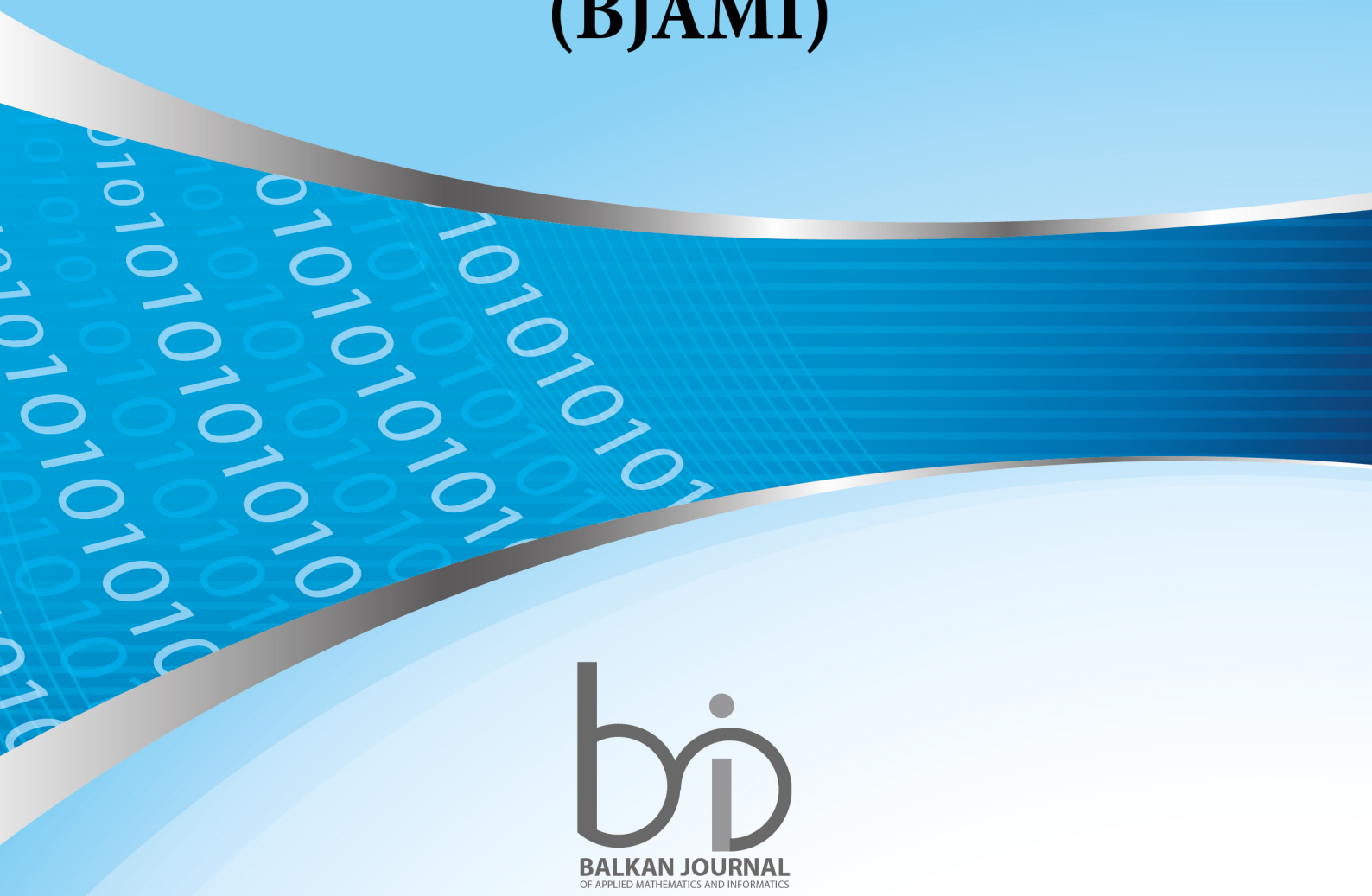
GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI)



YEAR 2021

VOLUME IV, Number 2

GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS

(BJAMI)

YEAR 2021 VOLUME IV, Number 2

AIMS AND SCOPE:
BJAMI publishes original research articles in the areas of applied mathematics and informatics.

Topics:
1. Computer science;
2. Computer and software engineering;
3. Information technology;
4. Computer security;
5. Electrical engineering;
6. Telecommunication;
7. Mathematics and its applications;
8. Articles of interdisciplinary of computer and information sciences with education, economics, environmental, health, and engineering.

