OPEN ACCESS
Vietnam Journal of Computer Science
Vol. 8, No. 4 (2021) 1–15

DOI: 10.1142/S2196888821500214



# Efficient Mining of Non-Redundant Periodic Frequent Patterns

Michael Kofi Afriyie
*Computer Science and Engineering Department*
*University of Mines and Technology*
*P. O. Box 237, Tarkwa, Ghana*

Vincent Mwintieru Nofong*
*Computer Science and Engineering Department*
*University of Mines and Technology*
*P. O. Box 237, Tarkwa, Ghana*
*vnofong@umat.edu.gh*

John Wondoh
*University of South of Australia*
*Mawson Lakes Campus, Adelaide, Australia*

Hamidu Abdel-Fatao
*Computer Science and Engineering Department*
*University of Mines and Technology*
*P. O. Box 237, Tarkwa, Ghana*

Received 14 June 2020
Accepted 24 November 2020
Published 7 January 2021

Periodic frequent patterns are frequent patterns which occur at periodic intervals in databases. They are useful in decision making where event occurrence intervals are vital. Traditional algorithms for discovering periodic frequent patterns, however, often report a large number of such patterns, most of which are often redundant as their periodic occurrences can be derived from other periodic frequent patterns. Using such redundant periodic frequent patterns in decision making would often be detrimental, if not trivial. This paper addresses the challenge of eliminating redundant periodic frequent patterns by employing the concept of deduction rules in mining and reporting only the set of non-redundant periodic frequent patterns. It subsequently proposes and develops a Non-redundant Periodic Frequent Pattern Miner (NPFPM) to achieve this purpose. Experimental analysis on benchmark datasets shows that NPFPM is efficient and can effectively prune the set of redundant periodic frequent patterns.

*Keywords*: Frequent patterns; periodic frequent patterns; non-redundance.

*Corresponding author.

## 1. Introduction

Frequent pattern mining and their applications have been widely researched on over the past years. Several approaches and techniques have thus been proposed towards discovering categories of frequent patterns for domain specific applications. Such approaches and techniques can be found in Refs. 1–7.

Regardless of the usefulness of frequent patterns in revealing frequently occurring patterns in databases for decision making, they are incapable of revealing the occurrence shapes of patterns. A patterns' occurrence shape in a database comes in handy when decisions based on the periodicity of an event within the database are vital. For instance, in crime data analysis, frequently occurring crimes mined using any frequent pattern mining algorithm will not be able to reveal the occurrence shapes (periodic nature) of crimes in decision making. Revealing the occurrence shapes of crimes with time could be useful in decision making towards curbing future crimes. This usefulness of patterns' occurrence shapes in decision making resulted in research on discovering periodic frequent patterns.

Over the past years, periodic frequent pattern mining in transactional databases has been widely researched on. Several techniques and approaches have since been developed for discovering interesting categories of periodic frequent patterns (PFPs) in Refs. 8–13. Despite the numerous techniques and approaches proposed, one challenge which still exists is eliminating periodic frequent patterns that are redundant among those reported. This challenge exists as current approaches mostly discover and report a large number of periodic frequent patterns, most of which are often redundant since their periodic occurrences can be derived from the periodicities of their proper subsets. Mining and using these redundant periodic frequent patterns in decision making would not only consume memory, but could be detrimental if they are false positively periodic.

To address this challenge and eliminate redundant periodic frequent patterns during periodic frequent pattern discovery, this paper employs the concept of deduction rules in proposing and defining the set of non-redundant periodic frequent patterns. The proposed set of non-redundant periodic frequent patterns is not only devoid of redundant information, but their periodic occurrences cannot be inferred from other periodic frequent patterns.

The following are the main contributions of this paper in the discovery of periodic frequent patterns:

- It introduces the concept of non-redundant periodic frequent patterns (as the set of PFPs that are devoid of redundant information) which achieves a size reduction in the number of reported periodic frequent patterns.
- It proposes and develops NPFPM, an efficient algorithm for discovering the set of non-redundant periodic frequent patterns.

The rest of the paper is presented as follows. The preliminaries and related works are presented in Sec. 2, while Sec. 3 introduces the non-redundant periodic frequent

patterns. Section 4 presents the proposed approach while Sec. 5 presents the experimental analysis. Section 6 outlines the conclusions and future works.

## 2. Preliminaries

The problem of frequent itemset mining is given as follows. Let $I = \langle i_1, i_2, \ldots, i_n \rangle$ be a set of items. A transaction database is a set of transactions $D = \{T_1, T_2, T_3, \ldots, T_k\}$ such that for each transaction $T_a$, $T_a \in I$ and $T_a$ has a unique identifier $a$ called its transaction ID (TID). For example, consider the database in Table 1 (a sample customer transaction database — which will be used a running example), the set of items for this database become $I = \langle a, b, c, d, e, f \rangle$. Transaction $T_2$ which has three items $\{d, e, f\}$ is a length-3 itemset.

