An Improved Framework for Mining Periodic Patterns

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Vipul Chhabra
2019121001
vipul.chhabra@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "An Improved Framework for Mining Periodic Patterns" by Vipul Chhabra, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. P. Krishna Reddy

To my parents for their endless support and unconditional love.

Acknowledgments

I am immensely grateful for my decision to join the Data Science and Analytics Lab (DSAC) a few years ago. I owe a heartfelt thanks to my esteemed supervisors, Prof. P. Krishna Reddy and Prof. Rage Uday Kiran from the University of Aizu, for introducing me to research and providing invaluable supervision and support throughout my journey. Their constant guidance has helped me evolve both professionally and personally, and I cannot thank them enough.

I am also indebted to my co-authors, Prof. Minh-Son Dao, Prof. Koji Zettsu, and Saideep Chennupati, for their unwavering support and for providing me with the resources and help to guide my research. I am grateful for their contributions and the opportunity to work with them.

I want to express my gratitude to all the professors at IIIT-Hyderabad for broadening my knowledge of the domain. In particular, I thank Prof. P. Krishna Reddy, Prof. Kamalakar Karlapalem, and Prof. Deepak Gangadharan for providing me with the opportunity to work with them and teaching me things that go beyond academics or research.

I also want to thank my lab mates, Abinash Maharana, A Srinivas Reddy, Haranadh Pokala, Narendra Babu Unnam, and Sriharshita Bondogula, for helping me brainstorm new ideas and with their domain expertise. Additionally, I am thankful to my friends, Aaryamaan Jain, Akshat Goyal, Arohi Shrivastava, Avani Gupta, Bhavyajeet Singh, Dolton Fernandes, Gaurang Tandon, Jalees Jahanzaib, Mayank Goyal, and Yash Bhansali, for making my college life more enjoyable and being supportive throughout.

Finally, I thank my family, especially my grandparents and parents. My grandfather and grandmother are my role models, and I am grateful for all their sacrifices. My mom has always supported and motivated me to achieve big in life, and my dad taught me the values of hard work, patience, and ethics. They never failed to put their dreams and passions aside to uplift and fulfil my and my brother's dreams, and I owe them all my present and future achievements. Thanks to my younger brother for his constant encouragement and for making me more confident. Lastly, I would like to thank all the people who have played a crucial role in my journey but whom I could not name here due to space constraints.

Abstract

Data mining is a collection of algorithms for extracting valuable insights from large amounts of data. Data mining based approaches are being employed to improve the performance of decision support systems in the fields of customer relationship management, inventory management, fraud detection, surveillance, and recommendation systems. Pattern mining is an important task of data mining, which involves identifying significant associations in transactional databases. These associations reveal valuable trends and relationships that aid businesses in improving efficiency. The field of pattern mining has started with the model to extract frequent patterns from large transactional data. The model of frequent patterns has become popular and resulted in the development of algorithms to improve the performance of several applications, like recommendation systems. In the literature, encouraged by the potential of pattern mining, active research is going on to investigate new pattern mining models such as periodic, utility, coverage, and correlated patterns. In this thesis, we propose an improved approach to mine periodic patterns from temporal databases.

The model of periodic patterns facilitates the discovery of recurring behaviours and trends in the temporal databases. The model of periodic pattern, which we call partial periodic pattern ($3P$), captures periodic associations subject to periodic support (PS) and inter-arrival time (IAT) constraints. Here, the IAT value is the time difference between the successive occurrence of a pattern. The percentage of occurrences of a pattern in a database that satisfies the user-given maximum IAT constraint is called PS. Overall, all patterns that satisfy user given maximum IAT ($maxIAT$) and minimum PS ($minPS$) are called $3Ps$.

In this thesis, we address the following issue of the $3Ps$ model. It was observed that if we set the $minPS$ value too low, the number of 3Ps explodes. On the other hand, if set $minPS$ value high, several interesting 3Ps will be missed. We call this problem as a rare item problem. To address this issue, in this thesis, we have proposed an improved model and present an improved depth-first search algorithm to extract 3Ps. In the proposed model, we introduce a new measure called $periodic\text{-}confidence$, which satisfies both $null\text{-}invariant$ and $anti\text{-}monotonic$ properties. We also propose a better depth-first search algorithm to mine 3Ps. The proposed algorithm uses $irregularity\ pruning$ to reduce the search space and computational cost while maintaining the same amount of information. Through experimental

results, we show that the proposed algorithm is efficient and scalable. We illustrate the usefulness of the discovered patterns through case studies on air pollution and traffic congestion databases.

Extracting interesting trends from large temporal databases in different domains is an active research area. We hope that the proposed approach could facilitate the development of improved approaches to extract regular trends for fraud detection from sensor (surveillance) databases and loyalty mining in e-commerce systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the internet era, several applications are generating big data. Crucial information that can provide end-users with competitive knowledge to achieve socio-economic development lies hidden in this data. The field of data mining [1] has emerged to discover the beneficial information hidden in big data. The three important data mining tasks are *clustering*, *classification*, and *pattern mining*. Clustering involves organizing a large dataset into groups or clusters of similar objects, where the similarity can be based on distance or some other similarity metric. Classification, on the other hand, involves assigning objects to pre-defined categories based on their features or attributes. Pattern mining aims to discover user interest-based patterns hidden in big data. This thesis focuses on pattern mining, which has many practical applications, ranging from market basket analysis [2] to bioinformatics [3] and social network analysis [4].

Agrawal et al. [5] first introduced frequent pattern mining as a key step to discover interesting associations that exist between the items in a transactional database (TDB). Since then, the problem of finding frequent patterns has received a great deal of attention. Moreover, frequent pattern mining was extended to find a range of user interest-based patterns, such as closed frequent patterns [6], maximal frequent patterns [7], frequent sequential patterns [8], uncertain frequent patterns [9], fuzzy frequent patterns [10], and utility patterns [11]. However, the widespread adoption of frequent pattern mining has been hindered by the following obstacle: "*The frequent pattern mining technique implicitly assumes that the temporal occurrence information of the items, if any, will not play any role in determining the interestingness of the pattern in a database. However, this is seldom not the case as the user may consider a pattern occurring periodically to be more interesting than a pattern that is occurring irregularly in the data. Consequently, the frequent pattern mining technique fails to discover those periodically occurring interesting patterns that may exist in a temporal database*." [12] When confronted with this problem in real-world applications, researchers initially extended the frequent pattern mining technique to discover those frequent patterns that have exhibited full (or perfect) periodic behaviour in a database [13, 14, 15, 16]. A key limitation of this extended frequent pattern mining technique is that it fails to discover those interesting patterns that have exhibited partial periodic behaviour in the data. In the liter-

ature, few studies [17, 18, 19] described various approaches to finding partial periodic frequent patterns in a temporal database. Unfortunately, these approaches were impracticable due to their high computational cost as the generated patterns do not satisfy the *downward closure property*[1] [5]. Recently, Uday et al. [20] described a practicable model of partial periodic patterns (3Ps) that may exist in a temporal database. The 3Ps generated from a database satisfy the *downward closure property*. Consequently, this model was extended to find various types of interesting patterns, such as stable periodic-frequent pattern mining [21], fuzzy periodic-frequent pattern mining [22], and Geo-referenced periodic-frequent pattern mining [23, 24]. Notably, these knowledge discovery techniques are meant for real-world applications. However, most of the existing works are based on the assumption of an ideal database where all the items occur with a similar frequency, which is not valid for most real-world applications. Therefore, research efforts are being made to develop scalable algorithms for mining partial periodic patterns from large real-world temporal databases that can generate patterns containing both frequent and rare items. (Please note that classifying items into frequent or rare is a subjective issue that depends on the user and/or application requirements.)

In this thesis, we propose a novel framework for mining partial periodic patterns in non-uniform temporal databases that is highly scalable and efficient.

The rest of the chapter is organized as follows. In the next section, as a part of the background, we introduce transactional and temporal databases and briefly explain frequent patterns, periodic patterns, partial periodic patterns, and the research gap. Then, we briefly summarize the proposed model for mining 3Ps in non-uniform databases. Subsequently, we provide a list of contributions and explain the organization of the thesis.

1.1 Background

In this section, we provide an introduction to transactional and temporal databases, followed by a brief explanation of frequent patterns, periodic patterns, and partial periodic patterns.

1.1.1 Transactional Database (TDB)

A transactional database contains a collection of transactions, each comprising a set of items related to a specific event. For instance, a transaction in an online shopping platform might represent a customer's purchase of items in a single order. Similarly, in a healthcare system, a transaction could correspond to a patient's visit to the hospital.

Transactional databases are valuable for businesses and organizations as they contain information that can be used to identify interesting patterns and relationships. However, the sheer volume of data

[1]The downward closure property says that all non-empty subsets of an interesting pattern must also be interesting patterns

makes it challenging to extract meaningful insights manually. Therefore, various data mining techniques have been developed to automate the process of discovering useful information from these databases.

For the purpose of illustration, consider the following hypothetical transactional database shown in Table 1.1. This database contains a collection of transactions, where each transaction represents a set of items (represented by english letters) related to a specific event. Each transaction can be uniquely identified by a transaction id (tid). For instance, the transaction with tid =1 contains the items p, r, and s.

Table 1.1: Transactional database

tid	items
1	prs
2	$pqrt$
3	$pqrs$
4	stu

tid	items
5	stv
6	psv
7	psu
8	qrs

tid	items
9	psu
10	pt
11	$pqru$
12	$pqrs$

tid	items
13	$pqrv$
14	$pqrs$

1.1.2 Temporal Database

Useful information that can facilitate domain experts in gaining a competitive advantage lies hidden in this database as the time dimension becomes a critical factor in extracting valuable insights from this data. The temporal information in the data can help identify important patterns that may have been overlooked otherwise as it captures and analyzes what happened and when.

To illustrate the concept of temporal databases, consider the hypothetical dataset depicted in Table 1.2. This database consists of a set of transactions, each of which is associated with a timestamp (ts) indicating when the transaction has occurred.

Table 1.2: Temporal database

tid	ts	items
1	1	prs
2	3	$pqrt$
3	4	$pqrs$
4	5	stu

tid	ts	items
5	6	stv
6	7	psv
7	8	psu
8	9	qrs

tid	ts	items
9	10	psu
10	11	pt
11	12	$pqru$
12	13	$pqrs$

tid	ts	items
13	15	$pqrv$
14	16	$pqrs$

1.1.3 Frequent Patterns

Frequent patterns, also known as itemsets, represent an important category of regularities present within a TDB [25]. In a TDB, a frequent pattern is a group of items that satisfies the support constraint.

The support of a pattern refers to the percentage of transactions within the TDB in which that pattern occurs. If a pattern's support is at least the user-defined minimum support threshold, it is categorized as a frequent pattern. Frequent patterns are useful in revealing the associations between items that frequently co-occur within the TDB. An example of a frequent pattern found in Table 1.1 is as follows:

$$\{p, r\} \quad [support = 50\%].$$

This pattern indicates that the items p and r have together appeared in 50% of the transactions.

In the literature, several methods have been proposed for extracting frequent patterns from a TDB based on the user-given minimum support. These methods include Apriori [5], FP-Growth (Frequent Pattern Growth) [26] and ECLAT (Equivalence Class Clustering and Bottom-Up Lattice Traversal) [27]. ECLAT is efficient in space and appropriate for medium-sized data sets, while FP-Growth is suitable for large and dense data sets. It is worth noting that the frequent pattern model is one among the different types of pattern extraction models proposed in the literature. Each model captures a distinct type of association among the items in a pattern. For example, maximal frequent patterns [28], top-k patterns [29], periodic patterns [30], coverage patterns [31], and utility patterns [32] are some other pattern extraction models proposed in the literature.

1.1.4 Periodic Patterns

Periodic pattern mining is a data mining technique that extracts patterns from a temporal database that occur periodically. Unlike traditional frequent pattern mining, which ignores the time dimension of the database, periodic pattern mining takes into account the time or order in which transactions occur. The periodic pattern mining algorithm extracts patterns that occur regularly and repeatedly throughout the temporal database. A pattern is considered periodic if it occurs with a periodicity that does not exceed a specified maximum period and if the frequency of its occurrence exceeds a given support threshold. The maximum period is a user-given threshold value that limits the duration between two occurrences of the pattern in the database. By identifying frequent and periodic patterns in the data, we can better understand the behaviour of the underlying processes and use that information to improve decision-making and forecasting. One potential application is analyzing the customers' shopping behaviour in a supermarket. The dataset records the items each customer purchased and the time of the transaction. An example of a periodic pattern is as follows:

Example 1. *Considering the hypothetical temporal database (D) in Table 1.2 where each transaction has its transaction ID and timestamp (in days) at which it took place. If the user-specified period is 3 days, The following pattern is extracted:*

$$\{p, s\} \quad [support = 7, period = 3days].$$

This pattern indicates that items p and s co-occur every 3 days, and they are present in a total of 7 transactions in the temporal database. Unlike the frequent item model, which only considers the frequency of items in a dataset, the generated patterns consider temporal information. This enables a deeper understanding of the relationships between items and their occurrences over time.

To discover periodic patterns, we can use algorithms such as those proposed in [33, 34, 15, 14, 35]. These algorithms incrementally build patterns by adding one itemset at a time and pruning the search space to reduce computational complexity.

Periodic pattern mining has many applications, such as identifying seasonality in sales data. This allows businesses to optimize their inventory management and marketing strategies. It is also helpful in detecting periodic trends in social media activity, which provides valuable insights into user behaviour and preferences and predicts usage patterns of online services. It is a powerful tool for analyzing time-series data and can provide valuable insights for decision-making in various fields.

1.1.5 Partial Periodic Patterns (3P)

In the existing model of periodic patterns, patterns that exhibit minor deviations from periodic behaviour are often disregarded as uninteresting. This tendency is commonly observed across diverse real-world datasets, leading to the loss of valuable insights in real-world applications. To capture complete knowledge, the concept of partial periodic patterns was introduced. Partial periodic patterns are an important type of pattern in temporal databases that reflect the regular occurrence of specific events or behaviours during certain times or days. In many real-world scenarios, partial periodic behaviour exists, and it is important to identify such patterns for various applications, such as air pollution analysis and traffic congestion management. The public bus system is a great example of a partial periodic pattern. Buses usually follow a set timetable, but sometimes they experience delays due to traffic or other issues, causing deviations from their expected schedule. This behaviour showcases a partial periodic pattern, where there is a repeating pattern with occasional interruptions.

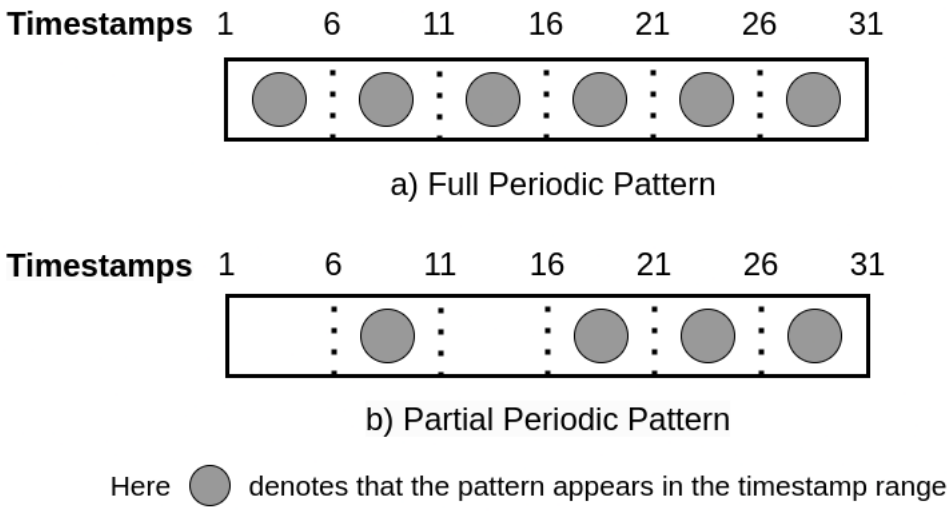


Figure 1.1: Difference between periodic and partial periodic patterns: (a) Periodic patterns show itemsets that appear in every specified interval, such as every 5 minutes. (b) Partial periodic patterns may only appear in some intervals due to noise or other factors

Fig 1.1 depicts the difference between periodic and partial periodic patterns. For periodic patterns (a.k.a full periodic patterns), an itemset must appear at defined intervals, while in partial periodic patterns, the itemset might appear in some or all intervals due to noise or other factors. The model for mining partial periodic patterns determines the interestingness of a pattern by counting the number of periodic occurrences of an item. An item's occurrence is considered periodic if the time gap between two consecutive transactions is less than the user-defined maximum interarrival time ($maxIAT$). For a pattern to be recognized as a partial periodic pattern, the itemset must appear periodically in at least $minPS$ (minimum periodic-support) number of transactions.

To better understand this, an example of a partial periodic pattern is as follows:

Example 2. *Considering the same hypothetical temporal database (D) in Table 1.2 where each transaction has its transaction id and timestamp (in days) at which it took place. If the user-specified maximum inter-arrival time ($maxIAT$) is 3 days and minimum periodic-support ($minPS$) is 4 or 30%. The following pattern is extracted*

$$\{p, r\} \quad [periodic\text{-}support = 5 \text{ or } 38.46\%].$$

This pattern indicates that items p and r co-occur together at the maximum interval of 3 days in certain segments of the database, and 5 of its occurrences in the temporal database are periodic. Since this pattern is not periodic over the entire database, this knowledge is missed by the periodic pattern model.

The discovery of partial periodic patterns in a temporal database involves two key tasks: assessing the periodic interestingness of a pattern and discovering all partial periodic patterns in the database.

The assessment of periodic interestingness is non-trivial because existing periodic pattern models only consider the support of a pattern when determining its interestingness.

Partial periodic pattern mining is a challenging problem that requires the development of novel algorithms and techniques to effectively capture the temporal behaviour of items in discovering interesting regularities. Despite the challenges, the discovery of the 3Ps has numerous real-world applications and can provide valuable insights and recommendations for various domains. The 3P model was initially introduced by Uday et al. [20] with the aim of identifying meaningful regularities in temporal databases. They have proposed a new measure, called *periodic-support* (PS) to identify the number of periodic occurrences of a pattern and proposed a unique tree structure, known as the partial periodic frequent pattern tree *(3P-tree)*, as well as an algorithm called *3P-growth*, which was developed specifically for mining partial periodic patterns in temporal databases.

