rare itemset mining with the FP-Growth algorithm. Specifically, the paper has proposed an algorithm for obtaining *MIS* values based on Shannon entropy equation. This approach scans the entire dataset and assigns *MIS* values on each crime item in the dataset basing on its frequency of occurrence. In this way the proposed approach tackles the rare item problem of the FP-Growth. The solution was tested on varying crime datasets and compares its execution time and memory consumption over classical FP-Growth algorithm. The testing was achieved through a developed crime pattern-mining prototype. Experimental results show that the proposed solution is reasonable and more effective as it outperforms classical FP-Growth in terms of execution time and memory use.

6. ACKNOWLEDGMENTS

We are grateful to the Tanzania Communications Regulatory Authority (TCRA) for financial support that enabled us to accomplish this research work.

7. REFERENCES

[1] Han J., Jian P., and Yiwen Y. 2000. “Mining frequent patterns without candidate generation”, ACM SIGMOD Record. Vol. 29. No. 2. ACM.

[2] Isafiade, O., Bagula, A. Berman, S. 2015. A Revised Frequent Pattern Model for Crime Situation Recognition Based on Floor-Ceil Quartile Function. Information Technology and Quatitative Management (ITQM2015). Procedia Computer Science, 55, 251 – 260.

[3] Wang, T., Rudin, C., Wagner, D. and Sevieri, R. 2013. Learning to Detect Patterns of Crime. *Springer*, *Berlin*, *Heidelberg*, 515-530.

[4] Kumbhare, T. A. and Chobe, S. V. 2014. An Overview of Association Rule Mining Algorithms. *International Journal of Computer Science and Information Technologies*, 5 (1), 927-930.

[5] Isafiade, O., Bagula, A. 2013. Citisafe: Adaptive spatial pattern knowledge using FP-Growth algorithm for crime situation recognition, i*n: Proc. IEEE International Conference on Ubiquitous Intelligence and Computing, IEEE*, pp. 551-556.

[6] Pereira, B. L. and Brandão, W. C. 2014. ARCA: Mining Crime Patterns Using Association Rules. *11th*

International Conference Applied computing, ISBN: 978-989-8533-25-8.

[7] Hu, Li‑Yu, Hu, Ya‑Han, Tsai, Chih‑Fong, Wang, Jian‑Shian and Huang, Min‑Wei. 2016. Building an associative classifier with multiple minimum supports. *SpringerPlus*. DOI 10.1186/s40064-016-2153-1.

[8] Fournier-Viger, P., Wu, C. W., Tseng, V. S. 2012. "Mining Top-K Association Rules**".** *Proceedings of the 25th Canadian Conf. on Artificial Intelligence (AI 2012), Springer*, LNAI 7310, pp. 61-73.

[9] Kiran R. Uday. 2011. "Generating Huge Number of Candidate Patterns and Multiple Scans on the Dataset" *PhD Thesis*. Center for Data Engineering International Institute of Information Technology Hyderabad, 500 032, India.

[10] Zeng, Y., Yin, S. Liu, J. and Zhang, M. 2015. Research of Improved FP-Growth Algorithm in Association Rules Mining. *Hindawi Publishing Corporation*. Scientific Programming, Article ID 910281, 6 pages.

[11] Han, J., Jian Pei, and Yiwen Yin, Runying Mao. 2004. "Mining frequent patterns without candidate generation: A Freequent Pattern Tree Approach", *Data Mining and Knowledge Discovery*, 8, 53–87.

[12] Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487-499, Santiago, Chile.

[13] Tohidi, H. and Ibrahim, H. 2010. A Frequent Pattern Mining Algorithm Based on FP-growth without Generating Tree. Proceedings of the Knowledge Management International Conference 2010, Kuala Terengganu (Malaysia), 25-27 May 2010, pages 671-676.

[14] Xia, D., Zhou, Y., Rong, Z. and Zhang, Z. 2013. IPFP: An Improved Parallel FP-Growth Algorithm for Frequent Itemsets Mining, *in: Proc. 59th ISI World Statistics Congress, Hong Kong (Session CPS026)*, pp. 4034-4039.

[15] Li, H., Wang, Y., Zhang, D., Zhang, M. and Chang, E. Y. 2008. "PFP: Parallel FP-Growth for query recommendation," In: Proceeding of the 2008 ACM conference on Recommender systems, Lausanne, Switzerland, 107-114.

[16] Pramudiono, I. and Kitsuregawa, M. 2003. "Parallel FP-Growth on PC Cluster," Advances in Knowledge Discovery and Data Mining, 2637, 467-473.

[17] Lee, J., Clifton, C. W. 2014. Top-k frequent itemsets via differentially private FP-Tree. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 931-940.

[18] Lingling Deng, Yuansheng Lou, 2015. "Improvement and Research of FP-Growth Algorithm Based on Distributed Spark", IEEE, pp. 105-108, doi: 10.1109/CCBD.2015.15.

[19] Itkar, S. A. and Kulkarni, U. V. 2013. Distributed Algorithm for Frequent Pattern Mining using Hadoop MapReduce Framework. *In Journal of Association of Computer Electronics and Electrical Engineer*, pp. 15-24.

[20] Liu, B., Hsu W. and Ma, Y. 1999. Mining association rules with multiple minimum supports, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, August 15-18, San Diego, California, United States, p.337-341.

[21] Ya-Han, H. and Yen-Liang, C. 2006. Mining association rules with multiple minimum supports: A new mining algorithm and a support tuning mechanism. *Decis. Support Syst*., 42: 1-24.

[22] Kiran, P., Uday, R. and Reddy, K. 2009. "An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules", IEEE Symposium on Computational Intelligence and Data Mining, pp. 340-347.

[23] Yi-Chun Chen, Grace Lin, Ya-Hui Chan, and Meng-Jung Shih. 2014. "Mining Frequent Patterns with Multiple Item Support Thresholds in Tourism Information Databases". Springer International Publishing Switzerland, pp. 89-98.

[24] Lesne, A. 2014. Shannon entropy: A rigorous notion at the crossroads between probability, information theory, dynamical systems and statistical physics. *Math. Struct. Comput. Sci. 24*, e240311.

[25] Bhatt, U., & Patel, P. 2015. A Novel Approach for Finding Rare Items Based on Multiple Minimum Support Framework. *Procedia Computer Science,* 57, 1088-1095.

[26] Matto, G. and Mwangoka, J. 2017. Detecting crime patterns from Swahili newspapers using text mining. *International Journal of Knowledge Engineering and Data Mining*, Vol. 4, No. 2, pp.145–156.