Address of the editorial office
Goce Delcev University – Stip
Faculty of philology
Krste Misirkov 10-A
PO box 201, 2000 Štip,
Republic of North Macedonia

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI), Vol 3

ISSN 2545-4803 on line
Vol. 4, No. 1, Year 2021

EDITORIAL BOARD

CONTENT

UDC:004.652.4.021:511.14

ON THE NUMBER OF CANDIDATES IN APRIORI LIKE ALGORITHMS FOR MINING FREQUENT ITEMSETS

SAVO TOMOVIC

Abstract. Frequent itemset mining has been a focused theme in data mining research for years. It was first proposed for market basket analysis in the form of association rule mining. Since the first proposal of this new data mining task and its associated efficient mining algorithms, there have been hundreds of follow-up research publications. In this paper first we present the ideas from our previous work where we consider two problems from linear algebra, namely the set intersection problem and the scalar product problem, and make comparisons to the frequent itemset mining task. Then we formulate and prove new theorems that estimate the number of candidate itemsets that can be generated in the level-wise mining approach.

1. Introduction

We suggest a new model for frequent itemset mining which is based on linear algebra theory. In that way we give new mathematical foundation for frequent itemset mining task.

The idea comes from the definition of frequent itemset: frequent itemset is set of items that appear in sufficiently high number of transactions in given database [1], [2]. In linear algebra terminology this means that sufficiently high number of transactions have to intersect on particular itemset in order to make it frequent.

In this paper we developed the idea that had been initially presented in [26], in more detailed manner. In [26] we considered frequent itemsets, while in this paper we found upper bounds on the number of candidate itemsets in level wise mining approach. As in [26], we adopted results from set intersection and scalar product theory to estimate maximal number of candidate itemsets that can be generated in level-wise mining approach. To illustrate the idea we use the modification of well known Apriori algorithm [2] for frequent itemset mining, such called Apriori Multiple algorithm. Details of the Apriori Multiple algorithm are presented in [24], [25] and [29].

Date: November 13, 2021.

Keywords. frequent itemset mining, Apirori algorithm, scalar product, set intersection.

Results from this paper can be effectively applied to all algorithms that use level-wise mining approach [19].

2. Preliminaries

This section contains definitions that are necessary for further text. We primarily use notions from [23].

Suppose that I is a finite set; we refer to the elements of I as *items*.

Proposition 2.1. *A transaction data set on I is a function $T : \{1, ..., n\} \to P(I)$. The set $T(k)$ is the kth transaction of T. The numbers $\{1, ..., n\}$ are the transaction identifiers (TIDs).*

Given a transaction data set T on the set I, we would like to determine those subsets of I that occur often enough as values of T.

Proposition 2.2. *Let $T : \{1, ..., n\} \to P(I)$ be a transaction data set on a set of items I. The support count of a subset K of the set of items I in T is the number $suppcount_T(K)$ given by: $suppcount_T(K) = |\{t|1 \leq TID(t) \leq n \wedge K \subset T(t)\}|$. The support of an itemset K is the number: $support_T(K) = \frac{suppcount_T(K)}{n}$.*

Proposition 2.3. *An itemset K is μ-frequent relative to the transaction data set T if $support_T(K) \geq \mu$. We denote by F_T^μ the collection of all μ-frequent itemsets relative to the transaction data set T and by $F_{T,r}^\mu$ the collection of all μ-frequent itemsets that contain r items for $r \geq 1$.*

Note that $F_T^\mu = \bigcup_{r \geq 1} F_{T,r}^\mu$.

Proposition 2.4. *Frequent itemset mining problem consists of finding the set F_T^μ for given minimal support μ and transaction data set T.*

The following rather straightforward statement is fundamental for the study of frequent itemsets. It is known as Apriori principle.

Theorem 2.1. *Let $T : \{1, ..., n\} \to P(I)$ be a transaction data set on a set of items I. If K and K_1 are two itemsets, then $K_1 \subset K$ implies $support_T(K_1) \geq support_T(K)$.*

Proposition 2.5. *An association rule on an itemset I is a pair of nonempty disjoint itemsets (X, Y). An association rule (X, Y) is denoted by $X \to Y$. The confidence of $X \to Y$ is the number $conf_T(X \to Y) = \frac{support_T(XY)}{support_T(X)}$.*

3. Level-wise Frequent Itemset Mining

In this section we will briefly explain our Apriori Multiple algorithm for frequent itemsets mining from [24]. It is modification of well known Apriori algorithm from [1].

Apriori Multiple implements level-wise approach in frequent itemset mining [19]. The main characteristic of this approach is in the following: it generates frequent itemsets starting with frequent 1-itemsets (itemsets consisted of just one item); next, it iteratively generates frequent itemsets of size 2, 3, etc.