1.1.6 Null Invariant Property

The *null-invariant* property is a desirable characteristic of a measure as it allows it to determine the interestingness of a pattern without being influenced by the absence of items or itemsets in a dataset. In other words, the measure should be able to identify both frequent and rare itemsets that exhibit meaningful associations without being biased towards frequent items or generating a large number of uninteresting patterns. To understand it better, consider a non-null-invariant measure like *support*. If we mine patterns using *support* from the example database in Table 1.2, the interesting patterns generated would be biased towards “p” and “s” because the database contains “p” and “s” more frequently however patterns like “p and v” may holds importance if “v” has higher utility over “s” but would be missed because of less occurrence in the database.

1.1.7 Anti Monotonic Property

The *anti-monotonic* property is another desirable characteristic of measures used in pattern mining. It states that if a pattern is considered interesting, then all of its subsets should also be considered interesting. This property is important because it helps to reduce the search space and computational cost of mining patterns by eliminating uninteresting or redundant itemsets from consideration. By enabling the efficient discovery of correlated patterns, the anti-monotonic property is a key feature that makes pattern mining practical in real-world applications. For example, consider the example database in Table 1.2, if pattern “p, q and r” is considered an interesting pattern, then all its subsets like “p and q” and “p and r” should also be considered interesting. Similarly if item “v” is uninteresting then all the patterns containing “v” would be uninteresting. This property helps to reduce the search space and computational cost of mining patterns by eliminating uninteresting or redundant itemsets from consideration.

1.2 Research gap and Motivation

The *rare item problem* is a common issue encountered in frequent itemset mining, which refers to the difficulty of identifying frequent patterns in a dataset where some items occur rarely. The problem arises from the use of a single minimum support threshold ($minSup$) for the entire database, which assumes that all items have similar frequencies or occurrence behaviour. However, for most real-world applications, the frequency of items varies widely, and finding frequent patterns using a single $minSup$ leads to the following problem.

1. At high $minSup$, we miss the frequent patterns containing rare items because many rare items fail to satisfy the high $minSup$ values.
2. To find the patterns containing both frequent and rare items, we have to set a low $minSup$ value. However, this may cause a combinatorial explosion, producing too many patterns, most of which may be uninteresting to the user.

This dilemma is known as the *rare item problem*. The rare item problem is not only limited to the traditional setting of frequent itemset mining but also affects other areas such as graph mining, time series analysis, and text mining. This problem also exists in partial periodic pattern mining. Existing models for mining 3Ps are based on the same assumption that items in the database occur uniformly, which is often not the case in real-world applications. Since only a single $minPS$ is used for finding 3Ps it leads to the following problem

1. At high $minPS$, we miss the patterns containing rare items because many rare items fail to satisfy the high $minPS$ values.
2. If we set $minPS$ to a low value to find patterns containing both frequent and rare items. It causes a combinatorial explosion producing too many uninteresting patterns.

These limitations highlight a gap in the research field of Partial periodic pattern mining that needs to be addressed.

The rare item problem poses a significant challenge for many real-world applications, such as market basket analysis, anomaly detection, and recommendation systems, where the discovery of infrequent but meaningful patterns can have a significant impact on business insights and user experiences. To overcome this dilemma, researchers have explored alternative measures such as χ^2 [36], *all-confidence* [37], *kulk* [38], *relative support* [39], and *Kulczynski* [38] each with its own selection bias. However, no universally accepted best measure exists to find frequent itemsets in any database. Researchers are attempting to suggest appropriate measures based on user or application requirements. Recently, null-invariant measures such as *all-confidence* [37] and $kulk$ [38] have been advocated because they disclose genuine correlations without being influenced by the object co-absence in a database.

1.3 Overview of the proposed approach

The objective of partial periodic pattern mining is to identify all patterns in a temporal database (D) that meet the user-specified minimum periodic-support ($minPS$) requirement and maximum inter-arrival time ($maxIAT$). However, the existing model for mining 3Ps suffers from the rare item problem. This problem arises when certain items appear infrequently in the database, making it challenging to capture their periodic behaviour. If we set the $minPS$ to a low value, we generate too many uninteresting patterns. On the other hand, if the $minPS$ is high, we miss the patterns containing the rare items.

To address this issue, we propose an improved model and present an improved depth-first search algorithm to extract 3Ps. In the proposed model, we introduce a new measure called *periodic confidence*. The minimum periodic confidence ($minPC$) measure is specifically designed to assess the interestingness of a pattern by considering how frequently the rare item appears in the database. It considers both the periodicity of the pattern and its association with the rare item in the pattern. It filters out the uninteresting patterns when the minimum periodic-support ($minPS$) is low, which is the case during the mining of patterns with rare items. The intuition behind proposing the measure is that if the periodic frequency of a pattern containing a rare item is close to the overall frequency of a rare item in the database, then the pattern is likely to be an interesting pattern. On the other hand, if a pattern containing a rare item occurs infrequently with respect to the frequency of the rare item in the database, then it is likely to be a random occurrence and not a meaningful pattern. The *periodic-confidence* measure satisfies both *null-invariant* and *anti-monotonic* properties.

We also proposed an improved depth-first search algorithm named G3P-Growth algorithm. This algorithm compresses the temporal database into a Generalized Partial Periodic Pattern-tree (G3P-tree) and recursively mines the tree to identify all partial periodic patterns. In existing periodic pattern mining algorithms, a tree structure records an item's periodic and irregular occurrence information to calculate periodic-support. However, this method requires storing timestamps for all irregular occurrences of an item or pattern, leading to a large tree structure that increases memory and runtime requirements. To address this issue, the G3P-growth algorithm uses a novel lossless data pruning technique called *irregularity pruning* to eliminate irregular occurrences of an item in the temporal database and reduce memory and runtime requirements. Experimental results demonstrate that the proposed algorithm is highly scalable and efficient. We also present two case studies to demonstrate the potential real-world applications of the proposed model. Overall, this research aims to provide a practical and effective model for mining partial periodic patterns that can be applied to real-world datasets.

1.4 Contributions

1. We introduce the problem of mining partial periodic patterns in the context of the rare item problem. We propose a novel measure, *periodic-confidence* (PC), to determine the periodic interestingness of a pattern.

2. We introduce an efficient depth-first search algorithm based on pattern-growth technique [40] to find partial periodic patterns in a database.

3. We performed experiments on several real datasets, demonstrating that our algorithm is not only memory and runtime-efficient but also scalable. We also present two case studies to demonstrate the real-world applications of the proposed model.

1.5 Thesis Organization

The rest of the thesis is organized as follows.

1. In Chapter 2, we discuss the related work on frequent pattern mining, periodic pattern mining, partial periodic pattern mining and the rare item problem.

2. In Chapter 3, we explain the background of partial periodic patterns.

3. In Chapter 4, we present the model for discovering partial periodic patterns in the non-uniform database

4. In Chapter 5, we evaluate our proposed algorithm using different datasets. It first explains the relevant details of the dataset and then presents the details of different experiments performed, followed by a discussion of the results. Two case studies are also presented to showcase the potential applications.

5. In Chapter 6, we provide the summary and conclusion of the thesis and discuss future research directions

Chapter 2

Related Work

This chapter provides a comprehensive review of the literature on frequent pattern mining, periodic pattern mining, partial periodic pattern mining, and the rare item problem.

2.1 Frequent Pattern Mining

In data mining, frequent pattern mining has proven to be a valuable tool for detecting interesting itemsets in large transactional databases. Motivated by the application of mining association rules in market basket database, Agarwal et al. [5] first introduced this concept and Apriori algorithm, defining frequent patterns as sets of items that appear frequently in a database, with the frequency of occurrence being measured by a *support threshold*. If the support of a pattern is equal to or greater than the user-specified *minimum support threshold*, it is considered a frequent pattern. By utilizing the concept of support, the Apriori algorithm can extract frequent patterns from a transactional database. The downward closure property of support reduces the search space and enables the algorithm to extract these patterns efficiently. Another important measure used in association rule mining is called *confidence*. *Confidence* helps determine the strength of the relationship between items in a rule. The *confidence* measure tells us how likely the consequent itemset will appear when the antecedent itemset is present. By comparing different confidence values, we can identify strong and meaningful associations between items in a dataset. Han et al. [40] introduced an alternative approach called FP-growth, which involves mining frequent patterns without the need for candidate generation. The FP-growth builds a compact data structure called FP-tree. Then, it recursively mines frequent patterns by exploring conditional pattern bases derived from the FP-tree. The compact representation of the dataset in the form of the FP-tree enables FP-Growth to perform pattern mining with reduced memory consumption and faster processing time. This method gained much recognition because it only necessitates a single database scan which enables it to efficiently mine patterns from large-scale transactional databases.

By identifying frequent patterns, it is possible to gain valuable insights into the associations among items that frequently co-occur in the database. As a result, the task of finding frequent patterns has

received considerable attention in the research community, leading to the development of various algorithms to find these itemsets in static databases [5], incremental databases [41], uncertain databases [42], fuzzy databases [43], and data streams [44].

All-confidence measure has emerged as a practical approach for discovering correlated patterns [45, 46]. It is computed as the ratio of the support of an itemset to the maximum support among its individual items. It differs from the confidence measure because it evaluates an itemset, while the confidence measure evaluates the association rule. The *all-confidence* measure ensures that association rules derived from the itemset have a minimum confidence equal to the all-confidence value. The *all-confidence* measure satisfies two important properties: anti-monotonicity and null-invariance. The anti-monotonic property states that all non-empty subsets of a correlated pattern must also be correlated, which reduces the search space and the computational cost of pattern mining. This makes it practical for real-world applications. On the other hand, the null-invariance property reveals genuine correlation relationships without being influenced by the absence of objects in a database. *All-confidence* facilitates the discovery of meaningful patterns involving both frequent and rare items while reducing irrelevant patterns.

A recent survey conducted by Luna et al. [25] provides comprehensive insights into the advancements and techniques employed in frequent itemset mining over the past 25 years.

2.2 Periodic Pattern Mining

Frequent pattern mining has numerous real-world applications in web and data mining, and over the years, many techniques have been proposed for discovering valuable and interesting patterns in large databases [47, 48, 49, 50]. The support or frequency of occurrence measures has typically been used to identify patterns in transactional databases. One area where earlier approaches have largely ignored crucial information is the temporal occurrence of patterns in the database. For example, in a traffic congestion database of an area, it is crucial to identify frequent patterns of traffic congestion and the time of their occurrence to manage traffic flow better. To address this shortcoming, researchers proposed incorporating periodicity, or the time of occurrence of a pattern, as a primary measure for discovering periodic patterns in temporal databases.

Ozden et al. [13] proposed two novel algorithms, named sequential and interleaved algorithms, to discover the temporal behaviour of patterns in transactional databases. To achieve this, the authors employed cycle pruning, cycle elimination, and cycle skipping techniques, which allow for the effective discovery of the full cyclic behaviour of patterns, leading to the design of cyclic association rules. They partitioned the complete dataset into disjoint subsets based on transaction time stamps to facilitate the mining process. Next, they applied pruning techniques to eliminate non-cyclic patterns from these subsets. The authors claimed that their proposed pruning techniques were highly effective and efficient,

enabling them to complete the mining process quickly. However, one of the shortcomings observed with this approach is that it cannot discover the patterns that span multiple windows.

Tanbeer et al. [14] extended the frequent pattern model to discover a class of full periodically occurring frequent patterns in a temporal database called *periodic-frequent patterns*. To achieve this, the authors proposed a pattern growth algorithm called *PFP-Growth* and introduced the periodic frequent pattern tree *(PF-tree)*, a new tree-based data structure that includes a tail node to store transaction identifiers. During the pruning of these nodes, the PF-tree preserves the occurrence information by transferring the list of identifiers to the parent node. The authors claimed that their complete mining process is highly efficient. They utilized a unique maximum periodicity measure and a support-based measure to generate full cyclic periodic frequent patterns. This approach has potential applications in various domains, such as market basket analysis and web usage mining. A key limitation of this extended model is that it suffers from an open problem of specifying the $minSup$ value.

Amphawan et al. [33] tackled this problem by proposing a non-support metric-based algorithm to find top-k periodic frequent patterns in a database. The authors proposed a single-scan algorithm that uses an efficient list-based data structure called *Top-k List Structure* to find all top-k periodic-frequent patterns.

Uday et al. [15, 16, 17, 51, 34, 52] discussed several techniques to discover full periodic frequent patterns in transactional databases effectively. Uday et al. [17] proposed a novel measure, the *minimum periodic ratio*, to identify complete frequent periodic patterns in transactional databases. They have also proposed a novel tree structure, known as the extended periodic frequent pattern tree *(ExPF-tree)*, and an algorithm called the extended periodic frequent pattern-growth *(ExPF-growth)* to mine the databases. They also introduced the concept of potential patterns consisting of only one item. The ExPF-tree comprises an ExPF-list and a prefix tree to preserve transactional identifiers, and the authors have applied two additional pruning techniques to eliminate uninteresting patterns.

Uday et al. [34] developed an efficient algorithm called periodic frequent pattern-growth++ *(PFP-growth++)*, which is an improved variant of Tanbeer's model for mining periodic frequent patterns in transactional databases. To store the patterns and complete the mining process, the authors have proposed a tree-based data structure called periodic frequent pattern tree++ *(PF-tree++)*, which includes a *PF-list++* and a prefix tree for maintaining the transactional identifiers of the patterns. Furthermore, the authors have introduced a new concept called *local periodicity*, which is utilized in two different phases, namely expanding and shrinking phases, to complete the mining process quickly. They also introduced two novel pruning techniques to improve the efficiency of the mining process.

2.3 Partial Periodic Pattern Mining

Periodic frequent patterns offer numerous real-world applications. However, there is a major limitation to this model that affects its usefulness. It can only detect fully periodic patterns within the data, which means all inter-arrival times must fall within the user-specified maximum threshold. In practice, this can lead to the loss of potential knowledge, as the random and noisy nature of real-world datasets can cause slight variations in patterns that deviate from their ideal behaviour, resulting in the model's failure to identify them. To address this issue, researchers introduced partial periodic patterns. Researchers also tried to address this issue by utilizing different alternative measures for identifying partial periodicity. For instance, Uday et al. [17] proposed *periodic-ratio* as a measure, while Rashid et al. [18] used the *standard deviation* of inter-arrival times as a measure, and Nofong [19] introduced a model that calculates the mean of *inter-arrival times*. However, a major drawback of these models is that they do not satisfy the downward closure property, which makes them computationally expensive and impractical for analyzing large, real-world databases.

To address the shortcomings of previous studies, Uday et al. [20] proposed a new model called Partial Periodic Pattern *(3P)* model to identify interesting partial periodic patterns in temporal databases. They introduced a novel tree structure called the partial periodic frequent pattern tree *(3P-tree)*, as well as a mining algorithm known as *3P-growth*. They also introduced a new measure called *periodic-frequency* to assess the importance of a pattern. Periodic-frequency is calculated as the number of periodic occurrences in the temporal database. A pattern is considered periodic if the time between two consecutive occurrences is less than or equal to a maximum inter-arrival time specified by the user. Since partial periodic itemsets satisfy the anti-monotonic property, the *3P-growth* algorithm utilizes this property to discover partial periodic itemsets in large real-world datasets.

Multiple attempts to extend the basic model for mining partial periodic patterns have been made in the literature to explore other applications or to improve the efficiency of the existing model. Yashwant et al. [53] extended the basic model to discover partial periodic patterns with high value in a quantitative temporal database. They addressed two limitations of the existing High Utility Itemset Mining $(HUIM)$ model: it allows for the external utility of items to vary over time and is designed to find recurring customer purchase behaviour. They proposed an efficient depth-first search algorithm, *PPHUI-Miner*, to enumerate all partial periodic high-utility itemsets in temporal databases.

Uday et al. [54] enhanced the basic partial periodic pattern model and made it flexible to discover spatially interesting patterns in a spatiotemporal database. The model used three constraints (maximum inter-arrival time, minimum period support, and maximum distance) to determine the interestingness of a pattern. They proposed an efficient Spatiotemporal-Equivalence Class Transformation *(ST-ECLAT)* algorithm to discover all partial periodic spatial patterns in a spatiotemporal database. The algorithm uses a smart depth-first search technique to find the desired patterns effectively.

Saideep et al. [55] proposed a parallel algorithm for mining partial periodic itemsets in large temporal databases. Most previous algorithms focused on centralized databases, making them non-scalable for big data environments. Their proposed approach increased the performance by distributing transactional identifiers among multiple machines and mining identical itemsets independently.

Likhita et al. [56] extended the basic model for discovering Maximal Partial periodic patterns for huge databases. They proposed a novel approach to tackle the combinatorial explosion problem in partial periodic pattern mining. They proposed a new model of maximal partial periodic patterns in a database and a pattern-growth algorithm called *max3P-growth*. They claimed that the proposed approach effectively prunes redundant patterns and is efficient and scalable.

2.4 The Rare item problem

The key element that makes frequent itemset mining practicable in real-world applications is $minSup$. It prunes the search space and limits the number of itemsets generated. Since the basic model of frequent itemset implicitly assumes that all items in a database have similar frequency (or occurrence behaviour), and only $minSup$ is used for the entire database. This assumption can lead to the rare item problem since the frequencies of items in most real-world datasets vary significantly. Consequently, if we set a high $minSup$ threshold, we may miss patterns that include rare items. Conversely, if we set the $minSup$ threshold to a low value, it can result in a combinatorial explosion of itemsets, and many of the patterns generated would be uninteresting. To address this issue in real-world applications, scholars have proposed a solution using the notion of multiple $minSups$ [57, 58, 59]. This concept involves using a *minimum item support (minIS)* constraint on each item within the database, with the $minSup$ of a pattern being represented by the lowest $minIS$ value of its constituent items. Consequently, each pattern can satisfy a distinct $minSup$ threshold depending on its underlying items. However, a major drawback of this approach is that determining the $minIS$ values for the items remains an unresolved problem.

Correlated pattern mining was introduced by Brin et al. [36] to tackle the issue of identifying rare items. To uncover correlated patterns, they employed the χ^2 statistical measure. Since then, various measures of interestingness have been proposed based on probability theory, statistics, or information theory. Examples include the all-confidence [37], kulk [38], relative support [39], and Kulczynski [38] measures. Each measure has its own selection bias, which explains why one pattern is preferred over another. Thus, there is no one universally accepted measure to discover correlated patterns in a given database. Researchers are currently making efforts to suggest a suitable measure based on user and/or application requirements [60, 37, 61].