The *coverset* of an itemset $(S)$ in a database $(D)$ denoted as $\text{cov}_D(S)$ is defined as $\text{cov}_D(S) = \{m | m \in D \wedge S \subseteq m\}$. For example, in Table 1, given $S = \{a, b\}$, then $\text{cov}_D(S) = \{1, 3, 5\}$ since $\{a, b\}$ appears in transactions 1, 3 and 5. The *support count* of $S$ in $D$ is defined as $|\text{cov}_D(S)|$ and the *support* of $S$ in $D$, denoted as $\text{sup}_D(S)$ is defined as

$$\text{sup}_D(S) = \frac{|\text{cov}_D(S)|}{|D|}. \tag{1}$$

For instance, in Table 1, given $S = \{a, b\}$, then $\text{sup}_D(S) = \frac{3}{6} = 0.5$ since $|\text{cov}_D(S)| = |\{1, 3, 5\}| = 3$ and $|D| = 6$.

**Definition 2.1 (Frequent Itemset Mining).** The problem of frequent itemset mining consists of discovering frequent itemsets.[1] An itemset $S$ is a frequent itemset in a database $D$ if its support $\text{sup}_D(S)$ is not less than a user-specified minimum support threshold *minsup* given by the user.

**Definition 2.2 (Periods of an Itemset).** Let $D = \{T_1, T_2, T_3, \ldots, T_k\}$ be a database in which an itemset $S$ occurs with coverset $\text{cov}_D(S) = \{n_1, n_2, n_3, \ldots, n_{x-1}, n_x\}$, the periods of $S$ in $D$ denoted as $P^S$ is defined as $P^S = \{n_1 - 0, n_2 - n_1, n_3 - n_2, \ldots, n_x - n_{x-1}, |D| - n_x\}$.

For example, consider the itemset $\{a, b\}$ in Table 1 (where $|D| = 6$) which appears in transactions $T_1, T_3$, and $T_5$, with $\text{cov}_D\{a, b\} = 1, 3, 5$, the periods of $\{a, b\}$ will be $P^{(a,b)} = \{1 - 0, 3 - 1, 5 - 3, 6 - 5\} = \{1, 2, 2, 1\}$.

Table 1. Sample customer transactions.

| TID | Transaction |
|---|---|
| $T_1$ | $\{a, b, c\}$ |
| $T_2$ | $\{d, e, f\}$ |
| $T_3$ | $\{a, b, c\}$ |
| $T_4$ | $\{c, d, f\}$ |
| $T_5$ | $\{a, b, c, e, f\}$ |
| $T_6$ | $\{d, e\}$ |

Though various definitions have been proposed for periodic frequent patterns in Refs. 8, 13 and 14, we present the definition proposed in Ref. 11 since that is the periodic frequent pattern definition this paper adopts.

**Definition 2.3 (Periodic Frequent Pattern[11]).** Given a database **D**, minimum support threshold $\varepsilon$, periodicity threshold $p$, difference factor $p_1$, a pattern $S$ and $P^S$, $S$ is a periodic frequent pattern if $\sup_D(S) \geq \varepsilon$, $(p - p_1) \leq Prd(S) - std(P^S)$ and $Prd(S) + std(P^S) \leq (p + p_1)$,

where $Prd(S)$ (the mean of $P^S$, that is, $\bar{x}(P^S)$) is the periodicity of $S$ and $std(P^S)$ the standard deviation in $P^S$.

Nofong[11] further incorporated the productiveness measure (proposed in Ref. 15) together with Definition 2.3 in defining the productive periodic frequent patterns.

Fournier-Viger *et al.*[8] introduced PFPM, a periodic frequent pattern miner with novel pruning techniques. PFPM, unlike other existing periodic frequent pattern mining algorithms proposed in Refs. 11, 13 and 14, introduced the *minimum*, *maximum* and *average* periodicity measures for mining user-desired periodic frequent patterns.

Notwithstanding these propositions, as mentioned previously, the propositions in Refs. 8, 11, 13 and 14 works that employ these propositions (see Refs. 9, 12, 16–22) have challenges of reporting redundant periodic frequent patterns and difficulty in early termination during the process of mining periodic frequent patterns. To the best of our knowledge, there exists no work which addresses the issue of ensuring the set of redundant periodic frequent patterns are eliminated while only the set of non-redundant periodic frequent patterns are reported during periodic frequent pattern mining. This paper thus proposes and defines the non-redundant periodic frequent patterns towards ensuring only periodic frequent patterns without redundant information are being mined and reported.

## 3. Identifying Nonredundant Periodic Frequent Patterns

We adopt the periodic frequent pattern (Definition 2.3) proposed in Ref. 11.