Iteration consists of two phases: candidate generation and support counting [11], [27].

In the candidate generation phase potentially frequent itemsets or candidate itemsets are generated. The Apriori principle is used in this phase (see Theorem 2.1). In this phase we have added new parameter named $multiple\_num$ that determines the "length" of iteration. Actually, in the original Apriori algorithm, in the iteration r, set $F^{\mu}_{T,r}$ is generated, while our Apriori Multiple algorithm in the iteration r generates sets $F^{\mu}_{T,i}, 0 \leq i < multiple\_num$. The parameters $multiple\_num$ and r are not connected.

The Apriori Multiple can use any value for $multiple\_num$ parameter. If $multiple\_num = 0$, our Apriori Multiple "becomes" the original Apriori algorithm. If we want to ensure that Apriori Multiple finishes in just two database scans, we need to choose value for $multiple\_num$ parameter such that $r_{max} < multiple\_num$ holds, where r_{max} is the maximal size of μ-frequent itemsets. The value r_{max} is not known in advance, but we can use the following very simple approach to find it. In the first scan, Apriori Multiple generates frequent 1-itemsets. During this scan Apriori Multiple can determine the length of the longest transaction in the database: t_{max}. It is clear that $r_{max} < t_{max}$, so the algorithm can set $multiple\_num$ parameter to t_{max}. Another approach is to set $multiple\_num$ parameter to average size of transactions. This does not guarantee that the algorithm finishes in two database scans, but it will generally finish in less database scans than the original Apriori algorithm [3]. Also, Apriori Multiple can start with some value for $multiple\_num$ parameter and change this value in the next iterations. The $multiple\_num$ parameter can also be defined by user, just like $\mu = minsup$ threshold. It means that user, according to domain knowledge or some other assessment can specify the value for $multiple\_num$ parameter.

The support counting phase consists of calculating support for all previously generated candidates (which are not pruned according to the Apriori principle in the preceding candidate generation phase). In the support counting phase, it is essential to efficiently determine if the candidates are contained in particular transaction $t \in T$ in order to increment their support. Because of that, many efficient structures and appropriate procedures for traversal were proposed in the literature [3]-[10], [12]-[18], [21], [28], [30]. In Apriori Multiple algorithm we organized candidates in special tree structure called TS-Tree. The candidates, which have enough support, are termed as frequent itemsets.

Pseudo-code for Apriori Multiple algorithm comes next.

The complexity of the Apriori like algorithms can be estimated as follows.

Apriori Multiple Algorithm

Require: database of transactions T, minimal support threshold μ

1: $F^{\mu}_{T,1}$ = frequent 1-itemsets in T
2: $multiple\_num$ = average transaction size
3: $r = 2$
4: **while** $F^{\mu}_{T,r-1} \neq \emptyset$ **do**
5: $C_r = candidate\_generation(F^{\mu}_{T,r-1})$
6: **for** i = 1 **to** $multiple\_num - 1$ **do**
7: $C_{r+i} = candidate\_generation(C_{r+i-1})$
8: **end for**
9: **for** i = 0 **to** $multiple\_num - 1$ **do**
10: $support\_count(C_{r+i})$
11: **end for**
12: **for** i = 0 **to** $multiple\_num - 1$ **do**
13: $F^{\mu}_{T,r+i} = \{c \in C_{r+i} | support(c) > \mu\}$
14: **end for**
15: $r = r + 1$
16: **end while**
17: **return** $F^{\mu}_{T} = \bigcup_{r \geq 1} F^{\mu}_{T,r}$

FIGURE 1. *Apriori Multiple algorithm*

- Generation of frequent l-itemsets. For each transaction, we need to update the support count for every item present in the transaction. Assuming that w is the average transaction width, this operation requires $O(Nw)$ time, where N is the total number of transactions.
- Candidate generation. To generate candidate k-itemsets, pairs of frequent $(k-l)$-itemsets are merged to determine whether they have at least $k-2$ items in common. Each merging operation requires at most $k-2$ equality comparisons. In the best-case scenario, every merging step produces a viable candidate k-itemset. In the worst-case scenario, the algorithm must

merge every pair of frequent $(k-1)$-itemsets found in the previous iteration. Therefore, the overall cost of merging frequent itemsets is

$$\sum_{k=2}^{w}(k-2)|C_k| \leq Costofmerging \leq \sum_{k=2}^{w}(k-2)|F_{k-1}|^2, \tag{3.1}$$

where w is the maximum transaction width.