Since the model for mining periodic frequent patterns and partial periodic patterns also uses one $minSup$ and $minPS$, respectively, for the entire database due to which, it also suffers from the is-

sue of the rare item problem. Several attempts have been made by Uday et al. [15, 16] in the past to tackle this issue. Uday et al. [15] extended Liu's model of multiple minimum support [57] that aims to discover rare, full periodic frequent patterns in non-uniform transactional databases. To improve the efficiency of the mining process, the authors utilized a list-based tree data structure, the multi-constraint periodic frequent pattern tree *(MCPF-tree)*, which comprises an *MCPF-list* and a prefix tree that stores the transactional identifiers of the patterns. Additionally, two novel constraint measures, the minimum item support ($minIS$) and maximum item periodicity ($maxIP$) were introduced to address the combinatorial explosion issues that arise during mining. Every item in the database were specified with $minIS$ and $maxIP$ values. The authors also proposed a dynamic method for assigning the maximum item periodicity value of any pattern. However, the proposed model was slower than the model in [14] because the generated periodic-frequent patterns do not satisfy *anti-monotonic property*.

Surana et al. [16] proposed an extension to the MCPF-tree-based approach [15] for mining rare full periodic frequent patterns in transactional databases. Since the MCPF-tree-based approach did not satisfy the downward closure properties required for this task. To address this limitation and improve the efficiency of the mining process, the authors introduced another list-based tree data structure, the maximum constraints periodic frequent pattern tree *(MaxCPF-tree)*, which includes a *MaxCPF-list* and a prefix tree for storing the transactional identifiers of the patterns. Like the MCPF-tree-based approach, the authors employed the minimum item support and maximum item periodicity constraint measures to manage combinatorial explosion issues during mining. Additionally, they used two pruning techniques to discard uninteresting patterns and showed that the MaxCPF-tree-based approach could identify rare full periodic frequent patterns more quickly than the MCPF-tree-based approach.

In the two studies mentioned above [15, 16], a shared challenge is the methodology used to define the $minIS$ and $maxIP$ values for items. Despite allowing for each item to have distinct $minIS$ and $maxIP$ values, the model encounters certain restrictions: (i) It requires multiple user input parameters, which becomes practically impossible to determine the appropriate values for large datasets, and (ii) We observed that when using this methodology to specify items' $maxIP$ values in temporal databases where items may have comparable $support$ value but distinct periodicities, the rare item problem can still arise.

2.5 Differences with the existing approaches

The current techniques for mining partial periodic patterns face challenges in identifying patterns with both frequent and rare items, as they use a single minimum periodic support ($minPS$) threshold to assess the interestingness of a pattern throughout the database.

None of the preceding approaches addresses the rare item problem in mining 3Ps from temporal transactional databases.

2.6 Summary

In this chapter, we have discussed the existing works related to frequent pattern mining, periodic pattern mining, and partial periodic pattern mining. In the next chapter, we will provide a brief overview of partial periodic pattern mining.

Chapter 3

Model of Periodic Patterns

As a part of the background, in this chapter, we explain the model of the 3Ps.

3.1 Introduction

In earlier chapters, we introduced the fundamental models of frequent patterns and periodic patterns. However, the periodic pattern model has some limitations that can be addressed by using partial periodic patterns. First, we will discuss these limitations, and then in the following subsection, we will delve into the model of partial periodic patterns and explain the associated terminologies.

Partial periodic patterns are an extension of Periodic patterns which tries to relax the existing constraints. Following are the set of limitations of the periodic patterns that are resolved by partial periodic patterns

1. Inability to mine complex patterns: Periodic patterns requires a fixed regular interval between their repetitions, which may not always be present in the data, especially for most real-world applications.

2. Inability to handle missing data and noise: The model of partial periodic patterns assumes that all items are present and equally spaced, but for real-world applications, there can be instances when the item is not recorded in the transaction or gets recorded in some other transaction. This phenomenon can lead to important patterns getting missed.

3. Inability to model real-life phenomenon: Periodic patterns assume ideal behaviour, while for most real-world applications, there are always deviations from the ideal behaviour. There can be instances when the pattern stops appearing for a short duration and later starts appearing regularly in the data. These patterns cannot be extracted using the model of periodic patterns.

3.2 Model of Partial Periodic Patterns

To overcome the above-listed limitations and mine the complex patterns from the database, the model of partial periodic patterns was introduced.

Let $I = \{i_1, i_2, \cdots, i_m\}$, $m \geq 1$, be a set of m distinct items appearing in transactional temporal database. Let $A \subseteq I$ be a pattern (or an itemset). A pattern containing n number of items is called a n-pattern. A transaction $t_{tid} = (tid, ts, B)$, where $tid \geq 1$ represents the transaction identifier, $ts \in R^+$ represents the timestamp and $B \subseteq I$ is a pattern. A **temporal database** D is a collection of transactions. That is, $D = \{t_1, t_2, \cdots, t_y\}$, $1 \leq y \leq |D|$, where $\mid D \mid$ represents the size of database. If a pattern $A \subseteq B$, it is said that A occurs in transaction t_{tid}. The timestamp of this transaction is denoted as ts^A_{tid}. Let $TS^A = \{ts^A_x, ts^A_y, \cdots, ts^A_z\}, x, y, z \in (1, \mid D \mid)$, denote the set of all timestamps in which the pattern A has appeared in the database. The $support$ of A in D, denoted as $sup(A)$, represents the number of transactions containing A in D. That is, $sup(A) = \mid TS^A \mid$.

Example 3. *Let $I = \{p, q, r, s, t, u, v\}$ be the set of items. The temporal database of the items in I is shown in Table 1.2. The set of items p, q and r, i.e., $\{p, q, r\}$ (or pqr, in short) is a pattern. It is a 3-pattern because it contains three items. The pattern pqr appears in the transactions with timestamps of 3, 4, 12, 13, 15 and 16. Therefore, $ts^{pqr}_2 = 3$, $ts^{pqr}_3 = 4$, $ts^{pqr}_{11} = 12$, $ts^{pqr}_{12} = 13$, $ts^{pqr}_{13} = 15$ and $ts^{pqr}_{14} = 16$. The complete set of timestamps at which pqr has occurred in Table 1.2, i.e., $TS^{pqr} = \{3, 4, 12, 13, 15, 16\}$. The support of pqr, i.e., $sup(pqr) = \mid TS^{pqr} \mid = 6$.*

To identify if an itemset appears regularly in the database, we need to identify the time gap between the two consecutive occurrences of an item. To capture this information, we use the notion of inter-arrival time. Basically, it tells us the time gap between the two consecutive appearances of an itemset in the complete database. By using the set of inter-arrival times of an itemset, we can get a better idea of how regularly and consistently the itemset appears in the database. It can be formally defined as

Definition 1. ***(Inter-arrival time of itemset A)*** *Let ts^A_j, $ts^A_k \in TS^A$, $1 \leq j < k \leq m$, denote any two consecutive timestamps in TS^A. The time difference between ts^A_k and ts^A_j is referred to as* ***an inter-arrival time*** *of A, and denoted as iat^A. That is, $iat^A = ts^A_k - ts^A_j$.*

Example 4. *The pattern pqr has consecutively appeared in the transactions with timestamps of 3 and 4. The difference between these two timestamps gives an inter-arrival time of pqr. That is, $iat^{pqr}_1 = 4 - 3 = 1$. Similarly, other inter-arrival times of pqr are: $iat^{pqr}_2 = 12 - 4 = 8$, $iat^{pqr}_3 = 13 - 12 = 1$, $iat^{pqr}_4 = 15 - 13 = 2$ and $iat^{pqr}_5 = 16 - 15 = 1$.*

To extract partial periodic patterns from the temporal database, it is necessary to establish criteria for identifying consecutive occurrences of an item that can be considered periodic occurrences. The two consecutive occurrences of an itemset are considered periodic if their inter-arrival time is less than

the user-defined maximum interarrival time ($maxIAT$). The notion of the periodic appearance of an itemset is formally defined as

Definition 2. ***(Periodic appearance of itemset** A**.)*** *Let* $IAT^A = \{iat_1^A, iat_2^A, \cdots, iat_k^A\}$, $k = sup(A) - 1$, *be the set of all inter-arrival times of* A *in* D. *An inter-arrival time of* A *is said to be* ***periodic*** *if it is no more than the user-specified maximum inter-arrival time (*$maxIAT$*). That is, a* $iat_i^A \in IAT^A$ *is said to be* ***periodic*** *if* $iat_i^A \leq maxIAT$.

Example 5. *The set of all inter-arrival times of* pqr *in Table 1.2, i.e.,* $IAT^{pqr} = \{1, 8, 1, 2, 1\}$. *If the user-specified* $maxIAT = 2$, *then* iat_1^{pqr}, iat_3^{pqr}, iat_4^{pqr} *and* iat_5^{pqr} *are considered as the periodic occurrences of* pqr *in the data. On the contrary,* iat_2^{pqr} *is considered as an irregular occurrence of* pqr *because* $iat_2^{pqr} \not\leq maxIAT$.

To extract meaningful patterns from the temporal database, a constraint is required to filter out uninteresting patterns. This constraint allows us to focus on those patterns that occur frequently enough to be considered meaningful. To achieve this, the measure of periodic support was introduced. It is a way to define a minimum percentage threshold of transactions in which the itemset should appear periodically to be considered a partial periodic pattern. The measure can be formally defined as

Definition 3. ***(Periodic-support of pattern** A **[62].)*** *Let* $\widehat{IAT^A} \subseteq IAT^A$ *be the set of all inter-arrival times that have value no more than* $maxIAT$. *That is,* $\widehat{IAT^A} \subseteq IAT^A$ *such that if* $\exists iat_k^A \in IAT^A : iat_k^A \leq maxIAT$, *then* $iat_k^A \in \widehat{IAT^A}$. *The* periodic-support *of A, denoted as* $PS(A) = \frac{|\widehat{IAT^A}|}{|D|-1}$, *where* $|D| - 1$ *denote the maximum number of inter-arrival times a pattern may have in* D.

Example 6. *Continuing with the previous example,* $\widehat{IAT^{pqr}} = \{1, 1, 2, 1\}$. *Therefore, the* periodic-support *of 'pqr,' i.e.,* $PS(pqr) = \frac{|\widehat{IAT^{pqr}}|}{|D|-1} = \frac{|\{1,1,2,1\}|}{13} = 0.307$. *To find this pattern as a partial periodic pattern in the basic model, we must set a low* $minPS$ *value, say 0.15.*

In simple terms, The *periodic-support* refers to the relative frequency of periodic occurrences of an itemset in a dataset. It helps in determining the interestingness of a pattern by considering both its frequency and inter-arrival times of items in the data.

Using the terms defined above, the concept of partial periodic patterns is introduced.

Definition 4. *(Partial periodic pattern* A*.) The pattern* A *is said to be a partial periodic pattern if* $PS(A) \geq minPS$ *and* $\forall i \in \mathbb{N}$, $\widehat{IAT_i^A} \leq maxIAT$, *where* $minPS$ *and* $maxIAT$ *represents the user-specified* minimum periodic-support *and* maximum inter-arrival time.

Example 7. *If the user-specified* $minPS = 0.15$ *or* 15% *and* $maxIAT$ *is 2 then* pqr *is a partial periodic pattern because* $PS(pqr) \geq minPS$ *and* $\forall i \in \mathbb{N}$, $\widehat{IAT_i^{pqr}} \leq maxIAT$

We now define a clear problem statement of mining partial periodic patterns using the terminologies described and explained above.

Problem Statement

Given a temporal database (D), a set of items (I) and the user-specified *maximum inter-arrival time* ($maxIAT$), and *minimum periodic-support* ($minPS$), the problem of partial periodic pattern mining is to find all patterns in D that satisfy the $minPS$ constraint. The partial periodic itemsets discovered by the proposed model satisfy the downward closure property. The following property 1 can easily prove that the model follows the downward closure property.

Property 1. *If $X \subseteq Y$, then $TS^X \supseteq TS^Y$, therefore, $PS(X) \geq PS(Y)$.*

Example 8. *Consider two patterns, p and pq. Since $p \subseteq pq$ and the TS^p = {1, 3, 4, 7, 8, 10, 11, 12, 13, 15, 16} and TS^{pq} = {3, 4, 12, 13, 15, 16}, so $TS^p \supseteq TS^{pq}$. Therefore $PS(p) \geq PS(pq)$, i.e $\frac{11}{14-1} \geq \frac{6}{14-1}$ and $0.846 \geq 0.462$.*

3.3 Partial Periodic Pattern-growth (3P-growth) Algorithm

Partial Periodic Pattern-growth (3P-growth) is a depth-first search algorithm [20] for extracting partial periodic patterns from a given temporal database D. The 3P-growth algorithm involves the following two steps:

1. Compress the database into a partial periodic pattern tree (3P-tree)
2. Recursively mine the 3P-tree to find all partial periodic itemsets.

The idea behind converting the database into a 3P-tree is to remove redundancies and represent the information in a more compact format. This is accomplished by taking advantage of the fact that transactions in the database tend to have a common subset of items. By storing the database in this compressed form, the 3P-growth algorithm can perform partial periodic pattern searches more effectively without having to repeatedly scan the entire database.

The two steps of the 3P-Growth algorithm are

Construction of 3P-tree

A 3P-tree consists of two components:

1. **3P-list** - It contains information of each distinct *item* (i), with *periodic-frequency* (pf) and a pointer pointing to the first node in the prefix tree for the same item.
2. **Prefix-tree** - It stores the database in compressed form to facilitate efficient mining. The prefix-tree is similar to an FP-tree (Frequent Pattern-tree), but it differs in the way it stores occurrence information for each transaction. It consists of two types of nodes: ordinary node and tail-node.

The ordinary node is similar to the FP-tree node, whereas tail-node represents the last item of any sorted transaction and maintains a transaction-id list called "tid-list". The structure of a tail-node is represented as $N : [t_1, t_2, ..., t_n]$, where N is the node's item name and $t_i, i \in [1, n]$ represents the list of transactions where N is the last item in the set of transactions. Each node in the tree maintains parent, children, and node traversal pointers, but unlike an FP-tree, no node in a prefix-tree maintains a support count value.

Partial periodic patterns satisfy the anti-monotonic property, so partial periodic items (or 1-itemset) play an important role in the mining process. Figure 3.1(a)-(c) show the construction of 3P-list after scanning the first, second and every transaction in the database, respectively. Figure 3.1(d) shows the 3P-list containing partial periodic items in descending order of their pf value. The period and minPF values used to find these items are 2 and 2, respectively. In Figure 3.1 I denotes the item name, frq denotes the frequency of the item, pf denotes the periodic-frequency of the item and ts_l denotes last timestamp in which the item appeared. Let CI denote this sorted list of partial periodic items.

I	frq	pf	ts_1
p	1	0	1
r	1	0	1
s	1	0	1

(a)

I	frq	pf	ts_1
p	2	1	3
r	2	1	3
s	1	0	1
q	1	0	3
t	1	0	3

(b)

I	frq	pf	ts_1
p	11	9	16
r	8	5	16
s	10	6	16
q	7	4	16
v	3	1	15
u	4	2	12
t	4	2	11

(c)

I	frq	pf	ts_1
p	11	9	16
s	10	6	16
r	8	5	16
q	7	4	16
u	4	2	12
t	4	2	11

(d)

Figure 3.1: Construction of 3P-list. (a) After scanning the first transaction, (b) after scanning the second transaction, (c) after scanning the entire database, and (d) the final 3P-list after pruning all aperiodic items

The 3P-list obtained contains the partial periodic items(or 1-itemset). We perform another scan on the database to construct the prefix tree. Fig 3.2 (a), (b) and (c) show the 3P-tree constructed after scanning the first, second and every transaction in the database, respectively. For simplicity, we do not show the node traversal pointers in trees; however, they are maintained as an FP-tree does. The initial node in the FP-Tree is represented by an null node.

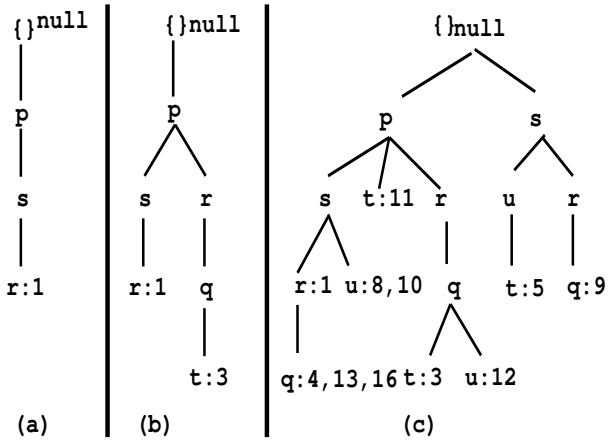


Figure 3.2: Construction of 3P-tree. (a) After scanning the first transaction, (b) After scanning the second transaction, and (c) After scanning the entire database

Recursive mining of 3P-tree

To mine the 3P-tree, take each partial periodic item as the initial suffix itemset. Next, create a conditional pattern base. Using this information, create the conditional 3P-tree and recursively perform mining on it. Concatenate the suffix itemset with partial periodic itemsets produced from a conditional 3P-tree to accomplish pattern growth.

Continuing with the same example, We start with the bottom-most item in the 3P-list. Since t is the bottom-most item, each node with $item = t$ in 3P-tree must be a tail-node. Fig 3.3(a) shows the prefix tree for $\langle t \rangle$. As no item in the prefix tree of t has *period-support* more than the user-specified $minPS$, the conditional tree of t will be empty as shown in Fig 3.3(b). Only t will be generated as a partial periodic pattern. Next, the item t will be completely pruned from the original 3P-tree by pushing its ts-lists to the respective parent nodes. Fig 3.3(c) shows the resultant 3P-tree after pruning item t. The entire process is repeated until the 3P-list in the 3P-tree is empty.