With Definition 2.3, the set of periodic frequent patterns with similar periods will be reported. However, some periodic frequent patterns may be reported as periodic because their proper subsets are periodic. Such periodic frequent patterns which would most likely contain redundant information might be trivial if not detrimental[a] in decision making. To ensure only the set of non-redundant periodic frequent patterns is mined and reported, we employ the concept of frequent generators and define a non-redundant periodic frequent pattern as follows:

**Definition 3.1.** Given a periodic frequent pattern set, $Per_D = \{S_1, S_2, \ldots, S_j\}$, a periodic frequent pattern, $S_n$, is a non-redundant periodic frequent pattern if $\nexists S_u \in \mathrm{Per}_D$ such that, $S_u \subset S_n$ and, $\sup_D(S_n) = \sup_D(S_u)$.

[a] That is, they will be be detrimental in decision making if they happen to be false positively periodic.

We term the set of periodic frequent patterns in Definition 3.1 as non-redundant because their periodicities cannot be inferred or obtained from their subset periodic frequent patterns. The defined non-redundant periodic frequent patterns in Definition 3.1 based on the frequent generators are not same as closed periodic frequent patterns as generator frequent patterns are different from closed frequent patterns. That is, all proper subsets of the defined non-redundant periodic frequent patterns will also be non-redundant. In the case of closed periodic frequent patterns, some proper subsets may not be closed. Additionally, Definition 3.1 ensures only non-redundant periodic frequent patterns having same periodicities are reported.

Employing the concept of frequent generators in mining the defined non-redundant periodic frequent patterns have the following advantages:

- The ability to find an early termination mechanism during the discovery of periodic frequent patterns — due to the the anti-monotonic property of frequent generators.
- Mining and returning the set of periodic frequent patterns that do not contain redundant information and whose periodicities cannot be derived from other periodic frequent patterns.
- Reducing the number of "likely false positive" periodic frequent patterns.
- Discovering the set periodic frequent patterns that are more preferable in model selection.[23]

### 3.1. *Pruning redundant periodic frequent patterns*

To eliminate the set of redundant periodic frequent patterns, we employ the Redundance-Test function in identifying the set of non-redundant periodic frequent patterns. For a given periodic frequent pattern $S$, the Redundance-Test function tests if it is redundant or non-redundant as follows.

Line 1 of the Redundance-Test function creates the class of $S$ as null while Line 2 assigns $\Gamma$ as the set of all periodic frequent patterns. In Line 5, if the periodic frequent pattern $S$ has a length of one, and $\sup_D(S) = \sup(\emptyset)$ (where $\sup(\emptyset) = 1.0$), $S$ is classified as a redundant periodic frequent pattern, else, it is classified in Line 7 as a non-redundant periodic frequent pattern.

Given the periodic frequent pattern $S$ has a length more than one, in Line 10, it is classified as a redundant periodic frequent pattern provided its support can be obtained from any of its subsets that is also a periodic frequent pattern. That is, if there exists $S_l$ in $Per_D$ such that $S_l \subset S$ and $\sup_D(S_l) = \sup_D(S)$, then $S$ will be classified as a redundant periodic frequent pattern. If the conditions in Line 9 are not met, $S$ is classified as a non-redundant periodic frequent pattern in Line 12 as its periodicity cannot be inferred from its subsets that are also periodic. The Redundance-Test function in Line 13 returns the classification of $S$ as either redundant or non-redundant.

## 4. The Proposed Approach

To be able to mine the set of non-redundant periodic frequent patterns defined in Definition 3.1, we adopt and modify the PPFP algorithm proposed in Ref. 11. The Redundance-Test function previously discussed is incorporated into PPFP as NPFPM. NPFPM employs two steps in discovering the non-redundant PFPs:

- Obtaining the set of frequent length-1 items from the given database.
- Mining the set of non-redundant periodic frequent patterns from the frequent length-1 items obtained.

These two steps are shown in Algorithms 1 and 2, respectively. The operations and functions of these two algorithms do not differ much from the algorithms used in Ref. 11. To avoid repetition, we briefly summarize their operations (with the aid of Table 1 as a running example) and refer readers to Ref. 11 for further details.

Given dataset $D$, from which non-redundant periodic frequent patterns are to be mined based on the user desired thresholds, Lines 1–15 of Algorithm 1 identify the set of frequent length-1 items as per the user desired minimum support threshold. The set of frequent length-1 items identified is then sorted in Line 16 in descending order of items. From Table 1, for example, using a minimum support ($\varepsilon$) of 0.3, the set of frequent length-1 items identified and sorted in Line 16 of Algorithm 1 will be as shown in Table 2 (for illustration purposes, we indicate their coversets, sets of periods, mean period and standard deviations among the set of periods).

Line 17 of Algorithm 1 calls Algorithm 2 to mine the set of non-redundant periodic frequent patterns from the set of frequent length-1 items based on the user specified minimum support and periodicity thresholds.