- Support counting. Each transaction of length $|t|$ produces $\binom{|t|}{k}$ itemsets of size k. This is also the effective number of hash tree traversals performed for each transaction. The cost for support counting is $O(N\sum_k \binom{w}{k})$, where w the maximum transaction width.

4. Upper Bounds on the Number of Candidates in Apriori Like Algorithms

In this section we further develop the ideas from [26]. Because of completeness, let us briefly introduce set intersection problem and make comparison between it and frequent itemset mining.

Let N_n be a finite set that contains n different elements. We can consider that $N_n = \{1, 2, ..., n\}$. Let $M = \{M_1, M_2, ..., M_s\}$ be subset of power set of N_n that satisfies $|M_i| = k, i = 1, 2, ..., s$, where $1 \leq k \leq n$ is in advance given integer. We fix $t, 0 \leq t \leq k$ and define condition $|M_i \cap M_j| \neq t, i \neq j \in \{1, 2, ..., s\}$. With $m(n, k, t)$ we denote $max|M|$ where maximum is taken on all M that satisfies previously defined conditions.

In frequent itemset mining task terminology, set N_n is set of items I, while set M can be considered as collection of candidate k-itemsets in Apriori based algorithms.

With $f(n, r, t)$ denote the maximal number of candidate r-itemsets from set of items $I = \{i_1, ..., i_n\}$ such that any two itemsets intersect on $\geq t$ items. Notice that now we do not forbid itemsets to intersect on specific number of items; we force them to intersect on $\geq t$ items.

Theorem 4.1. *Let $2r - n < p, 1 \leq t \leq r \leq n$. Also, under the conditions $0 \leq i \leq (n-t)/2$ and $i \leq k - t$ define set*

$$F_i(n, r, t) = \{F \subset R^n : |F| = r, |F \cap \{1, 2, ..., t + 2i\}| \geq t + i\}. \tag{4.1}$$

In other words, the set $F_i(n, r, t)$ is collection of r-itemsets in I_n such that at least $t + i$ items is taken from $\{1, 2, ..., t + 2i\} \subset I_n$. If for some $w \in N \bigcup \{0\}$ holds

$$(r - t + 1)(2 + \frac{t-1}{w+1}) \leq n < (r - t + 1)(2 + \frac{t-1}{w}), \tag{4.2}$$

we have $f(n, r, t) = |F_w(n, r, t)|$.

In the following paragraphs, we analyze conditions from the Theorem 4.1 and formulate new theorems in order to make upper bounds on the number of candidate itemsets in Apriori Multiple algorithm or any algorithm that relays on level-wise mining approach.

Conditions from the previous theorem just ensure that sets $F_i(n,r,t)$ are defined correctly. First, consider $2r-n<t$. The opposite is $2r-n\geq t$ and in that case we have that r-itemsets of $\{1,2,...,n\}$ surely intersect on $\geq t$ elements which implies $f(n,r,t)=\binom{n}{r}$. This situation represents trivial case and it is not under consideration. Second, consider $i\leq(n-t)/2$ that can be easily transformed to $t+2i\leq n$. This condition ensures that $\{1,2,...,t+2i\}\subseteq\{1,2,...,n\}$. Finally, consider $i\leq r-t$ that is equivalent to $t+i\leq r$. The previous condition says that r-itemsets F for which $|F\cap\{1,2,...,t+2i\}|\geq t+i$ is true, definitely exists.

We can now formulate theorem that estimates number of candidate itemsets in Apriori Multiple algorithm from the previous section. Recall that in Apriori Multiple we have longer iterations, which means that in $t+1$th candidate generation phase algorithm generates the following $multiple\_num$ sets of candidate itemsets: $C_{t+1},C_{t+2},...,C_{t+multiple\_num}$. Notice that all these candidate itemsets intersect on $\geq t$ items.

Figure 2 illustrates the idea. For fast implementation of Apriori Multiple algorithm special tree structure is used. The tree is called TS-tree and it is based on Ryman set enumeration tree [1]. Each level in the tree contains candidate itemsets; the 0th level contains just the root that represents empty set, the first level contains the set C_1 of candidate 1-itemsets, the second level contains the set C_2 of candidate 2-itemsets, the level t contains the set C_t of candidate t-itemsets.

Theorem 4.2. *Consider $t+1$th candidate generation step in Apriori Multiple algorithm. Let $r=t+multiple\_num$. In Apriori Multiple algorithm it holds $|C_r|\leq F_0(n,r,t)$.*

Proof. Notice that the set C_r corresponds to the set of r-itemsets from I_n with property that any two itemsets from C_r intersect on $\geq t$ items. In other words $|C_r|$ can be estimated with $|f(n,r,t)|$. In order to prove our theorem we will apply result from Theorem 4.1.