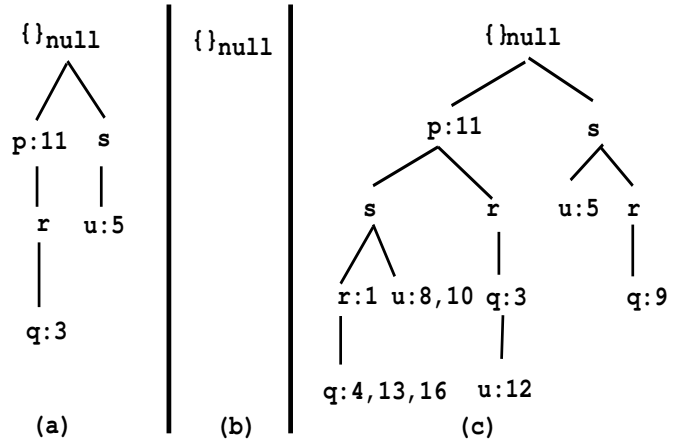


Figure 3.3: Recursive mining of 3P-tree. (a) Prefix tree for item t (b) conditional tree for item t and (c) 3P-tree after pruning item t

3.4 Summary

In this chapter, we explain the model of partial periodic pattern along with the 3P-Growth algorithm. We also explained the issues causing rare item problem and their possible solutions. In the next chapter, we explain the *periodic-confidence* measure for discovering partial periodic patterns in the non-uniform database along with the Generalized Partial Periodic Pattern-growth *(G3P-growth)* algorithm for efficient mining.

Chapter 4

Proposed Approach

In this chapter, we present the proposed model and the algorithm. We first introduce the modified problem statement and then present the proposed model. Next, we present the algorithm.

4.1 Issues with the existing model of 3P

The model of 3P (Partial Periodic Pattern) is as follows (refer to Chapter 3 for details). Given a temporal database, the problem of 3Ps is to extract all periodic regularities subject to user-given $minPS$(minimum periodic support) and $maxIAT$ (maximum inter-arrival time) constraints. The $minPS$ threshold is a key parameter in this task because it determines the minimum frequency of occurrence of a pattern in the database. Therefore, setting an appropriate $minPS$ threshold is crucial in achieving meaningful results in 3P mining. However, the model of 3P suffers from rare item problem. Rare items refer to the infrequently occurring items in the database that are crucial for discovering meaningful patterns. However, patterns containing rare items have low periodic support, which creates a dilemma while setting the minimum periodic support threshold ($minPS$):

1. If we set the $minPS$ threshold to a high value, then we might miss important patterns containing rare items.

2. If we set the $minPS$ threshold to a low value, then we generate too many patterns, and many of them would be uninteresting.

Consequently, patterns containing rare items can often get lost in the noise, resulting in missed opportunities for discovering valuable insights.

To illustrate this challenge, consider a database of customer purchases from a grocery store in Table 4.1, where each transaction contains the items purchased along with the timestamp.

Suppose we want to identify the items being bought together periodically for promotional and advertising activities. One observed pattern could be that “Chips and Soft Drinks” are bought together

Transaction ID	Timestamp	Items
1	2022-01-01 10:00:00	Chips, Soft drinks
2	2022-01-02 14:00:00	Chips, Beer
3	2022-01-03 12:00:00	Chips, Soft drinks
4	2022-01-06 11:00:00	Chips, Soft drinks
5	2022-01-07 16:00:00	Chips, Beer
6	2022-01-09 09:00:00	Chips, Soft drinks
7	2022-01-10 15:00:00	Chips, Soft drinks
8	2022-01-13 13:00:00	Chips, Beer
9	2022-01-14 10:00:00	Chips, Soft drinks
10	2022-01-15 12:00:00	Bread, Cheese, Soft drinks
11	2022-01-15 14:00:00	Bread, Chips, Beer
12	2022-01-16 11:00:00	Bread, Chips, Soft drinks
13	2022-01-16 16:00:00	Chips, Beer, Soft Drink
14	2022-01-17 09:00:00	Bread, Beer, Cheese, Soft drinks
15	2022-01-18 13:00:00	Bread, Chips, Beer

Table 4.1: Example database of customer purchases from a grocery store

every week. However, "Beer and Chips" are also bought together, but they are usually bought together on weekends. Since Beer is a higher-priced item and appears less frequently in the database with respect to chips, it creates the rare item problem. For instance, If we set a high $minPS$ threshold like 0.6. In that case, we might miss this pattern as "Beer and Chips" has PS of 0.4, leading to a loss of important insights. On the other hand, if we set a low $minPS$ threshold like 0.1, we might generate too many uninteresting patterns like "Beer and Bread" and "Beer and Soft Drinks", making it challenging to identify the meaningful ones that contain rare items. Pattern like "Beer and Bread" and "Beer and Soft Drinks" are uninteresting because these items appear together only twice and can be considered as random occurrence. In contrast "Beer and Chips" appear together for six times out of seven periodic occurrences of Beer making it an interesting pattern.

This observation highlights the importance of considering how often rare items appear in a database while evaluating the significance of a pattern. In simpler terms, If the periodic frequency of a pattern containing a rare item is close to the overall frequency of a rare item in the database, then the pattern is likely to be an interesting pattern.

On the other hand, if a pattern containing a rare item occurs infrequently with respect to the frequency of the rare item in the database, then it is likely to be a random occurrence and not a meaningful pattern.

Therefore, we propose a novel measure that captures the periodicity of a pattern and its association with rare items by using the frequency of the rare item in the database. This measure can be used to mine partial periodic patterns with rare items in a temporal database and is discussed in detail further.

4.2 Proposed Model

For the proposed model, we use the same example database shown in Table 1.2 for presenting the examples of terms in the proposed model. The proposed measure for addressing the rare item problem in partial periodic pattern mining is named periodic confidence. The periodic confidence of a measure can be formally defined as:

Definition 5. *(**Periodic-confidence of pattern** A) The periodic-confidence of pattern A, denoted as $PC(A) = \frac{|\widehat{IAT^A}|}{min(sup(i_j)|\forall i_j \in A)-1}$, where $min(sup(i_j) \mid \forall i_j \in A) - 1$ represents the minimum number of inter-arrival times of a least frequent item in A.*

Example 9. *Continuing with the previous example, the* periodic-confidence *of pqr, i.e., $PC(pqr) = \frac{|\widehat{IAT^{pqr}}|}{min(sup(p),sup(q),sup(r))-1} = \frac{|\{1,1,2,1\}|}{min(11,7,8)-1} = \frac{4}{6} = 0.667$ $(= 66.7\%)$. It can be observed that though the pattern pqr has low $period\text{-}support$ in the entire data, it has a high* periodic-confidence*. Thus, the usage of periodic-confidence facilitates us in finding patterns containing both frequent and rare items.*

The periodic-confidence measure is designed to capture the periodicity of the patterns and its association with the rare item in the pattern. It is the ratio of the number of periodic occurrences of a pattern with respect to the frequency of rare items in the database. The intuition behind the measure is that if a pattern occurs with high periodic-confidence, then that pattern is most likely to be a meaningful pattern, as the frequent occurrence of a pattern with a rare item implies there exists some relation between the pattern and the rare item and the occurrence of a rare item with the pattern is not a random event. By leveraging the frequency of a rare item in the complete database, the periodic confidence measure identifies the number of periodic occurrences of a pattern with a rare item. This enables the measure to distinguish meaningful patterns containing both frequent and rare items from random occurrences.

Considering the case for patterns containing only frequent items, those with high periodic support will also exhibit high periodic confidence. High periodic support indicates a significant number of periodic occurrences of the pattern throughout the entire database. Additionally, high periodic support also suggests a higher number of periodic occurrences with respect to the rarest item within the pattern. Therefore, patterns with high periodic support indicate an important and recurring relationship between the items in the pattern. Conversely, patterns with low periodic support among patterns containing only frequent items will also have low periodic confidence. This means that these patterns do not

show a strong recurring association between their items. This enables the proposed model to filter out uninteresting patterns that lack a strong periodic association between their items.

The periodic-confidence measure will enable the partial periodic model to be applied in various fields, such as identifying rare events in medical data or detecting anomalies in financial transactions. By using the measure, we can identify periodic patterns that contain rare items and have a high degree of confidence that they are not just random occurrences. This can enable proactive measures to be taken to mitigate the impact of the rare event.

The existing model for mining partial periodic patterns uses only minimum periodic support ($minPS$) to identify interesting patterns. However, incorporating the proposed measure of periodic confidence ($minPC$) is crucial for real-world applications. While it may appear as a better solution to remove the minimum periodic support constraint and use only periodic confidence to determine interestingness, this approach can lead to the identification of patterns with items that appear only once in the database. Additionally, some items may be rare due to random noise or errors, and their appearance in a pattern may not be significant or interesting. The definition of a rare item may also vary depending on the application and database size. Consequently, using only minimum periodic confidence may generate uninteresting patterns containing rare items that are of no interest. Therefore, the constraint of minimum periodic support is necessary to filter out such noise, errors, and uninteresting patterns.

We now modify the definition of partial periodic patterns incorporating the proposed measure.

Definition 6. *(Partial periodic pattern A.) The pattern A is said to be a partial periodic pattern if $PS(A) \geq minPS$ and $PC(A) \geq minPC$, where $minPS$ and $minPC$ represents the user-specified* minimum period-support *and* minimum periodic-confidence*, respectively. Our model employs $minPS$ constraint to prune the patterns with very few periodic occurrences in the database.*

Example 10. *If the user-specified $minPS = 15\%$ and $minPC = 50\%$, then pqr is a partial periodic pattern because $PS(pqr) \geq minPS$ and $PC(pqr) \geq minPC$.*

To establish a clear problem statement for mining partial periodic patterns from temporal databases, we utilize the terminology outlined earlier.

Definition 7. ***(The problem of mining Partial Periodic Patterns from a temporal database.)*** *Given a temporal database (D) and the user-specified* maximum inter-arrival time *($maxIAT$)* minimum period-support *($minPS$), and* minimum periodic-confidence *($minPC$), the problem of partial periodic pattern mining is to find all patterns in D that satisfy the $minPS$ and $minPC$ constraints. The inter-arrival times of a pattern can also be expressed in percentage of $(ts_{max} - ts_{min})$, where ts_{min} and ts_{max} represents the minimum and maximum timestamps in D, respectively.*

The patterns generated by the proposed model satisfy both the *null invariance* property [60, 61] and the *convertible anti-monotonic* property [63].

The following two properties for the proposed measure can be proved as follows

Table 4.2: A 2×2 ($utility$) contingency table for A and B

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	F

Let A and B be two variables (or itemsets). The 2×2 contingency table for these variables is shown in Table 4.2. The term f_{11} represents the $frequency$ of AB in the database. The term f_{10} represents the $frequency$ of A in the transactions that do not contain B in the database. The term f_{1+} represents the total $frequency$ of A in the entire database. The term f_{01} represents the $frequency$ of B in the transactions that do not contain A in the database. The term f_{00} represents the total $frequency$ of other items in the database. The term f_{0+} represents the total $frequency$ of all items excluding A. The term f_{+1} represents the $frequency$ of B in the entire database. The term f_{+0} represents the total $frequency$ of all items excluding B in the database. The term F represents the total $frequency$ of all items in the database. If $maxIAT \geq ts_{max} - ts_{min}$, then *periodic-confidence* of AB is computed as follows:

$$PC(AB) = \frac{f_{11}}{min(f_{1+}, f_{+1})}. \tag{4.1}$$

Since *PC* is not influenced by f_{00}, f_{0+}, f_{+0}, or F, it can be stated that $O(M + C) = O(M)$. Thus, *periodic-confidence* satisfies the *null-invariant property*.

Property 2. *(**Null-invariance property** [60].) A binary measure of association is null-invariant if $O(M + C) = O(M)$, where M is a 2×2 contingency matrix, $C = [0\ 0; 0\ k]$ and k is a positive constant.*

Property 3. ***(The convertible anti-monotonic property.)*** *Let $B = \{i_1, i_2, \cdots, i_k\}, k \geq 1$ be an ordered pattern such that $sup(i_1) \leq sup(i_2) \leq \cdots \leq sup(i_k)$. If B is a partial periodic pattern, then $\forall A \subset B$, $A \neq \emptyset$ and $i_1 \in A$, A is also a partial periodic pattern. The proof is based on Property 4, Lemma 5 and 6, and shown in Theorem 7.*

Property 4. ***(Apriori property [64].)*** *If $A \subset B$, then $TS^A \supseteq TS^B$. Thus, $sup(A) \geq sup(B)$.*

Lemma 5. *Let $B = \{i_1, i_2, \cdots, i_k\}, k \geq 1$ be an ordered pattern such that $sup(i_1) \leq sup(i_2) \leq \cdots \leq sup(i_k)$. If $A \subset B$ such that $i_1 \in A$, then $PS(A) \geq PS(B)$.*

Proof. Using the Property 4, If $A \subset B$, then $TS^A \supseteq TS^B$ also

$$\begin{aligned} PF(A) &\geq PF(B) && (4.2)\\ = \frac{PF(A)}{\mid D \mid -1} &\geq \frac{PF(B)}{\mid D \mid -1} && (4.3)\\ = PS(A) &\geq PS(B). && (4.4) \end{aligned}$$

Hence proved. □

Lemma 6. *Let $B = \{i_1, i_2, \cdots, i_k\}, k \geq 1$ be an ordered pattern such that $sup(i_1) \leq sup(i_2) \leq \cdots \leq sup(i_k)$. If $A \subset B$ such that $i_1 \in A$, then $PC(A) \geq PC(B)$.*

Proof. According to Lemma 5, if $A \subset B$, then

$$\begin{aligned} PS(A) &\geq PS(B) && (4.5)\\ = \frac{PS(A)}{sup(i_1) - 1} &\geq \frac{PS(B)}{sup(i_1) - 1} && (4.6)\\ = PC(A) &\geq PC(B). && (4.7) \end{aligned}$$

Hence proved. □

Theorem 7. *Let $B = \{i_1, i_2, \cdots, i_k\}, k \geq 1$ be an ordered pattern such that $sup(i_1) \leq sup(i_2) \leq \cdots \leq sup(i_k)$. If B is a partial periodic pattern, then $\forall A \subset B$, $A \neq \emptyset$ and $i_1 \in A$, A is also a partial periodic pattern.*

Proof. If B is a partial periodic pattern, then $PS(B) \geq minPS$ and $PC(B) \geq minPC$. Based on Lemmas 5 and 6, it turns out that $\forall A \subset B,\ i_1 \in A,\ PS(A) \geq PS(B) \geq minPS$ and $PC(A) \geq PC(B) \geq minPC$. Thus, A is also a partial periodic pattern. Hence proved. □

To demonstrate the efficacy of our model, we provide an example set of patterns in Table 4.3. We used high $minPS$, low $minPS$ and low $minPS$ with high $minPC$ values to demonstrate the rare item problem with the existing model. The proposed model could generate patterns containing frequent and rare items without producing too many uninteresting patterns.

Table 4.3: Partial periodic patterns generated from Table 1.2 at different $minPS$ and $minPC$ values. The terms 'P,' 'PS,' 'I,' 'II,' and 'III' respectively represent 'Pattern,' 'period-support,' 'partial periodic patterns found at high $minPS$ value of 0.6,' 'partial periodic patterns found at low $minPS$ value of 0.3,' and 'partial periodic found using the proposed model at $minPS = 0.3$ and $minPC = 0.7$

P	PS	**I**	**II**	**III**
p	0.77	T	T	T
q	0.38	F	T	T
r	0.46	F	T	T
s	0.69	T	T	T
pq	0.31	F	T	F
pr	0.38	F	T	T
ps	0.46	F	T	F
qr	0.38	F	T	T
pqr	0.31	F	T	F

4.3 Generalized Partial Periodic Pattern-growth (G3P-growth)

The G3P-growth algorithm involves the following two steps: (i) compress the database into Generalized Partial Periodic Pattern-tree (G3P-tree) and (ii) recursively mine the tree to discover the complete set of partial periodic patterns.

The ideation behind compressing the database into Generalized Partial Periodic Pattern-tree (G3P-tree) is to reduce redundancy and represent the database in a more compact form. The G3P-tree is a hierarchical data structure that captures partial periodic patterns in a compressed form. The tree structure allows for efficient pattern matching by exploiting the shared information between transactions. By storing the database in this compressed form, the G3P-growth algorithm can efficiently search for partial periodic patterns without scanning the database multiple times.

To further improve the mining process, the G3P-growth algorithm uses a technique called irregularity pruning. The ideation behind this technique is to eliminate the aperiodic occurrence of an item that does not contribute to any partial periodic pattern and is reduntant in the pattern discovery process. This helps in reducing the overall size of the tree and improves the runtime. This technique is explained in details in the next subsection.

4.3.1 Irregularity pruning

Existing periodic pattern mining algorithms record the periodic and irregular occurrence information of an item in a $tree$ structure to calculate the *period-support*. We have observed that storing the timestamps of all irregular occurrences of an item (or pattern) increases the size of the $tree$ structure, which in-turn increases the memory and runtime requirements of an algorithm. With this motivation, we have come up with a technique to prune irregular occurrences of an item (or pattern) in a database.

In simpler words, The basic idea of *irregularity-pruning* is to remove items from transactions while constructing the G3P-tree. The aim is to eliminate items that do not contribute to the periodic occurrence of an item, where the periodic occurrence is defined as when an item appears in two consecutive transactions with a time difference less than the user-defined maximum inter-arrival time ($maxIAT$).

During the construction of the G3P-tree, if an item in a transaction is not periodic with the timestamp of the previous and the next occurrence, then item in that transaction does not contribute to any partial periodic pattern and can be eliminated from the mining process. This technique helps in reducing redundancy and improving the runtime of the algorithm.

Now we define the property of irregularity pruning with a formal definition and demonstrate it with an example using the sample temporal database in Table 1.2.

Property 8. ***(Irregularity pruning.)*** *Let $TS^A = \{ts^A_a, ts^A_b, \cdots, ts^A_c\}, 1 \leq a \leq b \leq c \leq m$, be the timestamps at which A has occurred in D. Let ts^A_p, ts^A_q, ts^A_r, $a \leq p < q < r \leq c$, be three consecutive timestamps in TS^A. If $ts^A_q - ts^A_p > maxIAT$ and $ts^A_r - ts^A_q > maxIAT$, then pruning ts^A_q from TS^A will not result in missing any information (or $PS(A)$ remains the same).*

Example 11. *In Table 1.2, the item s appears in the timestamps of 1, 4, 5, 6, 7, 8, 9, 10, 13, and 16. Thus, $IAT^s = \{3, 1, 1, 1, 1, 1, 1, 3, 3\}$. If $maxIAT = 2$, then $\widehat{IAT^s}\{1, 1, 1, 1, 1, 1\}$ and $PS(s) = \frac{6}{15} = 0.4$. Now, consider the last three occurrences of s in TS^s, i.e., 10, 13 and 16. Since $(13 - 10) > maxIAT$ and $(16 - 13) > maxIAT$, it can be stated that timestamp 13 is not contributing to the period-support of s in the database. Thus, pruning timestamp 13 from TS^s will not result in losing the period-support of s. Similarly, pruning the timestamp 16 will not result in losing the period-support of s. In other words, the period-support of s and its supersets can be calculated by storing only the timestamps 1, 4, 5, 6, 7, 8, 9 and 10.*

Before we describe the two steps G3P-Algorithm in detail, we describe the structure of the G3P-tree.