**Function** Redundance-Test

**Input**: Set of periodic frequent patterns in $D$, $Per_D$ and a PFP, $S \in D$
**Output**: $S$.class $\in$ [Nonredundant, Redundant]

```
1  Create S.class = null
2  Let Γ = Per_D
3  if S is a length-1 pattern then
4  |  if sup_D(S) = sup(∅) then
5  |  |  S.class = Redundant
6  |  else
7  |  |  S.class = Nonredundant
8  else
9  |  if ∃S_l ∈ Per_D|S_l ⊂ S ∧ sup_D(S_l)=sup_D(S) then
10 |  |  S.class = Redundant
11 |  else
12 |  |  S.class = Nonredundant
13 return S.class
```

**Algorithm 1:** NPFPM($D, \varepsilon, p, p_1$)

**Input**: Dataset $D$, min. support $\varepsilon$, periodicity, $p$ and difference factor, $p_1$
**Output**: Non-redundant PFP set $Per_D$

```
1  Create HashMap h_n                    /* to store all length-1 items in D */
2  Create set L
3  for each transaction T ∈ D do
4  |  for each length-1 item a_y ∈ T do
5  |  |  if a_y ∉ h_n then
6  |  |  |  Create cov_D(a_y) = { TID of a_y}    /* TID = Transaction ID */
7  |  |  |  Add (a_y, cov_D(a_y)) to h_n
8  |  |  else
9  |  |  |  Let (a_y, cov_D(a_y)) = h_n(a_y)
10 |  |  |  Udate cov_D(a_y) as cov_D(a_y) = cov_D(a_y) ∪ TID of a_y
11 |  |  |  Update h_n with (a_y, cov_D(a_y))
12 for each item a_y ∈ h_n do
13 |  Let (a_y, cov_D(a_y)) = h_n(a_y)
14 |  if sup_D(a_y) ≥ ε then
15 |  |  Add (a_y, cov_D(a_y)) to L
16 Sort L in descending order of items
17 MinePFPs(L, ε, p, p_1)
18 return Per_D
```

For any set of frequent length-1 items, $L$, in Line 5, Algorithm 2 terminates and returns the set of non-redundant periodic frequent patterns if $|L| = 0$, that is, there are no frequent length-1 items in $L$. If there are frequent length-1 items however, Lines 7 to 26 of Algorithm 2 repeatedly mine the set of non-redundant periodic frequent patterns from $L$ until $|L| = 0$. For our running example, given a periodicity threshold ($p$) of 1.5 and a difference factor ($p_1$) of 0.5, during the first iteration, Line 13 will call the Redundance-Test function to eliminate redundant PFPs. Based on the Redundance-Test function, though all items in Table 2 are non-redundant, only $\{a\}$, $\{b\}$ and $\{f\}$ are periodic for the given periodicity thresholds ($p = 1.5$ and $p_1 = 0.5$). As such, only $\{a\}$, $\{b\}$ and $\{f\}$ will be added to the set of non-redundant periodic frequent items ($Per_D$) in Line 14 of Algorithm 2.

While in the first iteration, for the same Table 2, Lines 15–18 generate[b] the length-2 candidate frequent items from the length-1 frequent items. For each generated candidate frequent item, that passes the frequency and non-redundance test in Line 19, it is added to TempL in Line 20 and tested for periodicity in Line 23. If the pattern is periodic, it is added to the set of non-redundant periodic frequent patterns in Line 24. After the first iteration, the content in $L$ is replaced with that of TempL in

[b] Using the *A priori* candidate generation.

**Algorithm 2:** MinePFPs($L, \varepsilon, p, p_1$)

**Input**: Set $L$, periodicity, $p$, difference factor, $p_1$, and minimum support $\varepsilon$
**Output**: Non-redundant PFP set $Per_D$

```
1  Create Per_D
2  Create set TempL = ∅
3  Let P_{a_n}[0, b] be the the length-b prefix of a_n
4  if |L| = 0 then
5  |  return Per_D
6  else
7  |  while |L| > 0 do
8  |  |  for k = 0 to |L|-1 do
9  |  |  |  Let (a_k, cov_D(a_k)) = L[k]
10 |  |  |  if |a_k| = 1 then
11 |  |  |  |  Obtain P^{a_k} from e.cov_D(a_k)
12 |  |  |  |  Evaluate Prd(a_k) and std(P^{a_k}) from P^{a_k}
13 |  |  |  |  if a_k is periodic and non-redundant then
14 |  |  |  |  |  Add a_k to Per_D
15 |  |  |  for l = (k + 1) to |L|-1 do
16 |  |  |  |  Let (a_l, cov_D(a_l)) = L[l]
17 |  |  |  |  if P_{a_k}[0, |a_k|-1] = P_{a_l}[0, |a_l|-1] then
18 |  |  |  |  |  Create S = (a_k ∪ a_l, cov_D(a_k) ∩ cov_D(a_l))
19 |  |  |  |  if sup_D(S) ≥ ε and S is non-redundant then
20 |  |  |  |  |  Add S to TempL
21 |  |  |  |  |  Get P^S from e.cov_D(S)
22 |  |  |  |  |  Evaluate Prd(S) and std(P^S) from P^S
23 |  |  |  |  |  if S is periodic then
24 |  |  |  |  |  |  Add S to Per_D
25 |  |  L = TempL
26 |  |  TempL.clear()
27 return Per_D
```

Line 25, TempL is then cleared and the iteration repeats on $L$. The iteration repeats on $L$ and stops when $|L| = 0$. Line 27 of Algorithm 2 then returns the set of non-redundant periodic frequent patterns and the discovery process terminates.