In order to find w, we need w to satisfy the condition $w\leq r-t$. If we allow different, i.e. $w>r-t$ which is equivalent to $w\geq r-t+1$, we obtain

$$n<(r-t+1)(2+\frac{t-1}{w})\leq(r-t+1)(2+\frac{t-1}{r-t+1})\leq(r-t+1)\frac{2r-2t+2+t-1}{r-t+1}\leq 2r-t+1\leq n. \quad (4.3)$$

(4.

So, condition $w>r-t$ is not possible. We set $w=0\leq r-t=multiple\_num$ and check (4.2), i.e.

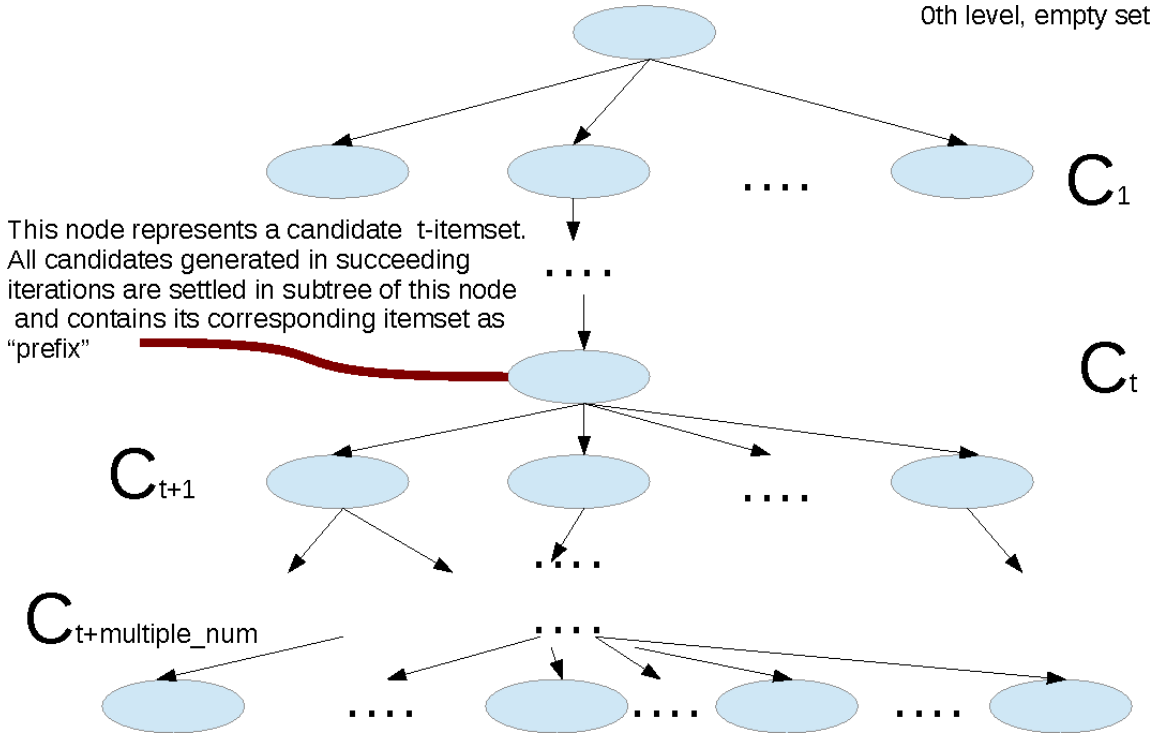


FIGURE 2. *TS-Tree example*

$$(r-t+1)(2+\frac{t-1}{0+1}) \leq n < (r-t+1)(2+\frac{t-1}{0}) \tag{4.4}$$

The inequalities in formula (4.4) are equivalent to

$$(r-t+1)(2+t-1) \leq n < (r-t+1)(2+\infty), \tag{4.5}$$

i.e.

$$(r-t+1)(t+1) \leq n < \infty. \tag{4.6}$$

Finally, we have

$$(multiple\_num+1)(t+1) \leq n < \infty. \tag{4.7}$$

Condition (4.7) is satisfied for sufficiently large n as in our case, because n is typically several thousands while $multiple\_num$ and t are much less. It can be shown that integer w that satisfies conditions from Theorem 4.1. is unique [20], so we use $w=0$, that implies $f(n,r,t)=F_0(n,r,t)$.