Structure of G3P-tree

The structure of G3P-tree is composed of a prefix-tree and a candidate event list, called G3P-list. The G3P-list indicates for each distinct *item* (i), its *support count* (s), *period-support count* (ps) and provides a pointer to the first node in the prefix-tree representing the event (nl). The prefix-tree in a

G3P-tree maintains two types of nodes: **ordinary** and **tail**. An ordinary node structure is similar to that used in an FP-tree except that it doesn't record the support count value. The structure of an ordinary node is $\langle item \rangle$. The tail node represents the last item found in the sorted transaction. The structure of the tail node is $\langle item : ts\text{-}list \rangle$, where **ts-list** represents the list of timestamps in which the corresponding event has occurred in a series. In other words, the tail node in the G3P-Tree is simply an ordinary node that includes a timestamp list (*ts-list*). Similar to an FP-Tree, each node in the G3P-Tree stores parent, children, and traversal pointers. However, unlike the FP-Tree, no node in the G3P-Tree contains a support count value, regardless of its node type. During the bottom-up mining phase of pattern-growth, the ts-list is pushed to parent nodes to determine the $period\text{-}support$ and *periodic-confidence* of the patterns. Like in an FP-tree, each node in G3P-tree maintains parent, children, and node traversal pointers. The key difference between an FP-tree and G3P-tree is that no node in G3P-tree maintains the support count as in an FP-tree. To facilitate a high degree of compactness, events in the prefix-tree are arranged in support-descending order.

Construction of G3P-tree

Since the proposed patterns satisfy the *convertible anti-monotonic property*, partial periodic items (PPIs) will play an important role in the efficient discovery of partial periodic patterns. Algorithm 1 describes the procedure to find PPIs using G3P-list. Fig 4.1(a), 4.1(b) and 4.1(c) show the G3P-list generated after scanning the first transaction, second transaction and entire database. As the proposed patterns satisfy the *convertible anti-monotonic property*, all items having *period-support* less than $minPS$ are pruned from the G3P-list. The final G3P-list generated after pruning the uninteresting items is shown in Fig 4.1(d). Let L denote the sorted list of PPIs in G3P-list.

(a)

I	s	ps	id_1	p-int
p	1	0	1	{[1,1]}
r	1	0	1	{[1,1]}
s	1	0	1	{[1,1]}

(b)

I	s	ps	id_1	p-int
p	2	1	3	{[1,3]}
r	2	1	3	{[1,3]}
s	1	0	1	{[1,1]}
q	1	0	3	{[3,3]}
t	1	0	3	{[3,3]}

(c)

I	s	ps	id_1	p-int
p	11	9	16	{[1,4],[7,16]}
r	8	5	16	{[1,4],[12,16]}
s	10	6	16	{[4,10]}
q	7	4	16	{[3,4],[12,16]}
v	3	1	15	{[6,7]}
u	4	2	12	{[8, 12]}
t	4	2	11	{[3,6]}

(d)

I	s	ps	id_1	p-int
p	11	9	16	{[1,4],[7,16]}
s	10	6	16	{[4, 10]}
r	8	5	16	{[1,4],[12,16]}
q	7	4	16	{[3,4],[12,16]}
u	4	2	12	{[8,12]}
t	4	2	11	{[3,6]}

Figure 4.1: Construction of G3P-list. (a) After scanning the first transaction, (b) after scanning the second transaction, (c) after scanning the entire database, and (d) the final G3P-list after pruning all aperiodic items

Algorithm 1 G3P-List (D: temporal database, I: set of items, $maxIAT$: maximum inter-arrival time, $minPS$: minimum period-support and $minPC$: minimum periodic-confidence)

Let id_l be a temporary array that records the *timestamp* of the last appearance of each item in S. Let ts_{cur} denote the current timestamp of a transaction. Let $[ts_a, ts_b]$ denote the last time interval recorded in *p-list*.
for each transaction $t \in D$ **do**
    for each item $i \in t$ **do**
        if i exists in G3P-list **then**
            ++$s(i)$.
            if $ts_{cur} - id_l(i) \leq maxIAT$ **then**
                ++$ps(i)$
                Set the last list of $p\text{-}list(i)$ as $[ts_a, ts_{cur}]$.
            else
                if $ts_a == ts_b$ **then**
                    Replace the previous ts_a value in the last list of $p\text{-}list(i)$ with $[ts_{cur}, ts_{cur}]$.
                else
                    Add another list entry into the $p\text{-}list(i)$ with $[ts_{cur}, ts_{cur}]$.
                end if
            end if
        else
            Add i to the G3P-list with $S(i) = 1$, $ps(i) = 0$ and $id_l(i) = ts_{cur}$.
            Add an entry, $[ts_{cur}, ts_{cur}]$, into the $p\text{-}list(i)$.
        end if
    end for
end for
Prune all uninteresting items from the list with *period-support* less than $minPS$.

Next, we construct the prefix-tree of the G3P-tree by performing another scan on the temporal database. Algorithms 2 and 3 describe the procedure for constructing the prefix-tree. Fig 4.2(a) shows the prefix-tree constructed after scanning the first transaction. It can be observed that though s is a PPI, it is not considered in the creation of a branch. The reason is that the occurrence of s is aperiodic in the first transaction. Based on the s's p-list in Fig 4.1(d), it can be said that the occurrence of item s in Table 1.2 will only be considered in the transactions whose timestamps range from 4 to 9. Fig 4.2(b) shows the prefix-tree generated after scanning the second transaction. A similar process is repeated for the remaining transactions, and the prefix-tree is updated accordingly. The final G3P-tree generated after scanning the entire database is shown in Fig 4.2(c).

Algorithm 2 G3P-Tree (D, G3P-list)

1: Create the root node in G3P-tree, $Tree$, and label it "$null$".
2: **for** each transaction $t \in D$ **do**
3: Select the PPIs in t and sort them in L order. Let the sorted list be $[e \mid E]$, where e is the first item, and E is the remaining list. Call $insert\_tree([e \mid E], ts_{cur}, Tree)$.
4: **end for**
5: call G3P-growth ($Tree$, $null$);

Algorithm 3 insert_tree($[e \mid E]$, ts_{cur}, $Tree$)

1: **while** E is non-empty **do**
2: **if** $Tree$ has a child N such that $e.item \neq N.item$ and $e.timestamp \neq N.timestamp$ **then**
3: Create a new node N. Let its parent-node be linked to $Tree$. Let its node-link be linked to only those nodes that have the same item via the node-link structure. Remove e from E.
4: **end if**
5: **end while**
6: Add ts_{cur} to the leaf node.

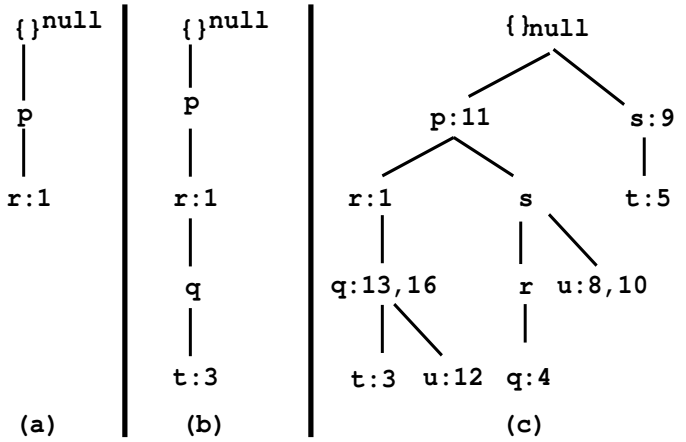


Figure 4.2: Construction of G3P-tree. (a) After scanning the first transaction, (b) After scanning the second transaction, and (c) After scanning the entire database

Recursive mining of G3P-tree

Due to the structural differences between nodes in an FP-tree and G3P-tree, we cannot directly apply FP-growth to mine a G3P-tree. Therefore, we developed another pattern-growth technique that can handle the additional features of G3P-tree. Algorithm 4 describes the procedure for finding partial periodic patterns from the G3P-tree. In this thesis, we are not explaining the procedure for calculating *period-support* and *periodic-confidence* of a pattern as they are simple procedures.

Algorithm 4 G3P-growth ($Tree$, α)

1: **while** items in the header of $Tree$ **do**
2: Generate pattern $\beta = i \cup \alpha$. Traverse $Tree$ using the node-links of β, and construct an array, TS^{β}, which represents the list of timestamps in which β has appeared periodically in D. Construct β's conditional pattern base and β's conditional G3P-tree $Tree_{\beta}$ if *period-support* is greater than or equal to $minPS$, *periodic-confidence* is greater than or equal to $minPC$.
3: **if** $Tree_{\beta} \neq \emptyset$ **then**
4: call G3P-growth ($Tree_{\beta}$, β);
5: **end if**
6: Remove i from the $Tree$ and push the i's ts-list to its parent nodes.
7: **end while**

We start with the bottom-most item in the G3P-list. Since t is the bottom-most item, each node with $item = t$ in G3P-tree must be a tail-node. While constructing the conditional pattern base of $\langle t \rangle$, we map the ts-list of every node $\langle t \rangle$ to all items in the respective path explicitly in a temporary array (one for each item). This facilitates the calculation of $periodic\ confidence$ for each item in the conditional

pattern base of $\langle t \rangle$. Fig 4.3(a) shows the prefix tree for $\langle t \rangle$. As no item in the prefix tree of t has *periodic-confidence* more than the user-specified $minPC$, the conditional tree of t will be empty as shown in Fig 4.3(b). Only t will be generated as a partial periodic pattern. Next, the item t will be completely pruned from the original G3P-tree by pushing its ts-lists to the respective parent nodes. Fig 4.3(c) shows the resultant G3P-tree after pruning item t. The entire process is repeated until the G3P-list in the G3P-tree is empty.

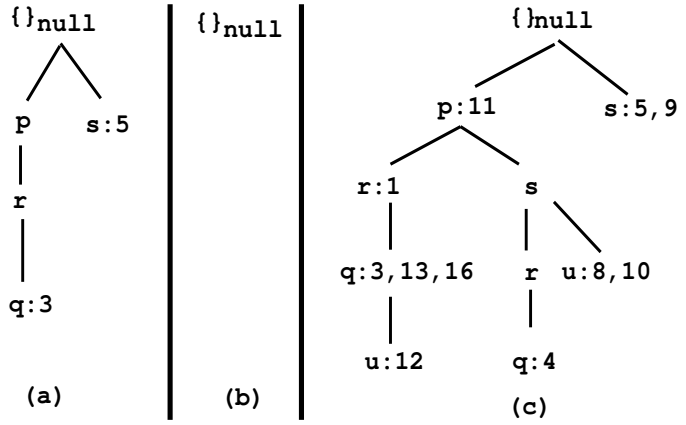


Figure 4.3: Recursive mining of G3P-tree. (a) Prefix tree for item t (b) conditional tree for item t and (c) G3P-tree after pruning item t

4.4 Summary

In this chapter, we have explained the proposed model for effectively mining 3Ps. We have introduced a novel measure named $periodic\ confidence$, which satisfies both null-invariant and anti-monotonic properties to determine the interestingness of the patterns. We have also proposed a pattern growth algorithm named G3P-Growth *(Generalized Partial Periodic Pattern Growth)*, which uses the novel concept of *irregularity pruning* to reduce the runtime and memory requirements of the algorithm. In the next chapter, we conduct experiments to demonstrate the scalability and efficiency of our model and explain the case studies.

Chapter 5

Experimental Evaluation

In this chapter, we present the experimental study. we first present the description of the datasets. Next, we present the experimental results. We also demonstrate the usefulness of our model with two case studies. Lastly, we provide the summary of the chapter.

5.1 Dataset Description

The G3P-growth algorithm was written in Python 3 and executed on a machine with a 2.5 GHz processor and 8 GB RAM. The experiments have been conducted on synthetic datasets (**T10I4D100K**, **T10I4D1000K** and **T20I6D100K**) and real-world (**Pollution**, **Congestion**, **BMS-WebView-1** and **BMS-WebView-2**) databases. To verify the repeatability of the experiments, the source code and datasets are provided at [65]

The **T10I4D100K** and **T20I6D100K** [64] are widely used synthetic databases for evaluating frequent pattern mining algorithms. These databases are transactional in nature, and for evaluation purposes, they are converted into a temporal database by considering $tids$ as timestamps. The T10I4D100K database contains 870 items and 100,000 transactions, while the T20I6D100K database contains 893 items and 99,845 transactions. The **T10I4D1000K** is another very large synthetic database containing 30387 items and 983,155 transactions. We use this database to demonstrate the scalability of our algorithm.

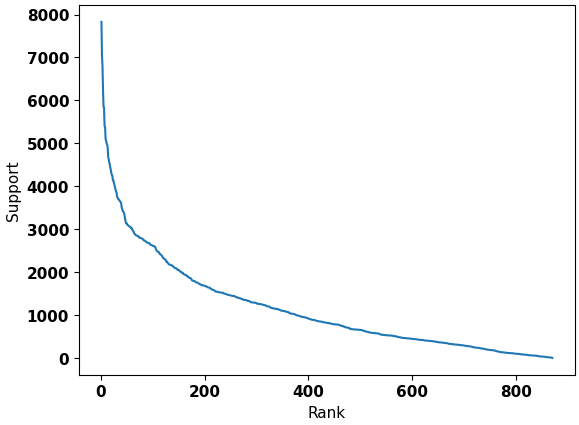
The **Pollution** database was downloaded from AEROS [66] website. AEROS is the Atmospheric Environmental Regional Observation System managed by the Ministry of the Environment Air Pollutant Wide Area Monitoring System, Japan. The database comprises sensor identifiers whose PM2.5[1] value is no less than 16 $\mu g/m^3$. The time period for the collection of data is from 1-April-2018 to 30-April-2018 and is collected at hourly intervals. It contains 1029 items and 719 transactions. The minimum, average and maximum transaction lengths are 11, 460 and 971, respectively.

The **Congestion** database was provided by JARTIC for Kobe, Japan. This database consists of road segment identifiers (or sensor ids) where the observed congestion is more than 300 meters. The data is

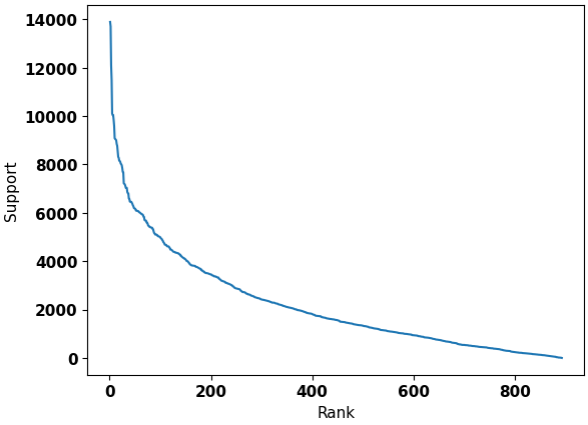
[1]Atmospheric particulate matter with a diameter of less than 2.5 micrometres

collected at 5-minute intervals. This data was collected on 17-July-2015. On this day, Kobe was struck by Typhoon Nangka. The database contains 938 items and 1439 transactions. The minimum, average and maximum transaction lengths are 11, 66.25 and 267, respectively.

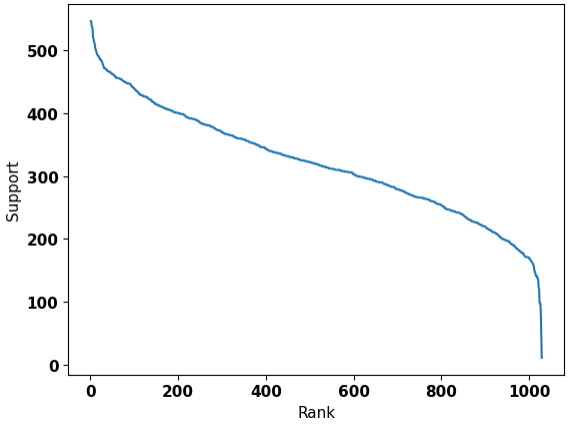
The **BMS-WebView-1** and **BMS-WebView2** are sparse real-world databases containing clickstream data from e-commerce sites. Each transaction is a viewing session consisting of all the viewed product detail pages where each product detail view is an item. These databases contain very long transactions, and they were used in KDD CUP 2000 competition [67]. These databases contain 497 and 3340 items and 59,602 and 77,512 transactions, respectively. The maximum length of the transaction in the database is 267 and 161, respectively.