For our running example, Table 3 shows the frequent length-2 itemsets (with their properties) generated from Table 2 during the first iteration. Though the frequent length-2 itemsets $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$ are all periodic based on the periodicity thresholds, they are all redundant since they fail the non-redundance test (that is, there exist their subset periodic frequent patterns ($\{a\}$ and $\{b\}$) that have same supports as $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$. As such, they will not be added to TempL in 20. This is because, based on the

Table 2. Frequent Length-1 Items from Table 1 at $\varepsilon = 0.3$

| Item (S) | Coverset ($cov_D(S)$) | Periods ($P^S$) | Mean Period ($Prd(S)$) | $std(P^S)$ |
|---|---|---|---|---|
| $\{a\}$ | $(1, 3, 5)$ | $(1, 2, 2, 1)$ | 1.5 | 0.5 |
| $\{b\}$ | $(1, 3, 5)$ | $(1, 2, 2, 1)$ | 1.5 | 0.5 |
| $\{c\}$ | $(1, 3, 4, 5)$ | $(1, 2, 1, 1, 1)$ | 1.2 | 0.4 |
| $\{d\}$ | $(2, 4, 6)$ | $(2, 2, 2, 0)$ | 1.5 | 0.866 |
| $\{e\}$ | $(2, 5, 6)$ | $(2, 3, 1, 0)$ | 1.5 | 1.118 |
| $\{f\}$ | $(2, 4, 5)$ | $(2, 2, 1, 1)$ | 1.5 | 0.5 |

Table 3. Frequent Length-2 Itemsets Obtained from Table 2 at $\varepsilon = 0.3$.

| Itemset (S) | $cov_D(S)$ | $P^S$ | $Prd(S)$ | $std(P^S)$ |
|---|---|---|---|---|
| $\{d, e\}$ | $(2, 6)$ | $(2, 4, 0)$ | 2 | 1.633 |

anti-monotone property of generators, the frequent supersets formed from $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$ will be non-generators (that is, their supports can be inferred from their generator subsets). This pruning thus enables our approach achieve an early termination during the PFP mining process. As such, no length-2 periodic frequent pattern from our example will be added to $Per_D$ since the generated pattern $\{d, e\}$ is not periodic.

The iteration repeats on Table 3 with no candidate length-3 frequent itemsets generated. The non-redundant periodic frequent pattern mining thus terminates since $|L| = 0$. Line 27 of Algorithm 2 thus returns only $\{a\}$, $\{b\}$ and $\{f\}$ as the set of non-redundant periodic frequent patterns. Assuming $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$ were not pruned out and left in Table 3, the only length-3 candidate frequent pattern (that is, $\{a, b, c\}$) which would have been generated, though frequent and periodic, would have failed the non-redundance test and hence not added to $Per_D$.

## 5. Experimental Analysis

The following implementations were used in our experimental analysis:

- NPFPM: An implementation of the proposed non-redundant periodic frequent pattern mining algorithm. Given any dataset and the desired user thresholds, NPFPM mines and returns the set of all non-redundant periodic frequent patterns.
- PFP*: An implementation for detecting the set of all periodic frequent patterns with similar periodicities based on Definition 2.3. PFP* does not employ the productiveness or non-redundance measure. As such, PFP* mines and returns all periodic frequent patterns based on the user specified thresholds. PFP* used in this comparison will be faster and more efficient than the naive approach — where all periodic frequent patterns will be mined and a post-processing approach employed to identify the non-redundant periodic frequent patterns.

Experimental analysis was conducted with regard to (i) runtime performance, (ii) reported patterns, and, (iii) scalability. The following datasets[c] were used for the experimental analysis: (a) Kosarak10K dataset — this is a partly dense dataset with 10,000 transactions; (b) Kosark45K — this is also a partly dense dataset with 45,000 transactions, (c) Tafeng Nov. 2000 — this is a very sparse dataset with 31,807 transactions from customers at the Tafeng retail store for the month of November 2000, and, (d) T10I4D100K — a dense dataset with 100,000 transactions.

### 5.1. *Runtime performance: Periodic frequent pattern discovery*

The runtime performance of the two compared implementions is shown in Figs. 1–3 on the Kosarak10K, Kosarak45k and Tafeng datasets, respectively. As can be observed, NPFPM in Figs. 1–3 is more efficient in discovering periodic frequent patterns compared to PFP*.