The set $F_0(n, r, t)$ is actually set of all r-itemsets in I_n such that any two of them intersect on $\{i_1, i_2, ..., i_t\}$. It is also illustrated on figure 2. □

Notice that $F_0(n, r, t) = \binom{n-t}{r-t}$. If we use estimation $\binom{n-t}{r-t} \sim (1.611... + o(1))^n$ from [20][pp. 23] we have $|C_r| \leq (1.611... + o(1))^n$.

The previous theorem can be reformulated in order to estimate number of candidate itemsets in any Apriori based algorithm that applies level-wise mining approach. Recall that in the iteration t Apriori based algorithms should generate set C_t consisting of candidate t-itemsets.

Theorem 4.3. *Consider tth candidate generation step in Apriori algorithm. Number of candidate itemsets in $r = t + xth$ step, where $x \geq 0$ is $|C_r| \leq F_0(n, r, t)$.*

Proof. The estimation directly follows from the result of the Theorem 4.2 when parameter $multiple\_num$ is changed with x. □

5. Experimental results

In this section experimental results are reported and discussed to demonstrate usability of presented methods. We performed a series of experiments on datasets that are known in the literature and used for similar analysis. These datasets are freely available. Some statistical characteristics are presented in Figure 3.

Dataset	Number of transactions, |T|	Number of items, |I|
mushroom	8124	119
T25I10D10K	9976	999
C20D10K	10000	386
C73D10K	10000	2178
pumbs*	49046	7116
connect	67557	129
T20I6D100K	99921	999
T10I4D100K	100000	999
T10I40D100K	100000	999
accidents	340183	468

Figure 3. *Datasets characteristics*

The task is to investigate how to reduce number of I/O operations, i.e. number of database passes. Time for I/O operations significantly contributes to whole execution time of Apriori like algorithms. It is illustrated in Figure 4. For example, for

Dataset	minsup	I/O in %
C20D10K	4000	15
C73D10K	9000	9
pumbs*	30000	56
connect	68000	43
T10I4D100K	1000	64
T20I6D100K	1000	26
T25I10D10K	150	31
T40I10D100K	1500	8

FIGURE 4. *Percentage of I/O operations in the execution time of Apriori algorithm*

the dataset T10I4D100K when *minsup* parameter is set to 1000 time for database readings is about 64% of the whole execution time.

We implemented Apriori Multiple algorithm with the procedure for estimating upper bounds for number of candidate itemsets. When the estimated number of candidates that must be generated is less then the buffer size, algorithm generates all of them and counts support in just one database reading. Results are presented in Figure 5. In all test cases time for I/O operations is significantly reduced.

Dataset	minsup	Time for I/O in Apriori (%)	Time for I/O in Apriori Multiple
C20D10K	4000	22	8
C73D10K	9000	9	3
pumbs*	30000	65	50
T10I4D100K	1000	64	60
T20I6D100K	1000	26	10
T25I10D10K	150	31	11

FIGURE 5. *Apriori Multiple with procedure for estimating a number of candidate itemsets*

6. FREQUENT ITEMSET MINING TASK IN VECTOR THEORY

In this section we give new mathematical formulation of frequent itemset mining problem that is based on scalar product of two vectors. After that we make estimation on the number of candidate itemsets in Apriori like algorithm as in the previous section.

Scalar product is an algebraic operation that takes two equal-length vectors and returns a single number. This operation can be defined as the sum of the products of the corresponding entries of the two vectors of numbers. Scalar product of two vectors $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ is defined as:

$$(x, y) = x_1y_1 + ... + x_ny_n \tag{6.1}$$

In the following paragraphs we will illustrate the modification of the original scalar product problem from [20][pp. 26] in order to make it more familiar with frequent itemset mining task.

Let n, k_0, k_1 be natural numbers that satisfy $k_0 + k_1 = n$. Consider the following set of vectors

$$\Sigma = \Sigma(\{0,1\}^n; k_0, k_1) =$$
$$= \{x = (x_1, ..., x_n) : x_i \in \{0,1\},$$
$$|i : x_i = 0| = k_0, |i : x_i = 1| = k_1\}.$$

Obviously, the set Σ is correctly defined and

$$|\Sigma| = Permutations(k_0, k_1) = \frac{n!}{k_0!\, k1!} \tag{6.2}$$

We fix $t \in N \cup \{0\}$ and want to estimate maximal size of any set $F \subset \Sigma$ in which there is no pair of vectors with scalar product t. Connection to the problem from the previous section is obvious.