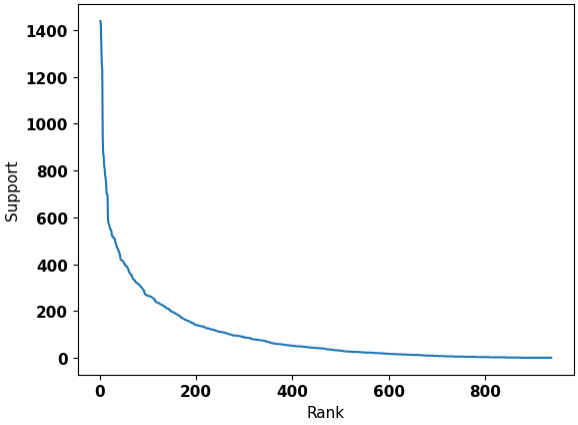
(a) T10I4D100K



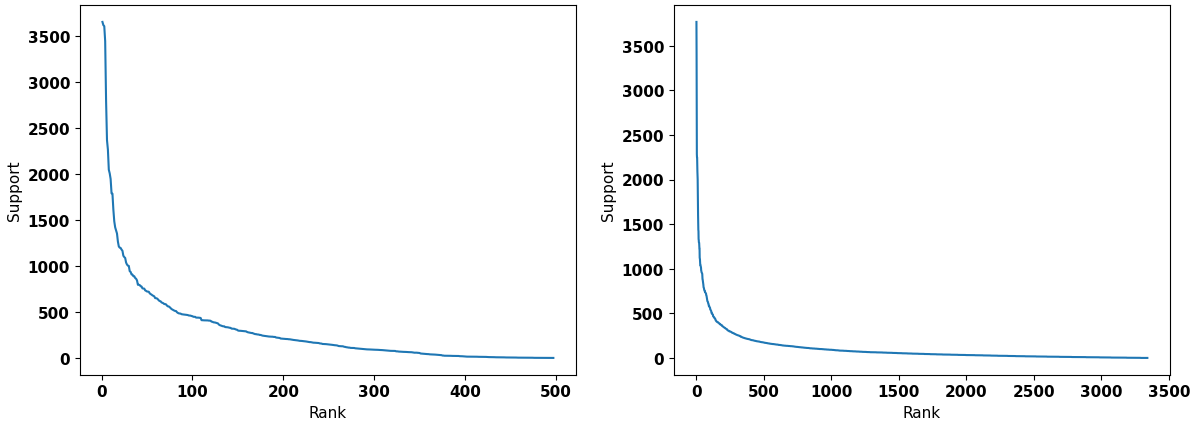
(b) T20I6D100K



(c) Pollution



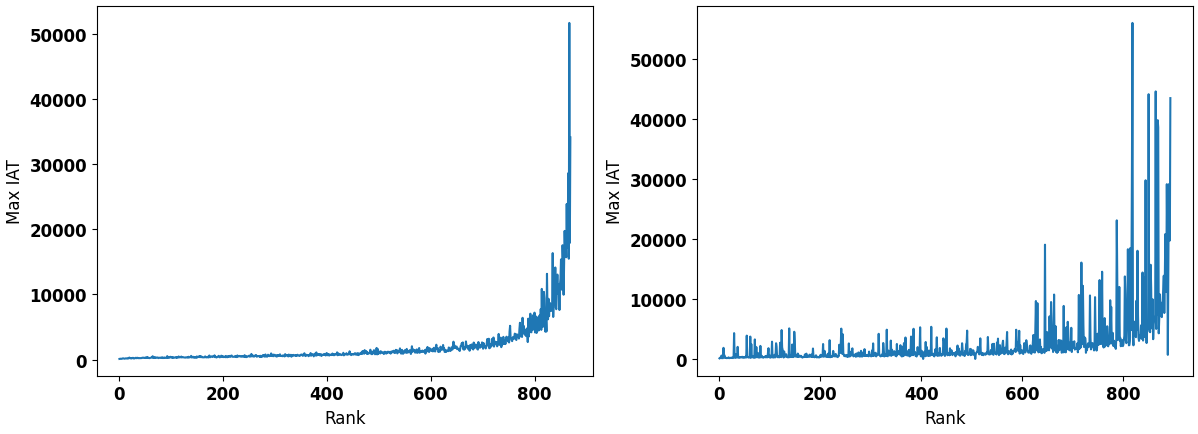
(d) Congestion



(e) BMS-WebView-1

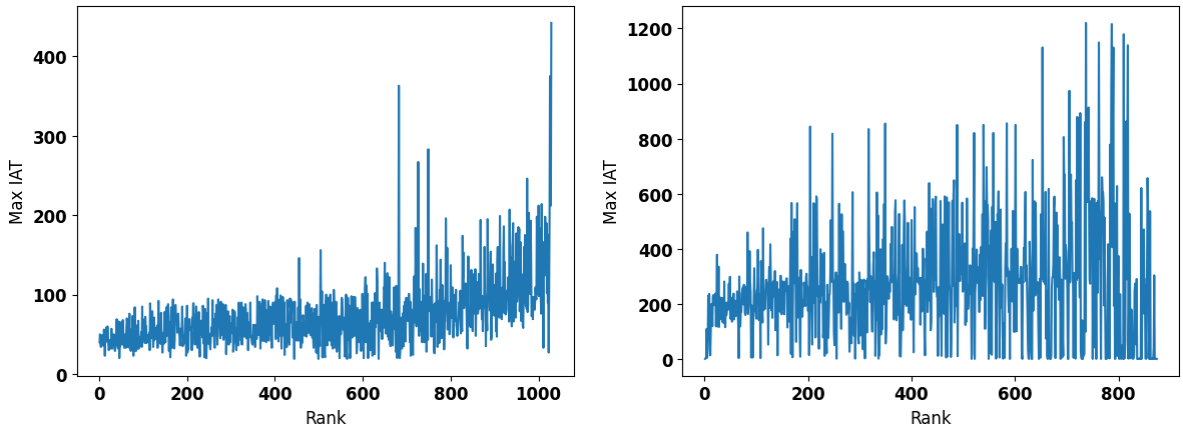
(f) BMS-WebView-2

Figure 5.1: Frequency Distribution of Datasets



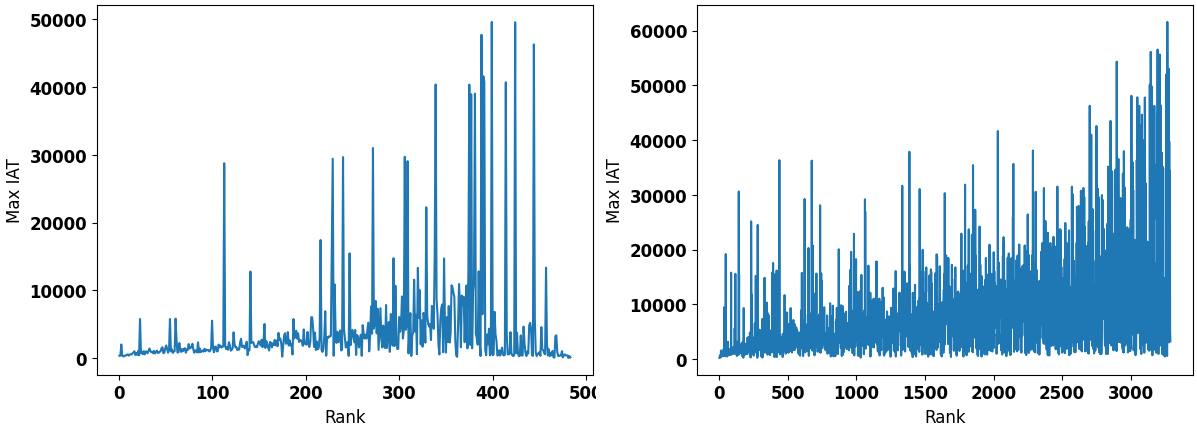
(a) T10I4D100K

(b) T20I6D100K



(c) Pollution

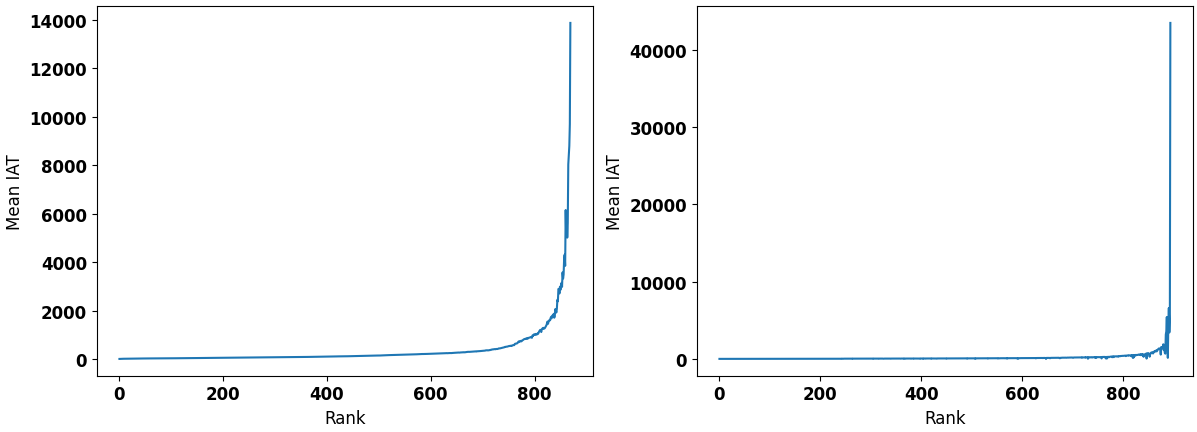
(d) Congestion



(e) BMS-WebView-1

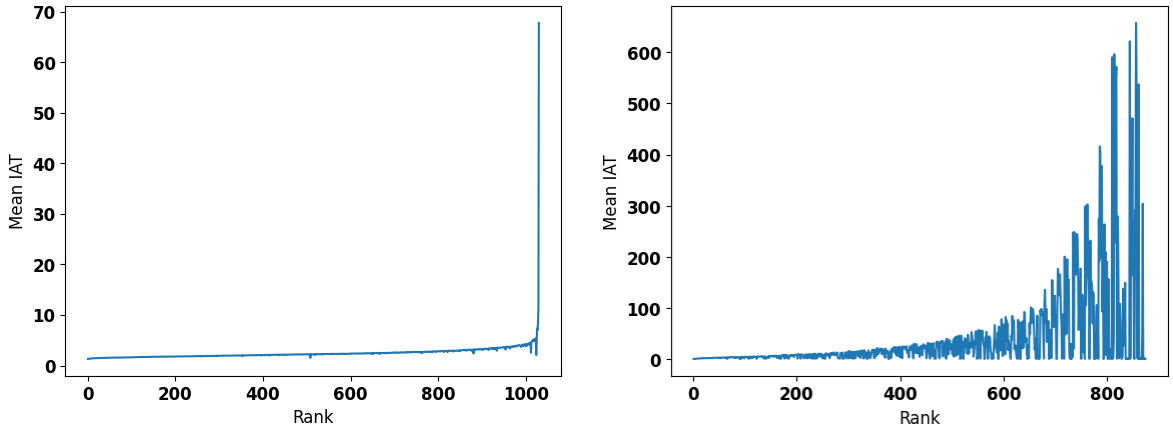
(f) BMS-WebView-2

Figure 5.2: Max IAT Distribution of Datasets



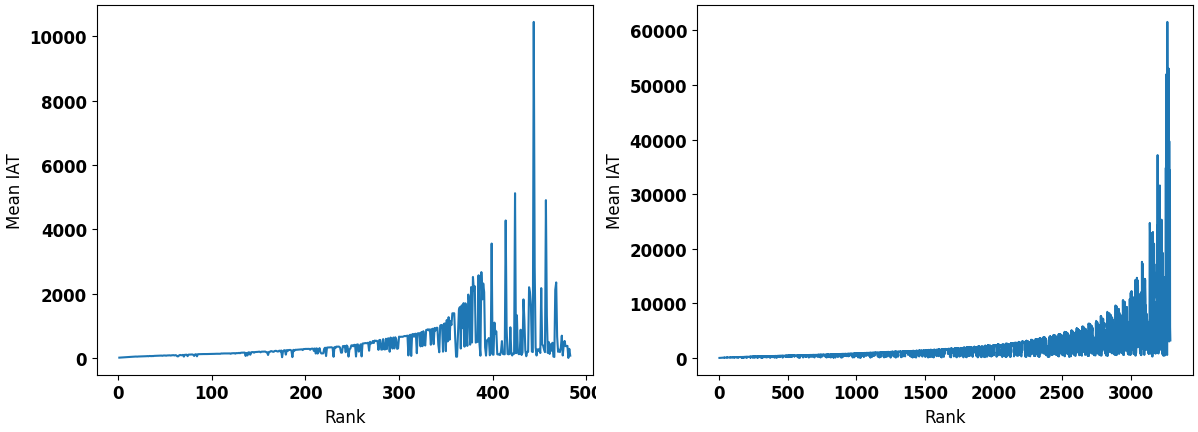
(a) T10I4D100K

(b) T20I6D100K



(c) Pollution

(d) Congestion



(e) BMS-WebView-1 (f) BMS-WebView-2

Figure 5.3: Mean IAT Distribution of Datasets

As illustrated in Fig. 5.1, the $support$ count values of the items within each database are arranged in descending order of their respective values. It is evident that the distribution of items in the majority of the databases does not conform to a uniform pattern, indicating that the data is reflective of real-world scenarios where certain items occur more frequently than others. Notably, a significant proportion of the datasets exhibit a long-tail distribution, with a small subset of items occurring frequently and the majority of the items occurring rarely within the database.

Figures 5.2 and 5.3 displays the maximum and average inter-arrival times, respectively, of items that have been arranged in descending order based on their $support$ values within each database. The figures highlight that the maximum and mean inter-arrival times increase as the $support$ count of items decreases, a phenomenon that is commonly observed in real-world data. Specifically, rare items tend to appear more sporadically within the dataset.

The description for each dataset is summarized in Table 5.1

S.No	Dataset name	N. of transactions	N. of Distinct items	Max Trans. Len.	Avg. Trans. Len.
1	T10I4D100K	100000	870	29	10.1
2	T20I6D100K	99845	893	47	20
3	Pollution	719	1029	971	459.3
4	Congestion	1439	938	267	66.2
5	BMS-WebView-1	59602	497	267	2.5
6	BMS-WebView-2	77512	3340	161	4.6
7	T10I4D1000K	983155	30387	31	10.2

Table 5.1: Dataset Description

5.2 Experimental Results

Since there exists no algorithm to find partial periodic patterns that may exist in a temporal database and mine patterns containing both rare and frequent items, we only evaluate the proposed G3P-growth algorithm to show that it is not only memory and runtime efficient but also highly scalable as well. Additionally, we attempted to compare it with the 3P algorithm, but due to the immense space requirements, the 3P algorithm took infinite time to process most of the datasets. This demonstrates that the existing algorithm is unfit for most real-world applications.

This section assesses how well the proposed model performs on various databases. The model relies on three user-defined parameters: the minimum periodic support ($minPS$), minimum periodic confidence threshold ($minPC$), and maximum interarrival time ($maxIAT$).

To gauge the model's effectiveness under different conditions, we measure its runtime requirements, memory usage, and the number of generated patterns across four real-world and two synthetic datasets with different characteristics. We conduct three experiments by varying two parameters while keeping the third one constant, as follows:

1. To analyze the effect of varying $minPS$ and $minPC$, while keeping $maxIAT$ constant.
2. To study the effect of varying $minPS$ and $maxIAT$, while keeping $minPC$ constant.
3. To examine the effect of varying $maxIAT$ and $minPC$, while keeping $minPS$ constant.

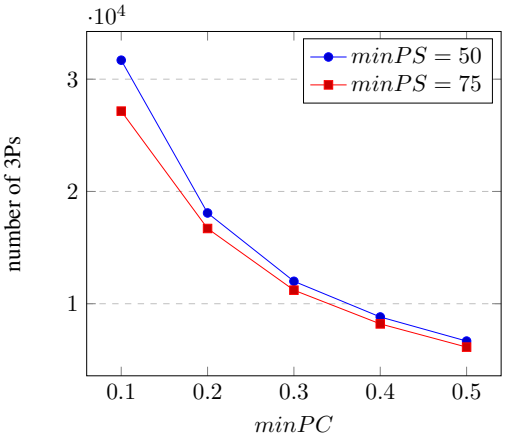
Please note the $minPS$ used in further subsections is specified in absolute terms and not in relative terms (or in %) due to the large sizes of databases.

5.2.1 Effect of varying $minPS$ and $minPC$ keeping $maxIAT$ constant

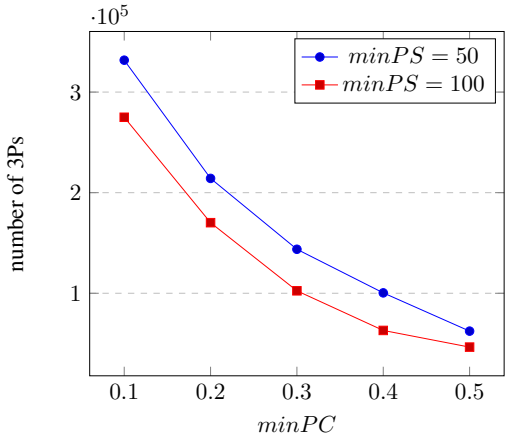
The parameters used for the experiment are summarised in the following table 5.2

S.No	**Dataset name**	$minPS$	$minPC$	$maxIAT$
1	T10I4D100K	{50, 75}	[0.1,0.5], SS = 0.1	5000
2	T20I6D100K	{50, 100}	[0.1,0.5], SS = 0.1	5000
3	Pollution	{350, 360}	[0.7, 0.9], SS = 0.05	5000
4	Congestion	{200, 220}	[0.4, 0.6], SS = 0.05	5
5	BMS-WebView-1	{29, 30}	[0.27, 0.31], SS = 0.01	5000
6	BMS-WebView-2	{8, 10}	[0.09, 0.13], SS = 0.01	5000

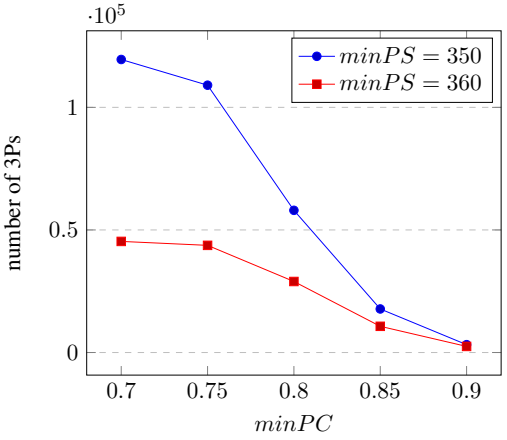
Table 5.2: Parameters used in the evaluation of varying $minPS$ and $minPC$ keeping $maxIAT$ constant (SS denotes the step size between the values of $minPC$)