It is also worth noting that the runtimes of the two implementation as depicted in Fig. 3 are almost the same. This is due to the sparse nature of the Tafeng dataset which results in both implementations (for the specified thresholds) reporting the same number of periodic frequent patterns (see Table 6 for the number of reported periodic frequent patterns).

### 5.2. *Reported patterns: Periodic frequent pattern discovery*

The number of reported periodic frequent patterns for the two algorithms (NPFPM and PFP*) on the described datasets is as shown in Tables 4–6.

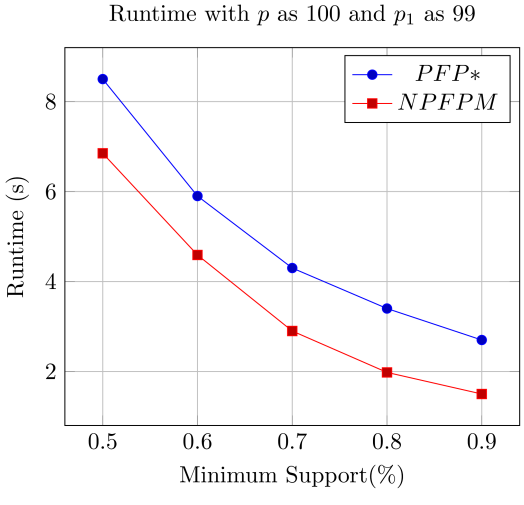


Fig. 1. Periodic frequent pattern discovery: Runtime in Kosarak10K dataset.

[c]The Korsarak10K and Kosarak45K datasets were obtained from Ref. 24, the Tafeng dataset was obtained from the AIIA Lab, and, the T10I4D100K dataset was obtained from the FIMI repository.

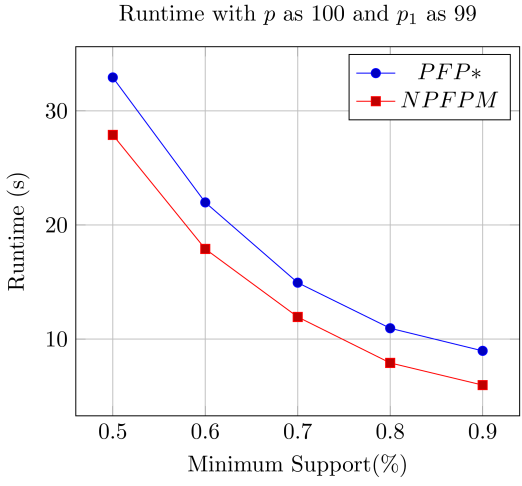


Fig. 2. Periodic frequent pattern discovery: Runtime in Kosarak45K dataset.

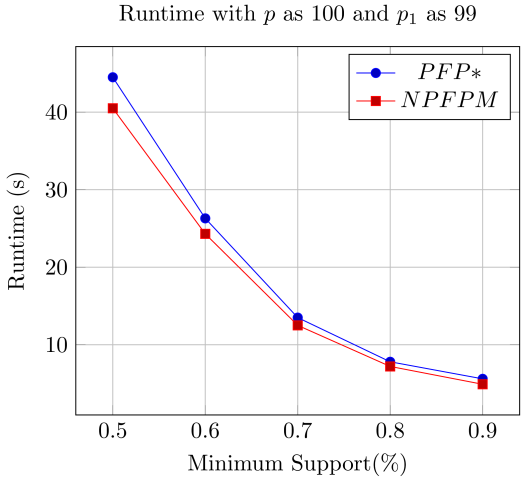


Fig. 3. Periodic frequent pattern discovery: Runtime in Tafeng Nov 2000 dataset.

It was observed that in the sparse dataset (that is, the Tafeng dataset — see Table 6), both approaches report the same number of periodic frequent patterns with near similar runtime during the discovery process (see Fig. 3). In the Kosarak10K and Kosarak45K datasets, NPFPM, however, reports a smaller number of periodic frequent patterns compared to PFP*.

As can be observed in Tables 4 and 5, with the non-redundance measure, NPFPM is able to prune the set of periodic frequent patterns with redundant information and hence, report a much smaller set of periodic frequent patterns compared to PFP*.