We can now define the candidate itemsets maximal number estimation problem using scalar product theory. Let $I_n = \{i_1, ..., i_n\}$ be set of items. Any vector from Σ represents candidate $k_1 = r$-itemset because $(0,1)$-vector $(x_1, x_2, ..., x_n) \in \Sigma$ corresponds to the canidate r-itemset $c \in C_r$ in the way that $x_j = 1$ if $i_j \in c$ and $x_j = 0$ if $i_j \notin c$ and number of 1 is $k_1 = r$. At the same time vector product of vectors from Σ corresponds to size of intersection of any two candidate r-itemset from C_r.

In the next paragraph we formulate the theorem that is modification of the corresponding theorem from [20][Theorem 9, pp. 28].

Theorem 6.1. *Let p be prime number, $I = \{i_1, i_2, ..., i_n\}$ set of items and $T : \{1, 2, ..., w\} \to \mathcal{P}(I)$ transaction database on I. Additionally, let $n = 2p$, $p = r$. Maximal number of candidate itemsets $|C_r|$ in rth iteration of level-wise mining approach with property that any two candidate itemsets $c_i, c_j \in C_r$ do have at least one item in common is $\leq \Sigma_{(m_1,m_2)\in A}\binom{n}{m_1}\binom{n-m_1}{m_2}$, where $A = \{(m_1, m_2) : m_1, m_2 \in N_0, m_1 + m_2 \leq n, m_1 + 2m_2 \leq p - 1\}$.*

The previous theorem provides estimation of the maximal number of candidate itemsets that can be generated from one common item in rth iteration. That common item can be any node from the first level in TS-tree as it is illustrated in figure 2.

From [20] we will just take estimation $\Sigma_{(m_1,m_2)\in A}\binom{n}{m_1}\binom{n-m_1}{m_2} = (2.4628... + o(1))^n$ [20][pp. 32].

Notice that in Theorem 5.1 we define the set $F \subset \Sigma$ that contains vectors with property $k_0 = k_1 = p = n/2$ and there is no pair of orthonormal vectors (vectors with scalar product 0).

7. Conclusion

In this paper we present new mathematical model for frequent itemset mining problem. We use linear algebra method to estimate the size of candidate itemsets in Apriori based algorithms that implement level-wise approach in mining frequent itemsets.

Results from this paper can be effectively used in implementation of any level-wise algorithm because it gives upper bounds on the number of candidate itemsets, so we can know maximal memory requirements in advance.

As future work we plan to:

- much better estimation in Theorem 4.3
- incorporate minimal support parameter μ in order to achieve better estimation and define eventually dependency between μ and measures $m(n, r, p)$ or $f(n, r, t)$.

References

[1] *Agrawal, R.* & *Imielinski, T.* & *Swami, A. N.* (1993). Mining association rules between sets of items in large databases, Proceedings of the ACM International Conference on Management of Data, 207-216.

[2] *Agrawal, R.* & *Srikant, R.* (1994). Fast Algorithms for Mining Association Rules. IBM Almaden Research Center, San Jose CA 95120.

[3] *Ahmed, S.* & *Coenen, F.* & *Leng, P.* (2003). Strategies for Partitioning Data in Association Rule Mining. Research andDevelopment in Intelligent Systems XX, Springer, 127-140.

[4] *Ahmed, S.* & *Coenen, F.* & *Leng, P.* (2004). A Tree Partitioning Method for Memory Management in Association Rule Mining. Proceedings of the DaWaK 2004 conference, Lecture Notes in Computer Science, Vol.3181, 331-340.

[5] *Ahmed, S.* & *Coenen, F.* & *Leng, P.* (2006). Tree-based Partitioning of Data for Association Rule Mining. Knowledge and Information Systems, 10(3), 315-331.

[6] *Brin, S.* & *Motwani, R.* & *Ullman, J.D.* & *Tsur, S.* (1997) Dynamic Itemset Counting and Implication Rules for Market Basket data. Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, AZ, USA, 1997, 255-264.

[7] *Coenen, F.* & *Goulbourne, G.* & *Leng, P.H.* (2001). Computing Association Rules Using Partial Totals. In de Raedt, Principles of Data Mining and Knowledge Discovery, Proc PKDD, Spring Verlag Lecture Notes in Computer Science, 2001, Vol.2168, 54-66.

[8] *Coenen, F.* & *Goulbourne, G.* & *Leng, P.* (2004). Tree Structures for Mining Association Rules. Data Mining and Knowledge Discovery, Vol.8, No.1, 2004, 25-51.