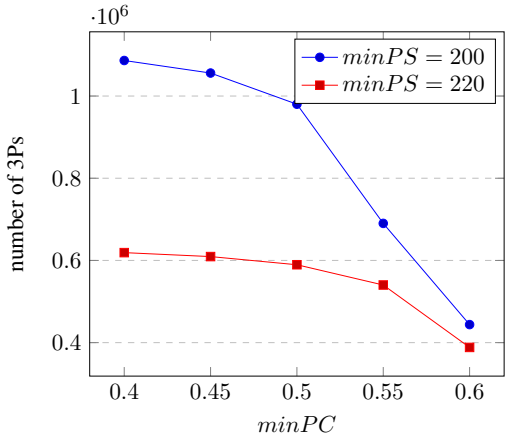
(a) T10I4D100K



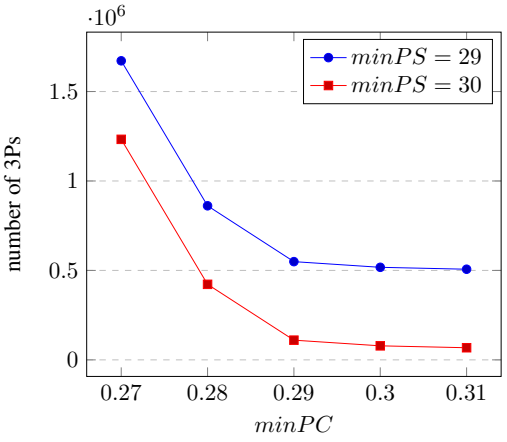
(b) T20I6D100K



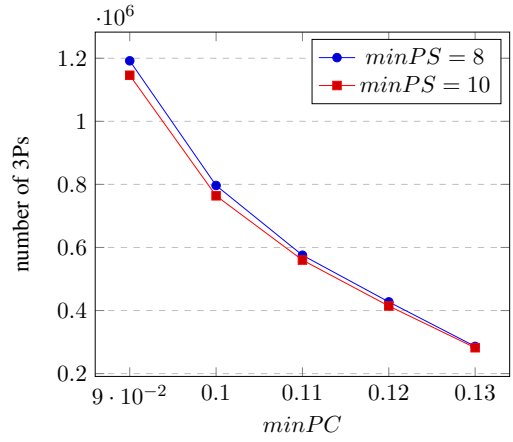
(c) Pollution



(d) Congestion

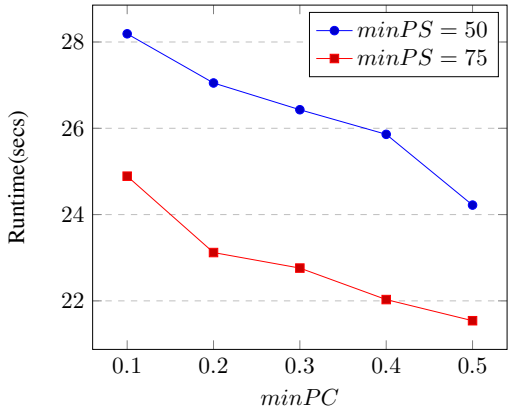


(e) BMS-WebView-1

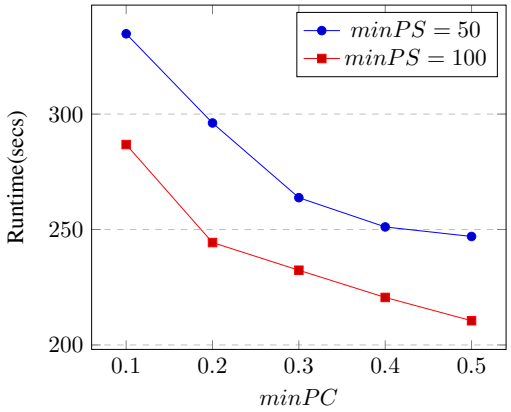


(f) BMS-WebView-2

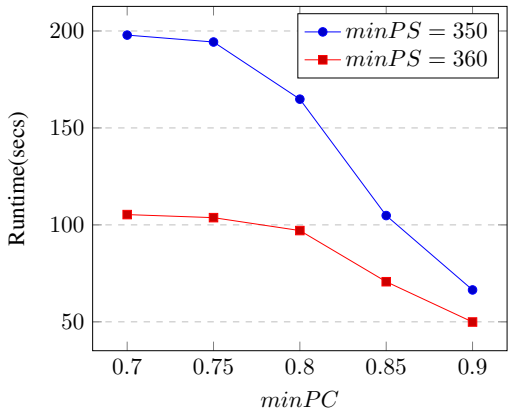
Figure 5.4: Number of patterns generated varying $minPS$ and $minPC$ values



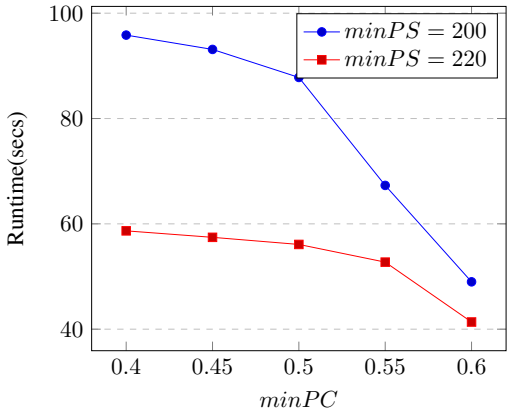
(a) T10I4D100K



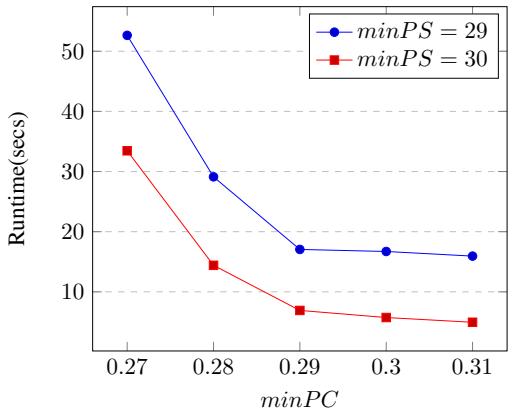
(b) T20I6D100K



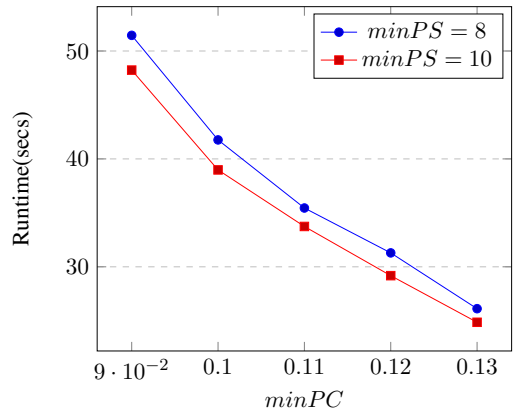
(c) Pollution



(d) Congestion

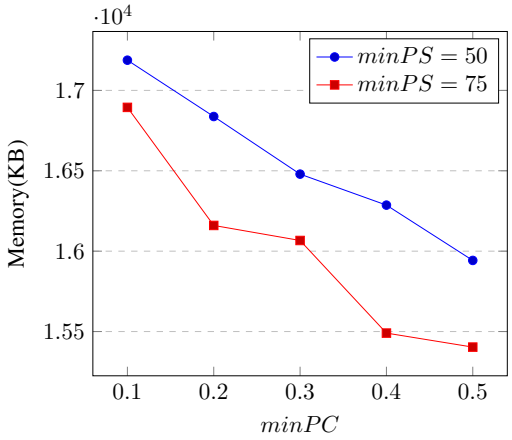


(e) BMS-WebView-1

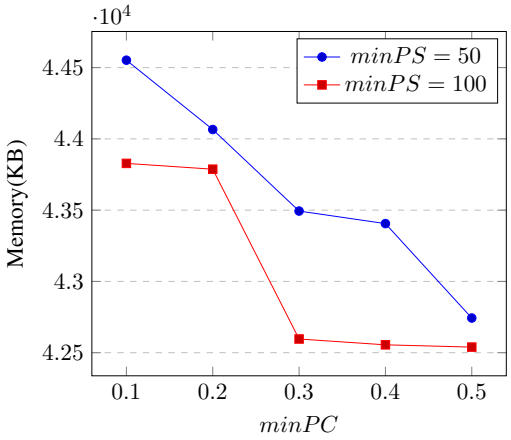


(f) BMS-WebView-2

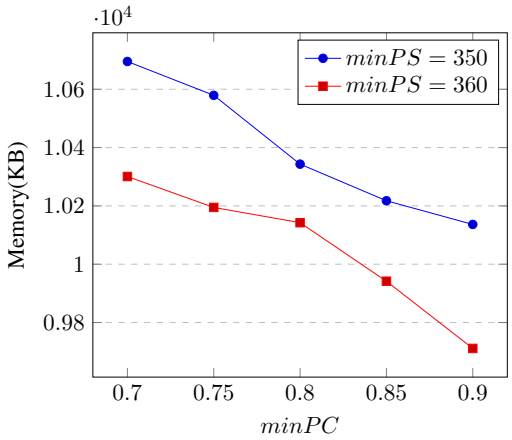
Figure 5.5: Runtime requirements of G3P-growth on varying $minPS$ and $minPC$



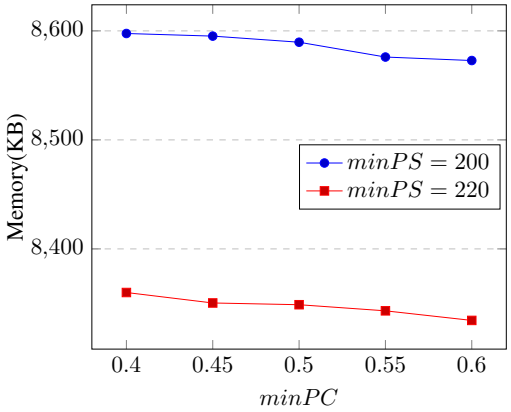
(a) T10I4D100K



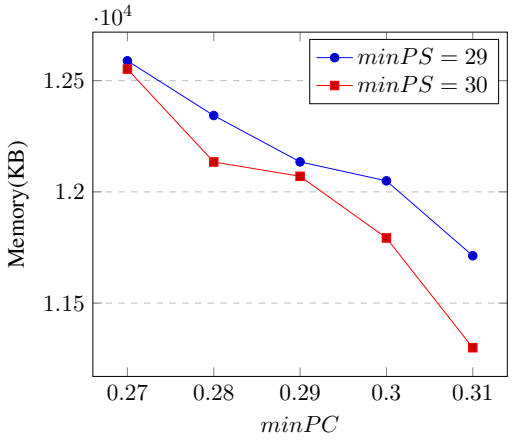
(b) T20I6D100K



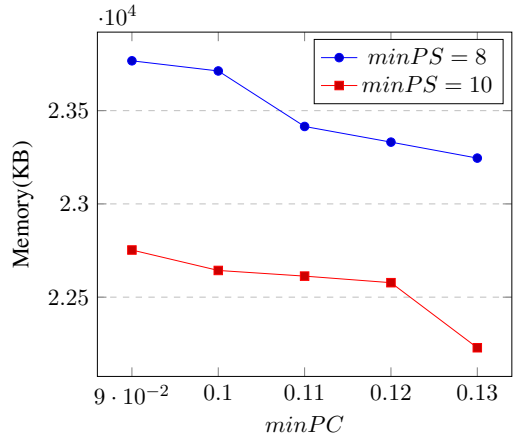
(c) Pollution



(d) Congestion



(e) BMS-WebView-1



(f) BMS-WebView-2

Figure 5.6: Memory used for the G3P-tree construction on varying $minPS$ and $minPC$

Fig. 5.4 depicts the variations in the number of partial periodic patterns (3Ps) generated on varying $minPS$ and $minPC$ for different datasets. Based on the results for all the datasets, it can be observed that increase in $minPC$ and/or $minPS$ have a negative effect on the generation of patterns. It is because many patterns fail to satisfy the increased $minPS$ and/or $minPC$ values.

Fig. 5.5 depicts the effect of varying $minPS$ and $minPC$ on runtime requirements of the G3P-growth algorithm. Based on the results for all the datasets, it can be observed that increase in $minPC$ and/or $minPS$ have a negative effect on the generation of patterns. It is because many patterns fail to satisfy the increased $minPS$ and/or $minPC$ values. The following observations can be drawn from these figures: (i) increase in $minPS$ and/or $minPC$ values may decrease the runtime requirements of G3P-growth. It is because G3P-growth has to find fewer patterns due to the reduced number of periodic items. (ii) As per as the database size and reasonably low $minPS$ and low $minPC$ values are concerned, it can be observed that mining patterns from the corresponding G3P-tree is rather time efficient (or practicable) for both synthetic and real-world databases.

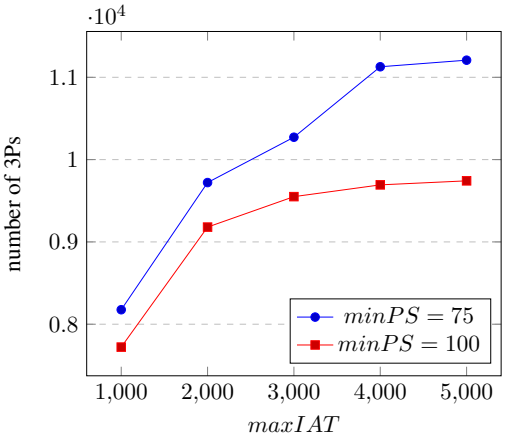
Fig. 5.6 depicts the changes in memory used for constructing G3P-tree and mining conditional patterns on varying $minPS$ and $minPC$ for different datasets. Based on the results for all the datasets, the following two observations can be drawn from these sub-figures: (i) increase in $minPS$ may decrease the memory requirements of G3P-tree construction. It is because many items fail to satisfy the $minPS$ and require mining of a lesser number of patterns. (ii) increase in $minPC$ has minimal effect on the memory requirements of the G3P-tree. It is because G3P-tree is constructed by periodic items.

5.2.2 Effect of varying $minPS$ and $maxIAT$ keeping $minPC$ constant

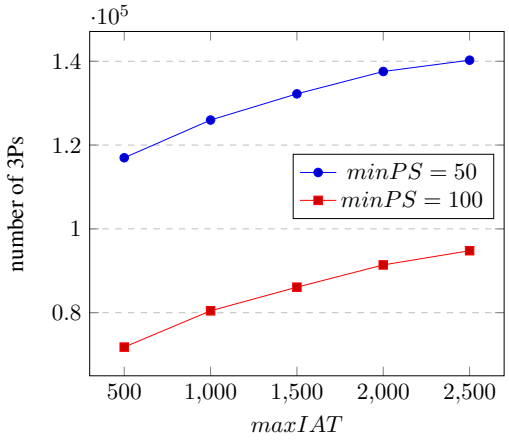
The parameters used for the experiment are summarised in the following table 5.3

S.No	**Dataset name**	$minPS$	$minPC$	$maxIAT$
1	T10I4D100K	{75, 100}	0.3	[1000, 5000], SS = 1000
2	T20I6D100K	{50, 100}	0.3	[500, 2500], SS = 500
3	Pollution	{350, 360}	0.7	[20, 100], SS = 20
4	Congestion	{200, 220}	0.3	[20, 100], SS = 20
5	BMS-WebView-1	{28, 29}	0.3	[500, 2500], SS = 500
6	BMS-WebView-2	{8, 10}	0.1	[500, 2500], SS = 500

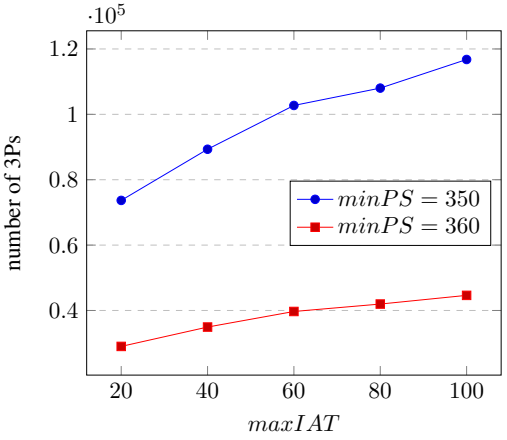
Table 5.3: Parameters used in the evaluation of varying $minPS$ and $maxIAT$ keeping $minPC$ constant (SS denotes the step size between the values of $maxIAT$)



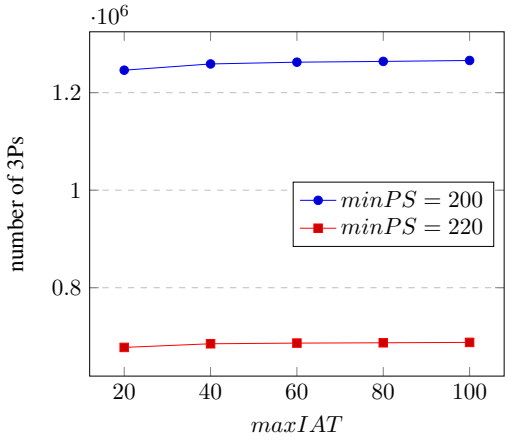
(a) T10I4D100K



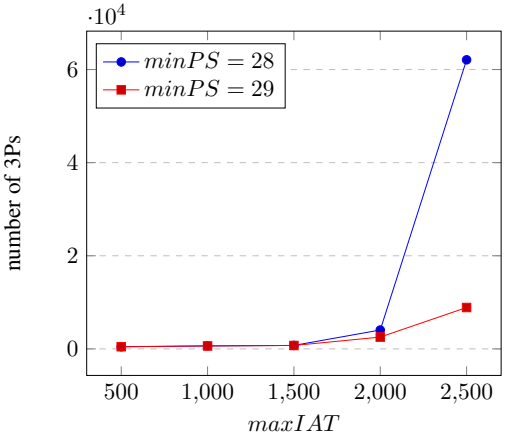
(b) T20I6D100K



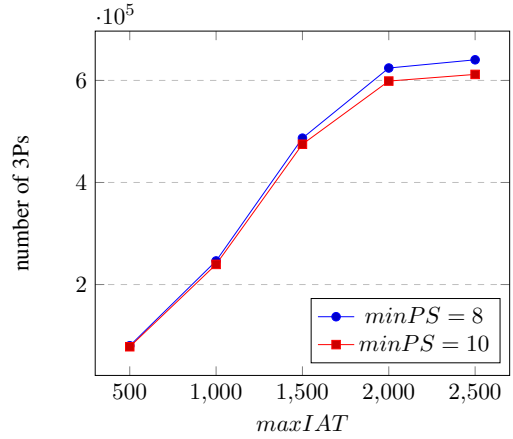
(c) Pollution



(d) Congestion

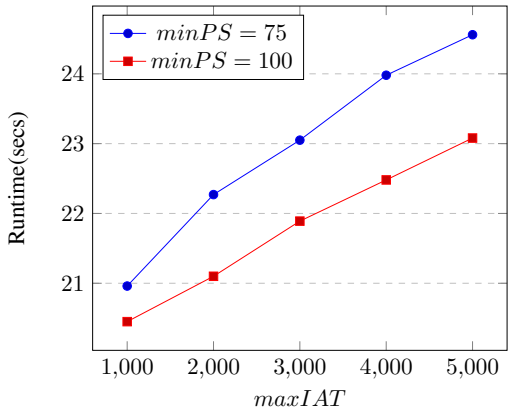


(e) BMS-WebView-1

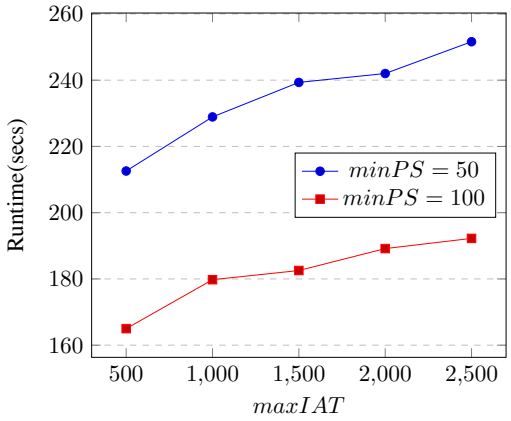


(f) BMS-WebView-2

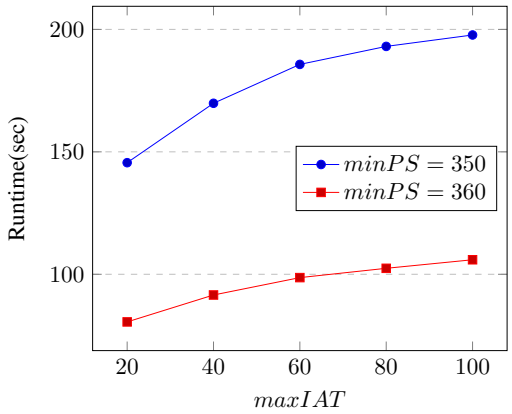
Figure 5.7: Number of patterns generated varying $minPS$ and $maxIAT$ values