Table 4. Reported periodic frequent patterns in Kosarak10K dataset.

| $\varepsilon$ | NPFPM | PFP* |
|---|---|---|
| | $p = 100$ | $p = 100$ |
| | $p_1 = 99$ | $p_1 = 99$ |
| 0.8% | 114 | 210 |
| 0.7% | 114 | 210 |

Table 5. Reported periodic frequent patterns in Kosarak45K dataset.

| $\varepsilon$ | NPFPM | PFP* |
|---|---|---|
| | $p = 100$ | $p = 100$ |
| | $p_1 = 99$ | $p_1 = 99$ |
| 0.8% | 173 | 188 |
| 0.7% | 173 | 188 |

Table 6. Reported periodic frequent patterns in Tafeng Nov 2000 dataset.

| $\varepsilon$ | NPFPM | PFP* |
|---|---|---|
| | $p = 130$ | $p = 130$ |
| | $p_1 = 120$ | $p_1 = 120$ |
| 0.8% | 20 | 20 |
| 0.7% | 26 | 26 |

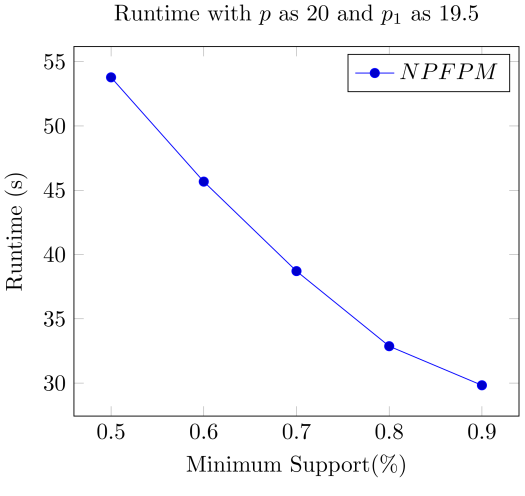


Fig. 4. Periodic frequent pattern discovery: Runtime in T10I4D100K dataset.

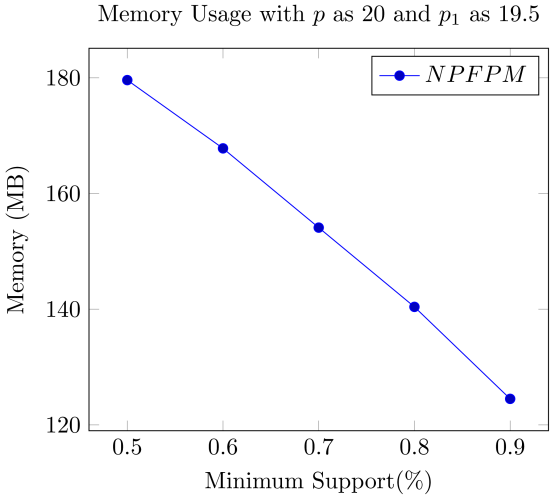


Fig. 5. Periodic frequent pattern discovery: Memory usage in T10I4D100K dataset.

### 5.3. *Scalability test: NPFPM*

Experiments on the scalability of NPFPM in large datasets were performed using the T10I4D100K dataset which has 100,000 transactions in total. The minimum support for this test was chosen from 0.005 to 0.009, that is, from 0.5% to 0.9% of the T10I4D100K dataset.

Figure 4 shows the runtime scalability test on NPFPM while Fig. 5 shows the memory used by NPFPM in discovering periodic frequent patterns in the T10I4D100K dataset. From Figs. 4 and 5, NPFPM is scalable in even at low minimum supports in large datasets.

As can be observed in Figs. 4 and 5, when the minimum support is 0.005 (that is, considering a pattern appearing 500 times out of the 100,000 transactions as frequent), NPFPM takes 53.78 s and 179.6 MB of memory to find all non-redundant periodic frequent patterns.

## 6. Conclusion

Non-redundant periodic frequent patterns are the set of periodic frequent patterns whose periodic occurrence cannot be inferred from their subset periodic frequent patterns. This work employs the concept of deduction rules in identifying the set of non-redundant periodic frequent patterns. Subsequently, a NPFPM is proposed and developed for mining the set of non-redundant periodic frequent patterns. Experimental results on benchmark datasets show that NPFPM is efficient and reports a smaller set of non-redundant periodic frequent patterns compared to the set of all periodic frequent patterns. In our future works, we will investigate on measures that can be employed in memory efficient mining of interesting periodic frequent patterns.

## Acknowledgment

This paper is an extended version of our paper presented at ACIIDS 2020 and included in the online proceedings [https://link.springer.com/chapter/10.1007/978-3-030-41964-6_28].