[9] *Coenen, F.* & *Leng, P.* & *Ahmed, S.* (2003). T-Trees, Vertical Partitioning and Distributed Association Rule Mining. Proceedings of the Third IEEE International Conference on Data Mining (ICDM-2003), 2003, 513-516.

[10] *Coenen, F. & Leng, P. & Ahmed, S.* (2004). Data Structure for Association Rule Mining. T-trees and P-trees. IEEE Transactions on Knowledge and Data Engineering, Vol.16, No.6, 2004, 774-778.
[11] *Dunham, M.H.* (2003). Data Mining Introductory and Advanced Topics. Prentice Hall, New Jersey 2003.
[12] *El-Hajj, M. & Zaiane, O.R.* (2003). Non-Recursive Generation of Frequent K-itemsets from Frequent Pattern Tree Representations. Proceedings of the 5th International Conference (DaWaK 2003), Lecture Notes in Computer Science, Vol.2737, 2003, 371-380.
[13] *Goulbourne, G. & Coenen, F. & Leng, P.* (2000). Algorithms for Computing Association Rules Using a Partial-Support Tree. Knowledge-Based Systems, Vol.13, 2000, 141-149.
[14] *Grahne, G. & Zh, J.* (2003). Efficiently Using Prefix-trees in Mining Frequent Itemsets. Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2003.
[15] *Han, J. & Fu, Y.* (1995). Discovery of multiple-level association rules from large databases. Proceedings of 21th International Conference on Very Large Data Bases (VLDB'95), September 11-15, 1995, Zurich, Switzerland, Morgan Kaufmann, 1995, 420-431.
[16] *Han, J. & Pei, J. & Yu, Y.* (2000). Mining Frequent Patterns without Candidate Generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dalas, Texas, USA, 2000, 1-12.
[17] *Liu, G. & Lu, H. & Lou, W. & Yu, J.* (2003). On Computing, Storing and Querying Frequent Patterns. .Proceedings ofthe Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 24-27, 2003, Washington, DC, USA, ACM, 2003, 607-612.
[18] *Park, J.S. & Chen, M.S. & Yu, P.S.* (1995). An Effective HashBased Algorithm for Mining Association Rules. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, 1995, 175-186.
[19] *Piatetsky-Shapiro, G. & Frawley, W.J.* (1991). Knowledge Discovery in Databases. MIT Press, 1991.
[20] *Raigorodsky, A. M.* (2007). Linear algebra method in combinatorics, Izdatelstvo MCNMO.
[21] *Savasere, A. & Omiecinski, E. & Navathe, S.* (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. Proceedings of 21th International Conference on Very Large Data Bases (VLDB'95), September 11-15, 1995, Zurich, Switzerland, Morgan Kaufmann, 1995, 432-444.
[22] *Silberschatz, A. & Korth, H.F. & Sudarshan, S.* (2006). Database System Concepts. Mc Graw Hill, New York, 2006.
[23] *Simovici, D. A. & Djeraba, C.* (2008). Mathematical Tools for Data Mining - Set Theory, Partial Orders, Combinatorics, Springer.
[24] *Stanisic, P. & Tomovic, S.* (2008). Apriori Multiple Algorithm for Mining Association Rules, Information technology and control, 37(4), 311-320.
[25] *Stanisic, P. & Tomovic, S.* (2010). A New Rymon Tree Based Procedure for Mining Statistically Significant Frequent Itemsets, International Journal of Computers, Communication and Control, 5(4), 567-577.
[26] *Stanisic, P. & Tomovic, S.* (2013). Frequent Itemset Mining as Set Interection Problem, Proceedings of the 2nd Mediterranean Conference on Embedded Computing, 231-234.
[27] *Tan, P.N. & Steinbach, M. & Kumar, V.* (2006). Introduction to Data Mining. Addison Wesley, Boston, 2006.
[28] *Toivonen, H.* (1996). Sampling Large Databases for Association Rules. Proceedings of 22th International Conference on Very Large Data Bases (VLDB'96), September 3-6, 1996, Mumbai, India, Morgan Kaufmann, 1996, 134-145.

[29] *Tomovic, S. & Stanisic, P.* (2009). Mining the Most k-Frequent Itemsets with TS-Tree, Proceedings of the IADIS International Conference WWW/Internet, 606-612.

[30] *Zaki, M.J. & Parthasarathy, S. & Ogihara, M. & Li, W.* (1997). New algorithms for Fast Discovery of Association Rules. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), August 14-17, 1997, Newport Beach, USA, AAAI Press, 1997, 283-286.

University of Montenegro,
Faculty of Mathematics and Natural Sciences,
Podgorica,
Montenegro
Email address: savot@ucg.ac.me