## References

1. R. Agrawal, T. Imieliński and A. Swami, Mining association rules between sets of items in large databases, *SIGMOD Rec.* **22**(2) (1993) 207–216.
2. J. Han, J. Pei and Y. Yin, Mining frequent patterns without candidate generation, *ACM SIGMOD Rec.* **29**(2) (2000) 1–12.
3. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang and D. Yang, H-mine: Hyper-structure mining of frequent patterns in large databases, *Proc. IEEE Int. Conf. Data Mining*, San Jose, CA, November 2001, pp. 441–448.
4. F. C. Tseng, Mining frequent itemsets in large databases: The hierarchical partitioning approach, *Expert Syst. Appl.* **40**(5) (2013) 1654–1661.
5. M. J. Zaki, Scalable algorithms for association mining, *IEEE Trans. Knowl. Data Eng.* **12**(3) (2000) 372–390.
6. M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li, Parallel algorithms for discovery of association rules, *Data Min. Knowl. Discov.* **1**(4) (1997) 343–373.
7. M. J. Zaki and K. Gouda, Fast vertical mining using diffsets, *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, August 2003, pp. 326–335.
8. P. Fournier-Viger, C. W. Lin, Q. H. Duong, T. L. Dam, L. Ševčík, D. Uhrin and M. Voznak, PFPM: Discovering periodic frequent patterns with novel periodicity measures, *Proc. 2nd Czech-China Scientific Conf.*, Ostrava, Czech Republic, June 2016.
9. R. U. Kiran and P. K. Reddy, Towards efficient mining of periodic-frequent patterns in transactional databases, in *LNCS*, eds. P. G. Bringas, A. Hameurlain and G. Quirchmayr, Vol. 6262 (Springer, Heidelberg, 2010), pp. 194–208.
10. R. U. Kiran and M. Kitsuregawa, Discovering quasi-periodic-frequent patterns in transactional databases, in *LNCS*, eds. V. Bhatnagar and S. Srinivasa, Vol. 8302 (Springer International Publishing, 2013), pp. 9–115.
11. V. M. Nofong, Discovering productive periodic frequent patterns in transactional databases, *Ann. Data Sci.* **3**(3) (2016) 235–249.
12. A. Surana, R. U. Kiran and P. K. Reddy, An efficient approach to mine periodic-frequent patterns in transactional databases, in *LNAI*, eds. L. Cao, J. Z. Huang, J. Bailey, Y. S. Koh and J. Luo, Vol. 7104 (Springer, Heidelberg, 2012), pp. 254–266.
13. S. K. Tanbeer, C. F. Ahmed, B. S. Jeong and Y. K. Lee, Discovering periodic-frequent patterns in transactional databases, in *LNAI*, eds. T. Theeramunkong, B. Kijsirikul, N. Cercone and T. Ho, Vol. 5476 (Springer, Heidelberg, 2009), pp. 242–253.
14. M. M. Rashid, M. R. Karim, B. S. Jeong and H. J. Choi, Efficient mining regularly frequent patterns in transactional databases, in *LNCS*, eds. S. Lee, Z. Peng, X. Zhou, Y. Moon, R. Unland and J. Yoo, Vol. 7238 (Springer, Heidelberg, 2012), pp. 258–271.
15. G. I. Webb, Self-sufficient itemsets: An approach to screening potentially interesting associations between items, *ACM Trans. Knowl. Discov. Data* **4**(1) (2010) 3:1–3:20.
16. W. N. Ismail, M. M. Hassan, H. A. Alsalamah and G. Fortino, Mining productive-periodic frequent patterns in tele-health systems, *J. Netw. Comput. Appl.* **115** (2018) 33–47.
17. R. U. Kiran and M. Kitsuregawa, Novel techniques to reduce search space in periodic-frequent pattern mining, in *LNCS*, eds. S. S. Bhowmick, C. E. Dyreson, C. S. Jensen,

M. L. Lee, A. Muliantara, B. Thalheim, Vol. 8422 (Springer International Publishing, 2014), pp. 377–391.
18. R. U. Kiran and P. K. Reddy, An alternative interestingness measure for mining periodic-frequent patterns, in *LNCS*, eds. J. X. Yu, M. H. Kim and R. Unland, Vol. 6587 (Springer, Heidelberg, 2011), pp. 183–192.
19. V. Kumar and V. Valli Kumari, Incremental mining for regular frequent patterns in vertical format, *Int. J. Eng.Tech.* **5**(2) (2013) 1506–1511.
20. V. M. Nofong, Fast and memory efficient mining of periodic frequent patterns, in *Modern Approaches for Intelligent Information and Database Systems, SCI*, eds. A. Sieminski, A. Kozierkiewicz, M. Nunez and Q. T. Ha, Vol. 769 (Springer, Cham, 2018), pp. 223–232.
21. V. M. Nofong and J. Wondoh, Towards fast and memory efficient mining of periodic frequent patterns, *J. Telecommun. Inf. Technol.* **3**(4) (2019) 480–493.
22. M. M. Rashid, I. Gondal and J. Kamruzzaman, Regularly frequent patterns mining from sensor data stream, in *LNCS*, eds. M. Lee, A. Hirose, Z. G. Hou and R. Kil, Vol. 8227 (Springer, Berlin, Heidelberg, 2013), pp. 417–424.
23. J. Li, H. Li, L. Wong, J. Pei and G. Dong, Minimum description length principle: Generators are preferable to closed patterns, *Proc. 21st National Conf. Artificial Intelligence* (Boston, MA, July 2006), pp. 409–414.
24. P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu and V. S. Tseng, SPMF: A Java open-source pattern mining library, *J. Mach. Learn. Res.* **15** (2014) 3389–3393.