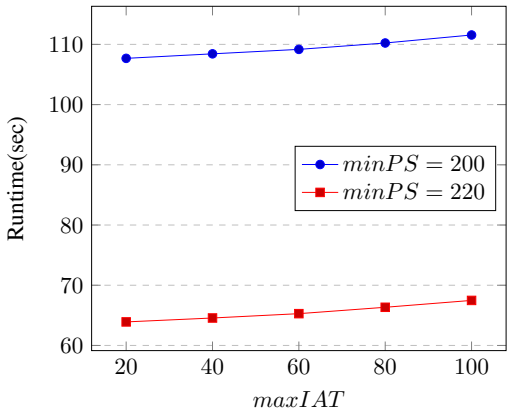
(a) T10I4D100K



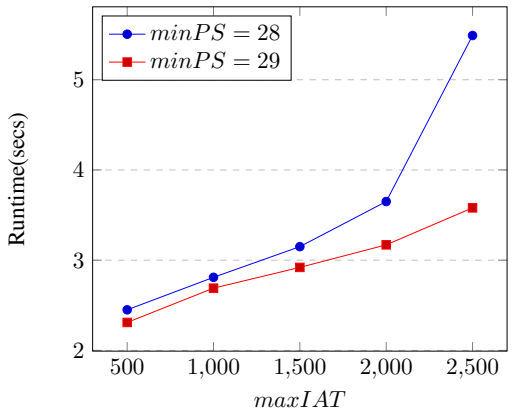
(b) T20I6D100K



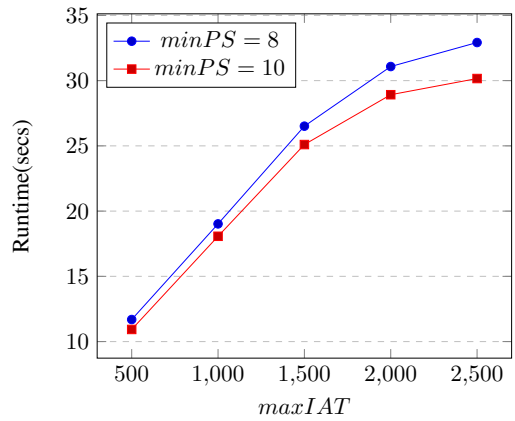
(c) Pollution



(d) Congestion

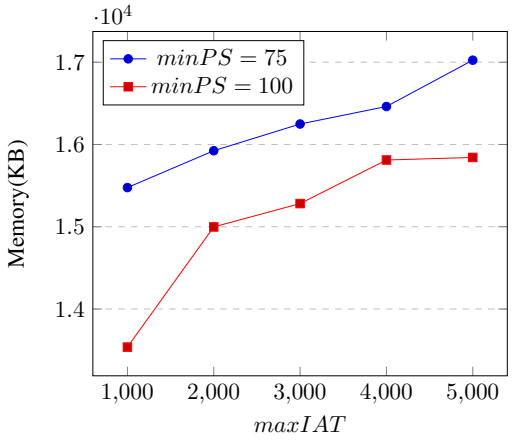


(e) BMS-WebView-1

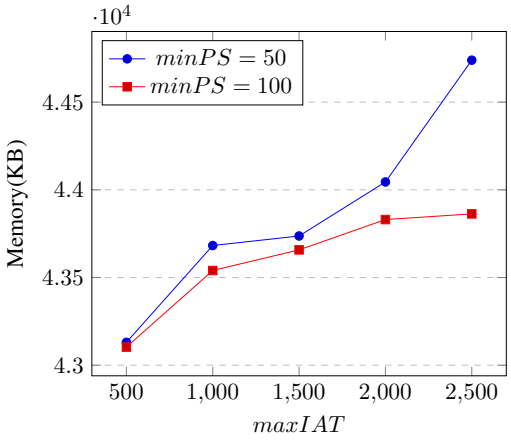


(f) BMS-WebView-2

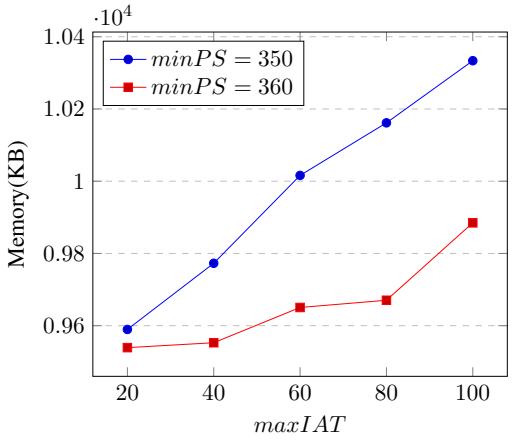
Figure 5.8: Runtime requirements of G3P-growth on varying $maxIAT$ and $minPS$



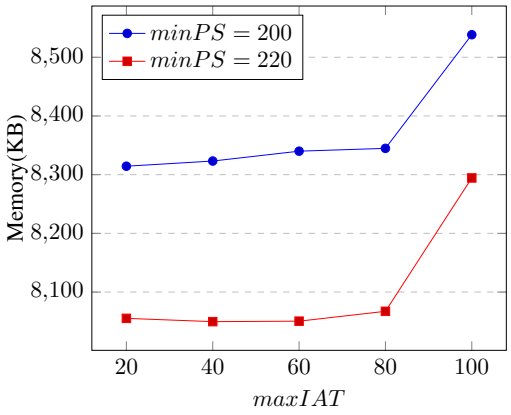
(a) T10I4D100K



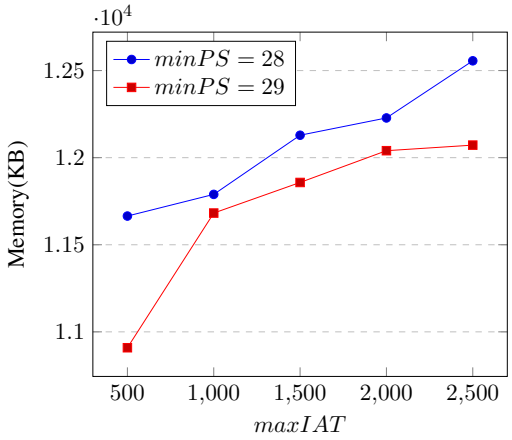
(b) T20I6D100K



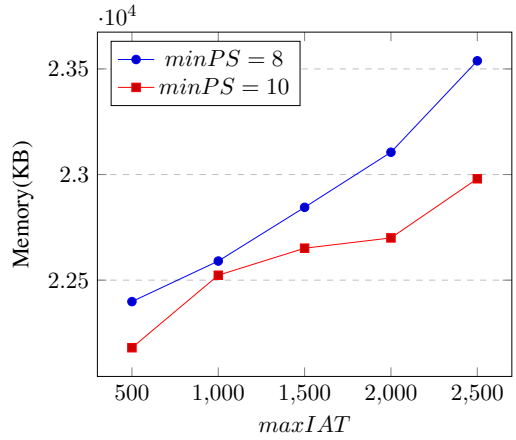
(c) Pollution



(d) Congestion



(e) BMS-WebView-1



(f) BMS-WebView-2

Figure 5.9: Memory used for the G3P-tree construction on varying $minPS$ and $maxIAT$

Fig. 5.7 represents the variations in the number of partial periodic patterns (3Ps) generated at different $minPS$ and $maxIAT$ values for different datasets. The following observations can be drawn from the sub-figures: (i) With the increase in $maxIAT$, there is an increase in the number of partial periodic patterns generated. It is because many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$. (ii) The increase in $minPS$ has a negative effect on the generation of patterns as many patterns fail to satisfy the increased $minPS$ values.

Fig. 5.8 represents the effect of varying $minPS$ and $maxIAT$ values on the runtime requirements of the G3P-growth algorithm for different datasets. The following observations can be drawn from the sub-figures: (i) With the increase in $maxIAT$, many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$ which increases the number of partial periodic patterns generated and leads to increase in the runtime requirements. It is primarily because the G3P-growth has to discover more patterns (ii) Increase in the $minPS$ value may decrease the runtime requirements of G3P-growth. It is because G3P-growth has to find fewer patterns as the number of items satisfying the criteria for periodic would decrease.

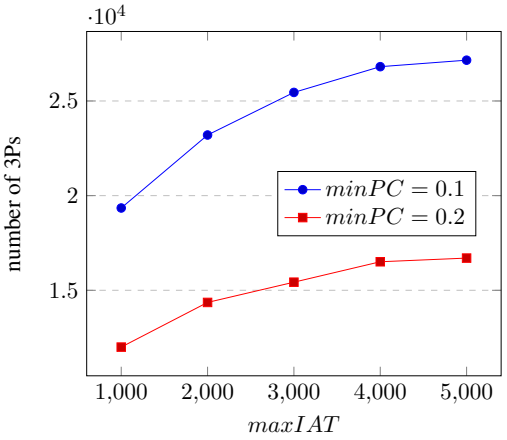
Fig. 5.9 represents the changes in memory used for constructing G3P-tree and mining conditional patterns on varying $minPS$ and $maxIAT$ for different datasets. The following two observations can be drawn from these sub-figures: (i) increase in $minPS$ may decrease the memory requirements of G3P-tree construction. It is because many items fail to be periodic items. (ii) An increase in $maxIAT$ may increase the memory requirements of G3P-Tree construction because many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$, and in the mining of patterns, more conditional G3P-Trees would have to be created.

5.2.3 Effect of varying $maxIAT$ and $minPC$ keeping $minPS$ constant

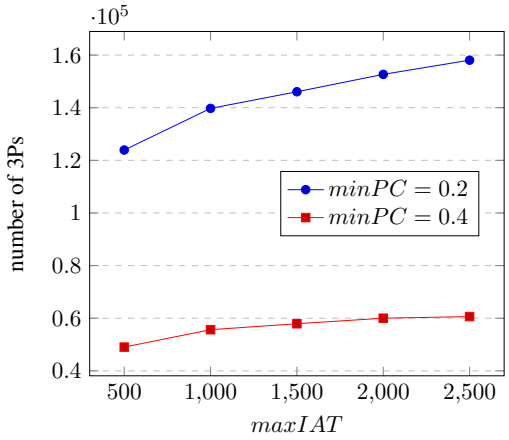
The parameters used for the experiment are summarised in the following table 5.4

S.No	**Dataset name**	$minPS$	$minPC$	$maxIAT$
1	T10I4D100K	75	{0.1, 0.2}	[1000, 5000], SS = 1000
2	T20I6D100K	100	{0.2, 0.4}	[500, 2500], SS = 500
3	Pollution	200	{0.5, 0.8}	[20, 100], SS = 20
4	Congestion	340	{0.5, 0.7}	[20, 100], SS = 20
5	BMS-WebView-1	30	{0.27, 0.28}	[1000, 3000], SS = 500
6	BMS-WebView-2	8	{0.1, 0.12}	[500, 2500], SS = 500

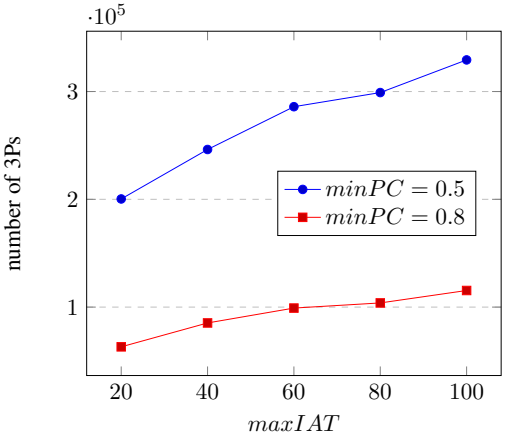
Table 5.4: Parameters used in the evaluation of varying $maxIAT$ and $minPC$ keeping $minPS$ constant (SS denotes the step size between the values of $maxIAT$)



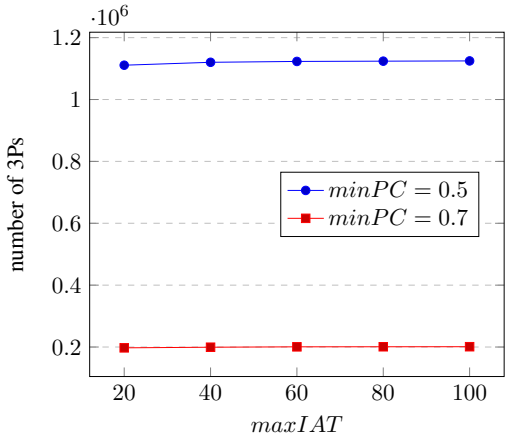
(a) T10I4D100K



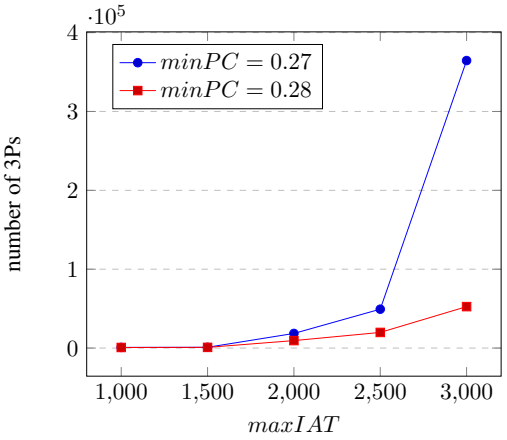
(b) T20I6D100K



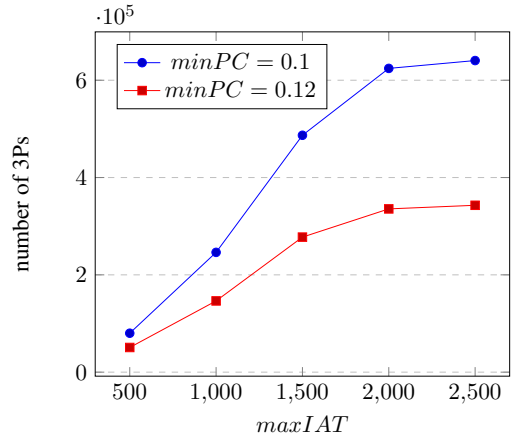
(c) Pollution



(d) Congestion

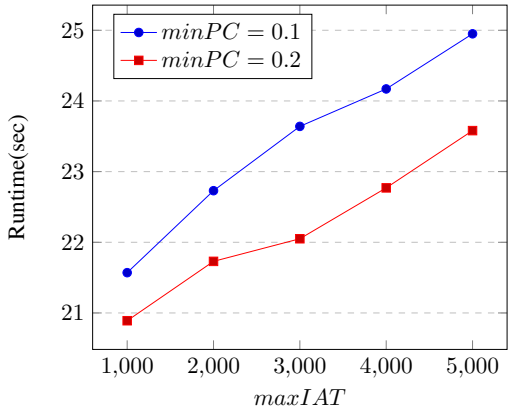


(e) BMS-WebView-1

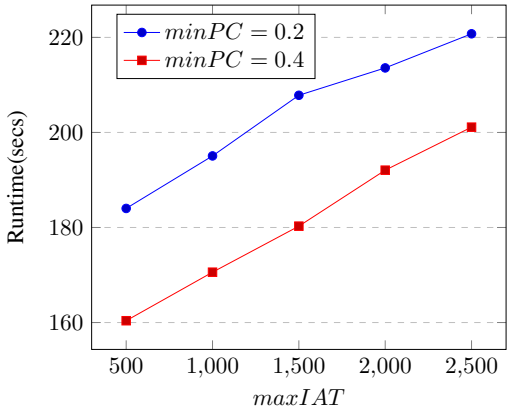


(f) BMS-WebView-2

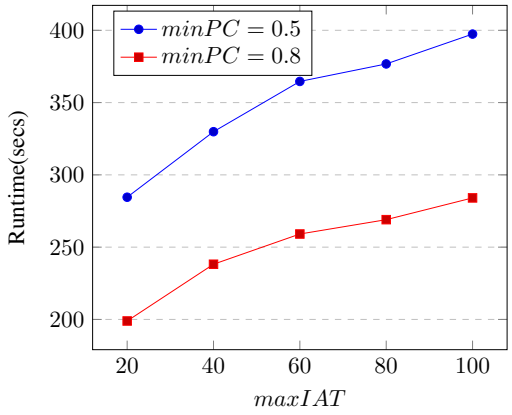
Figure 5.10: Number of patterns generated varying $maxIAT$ and $minPC$ values



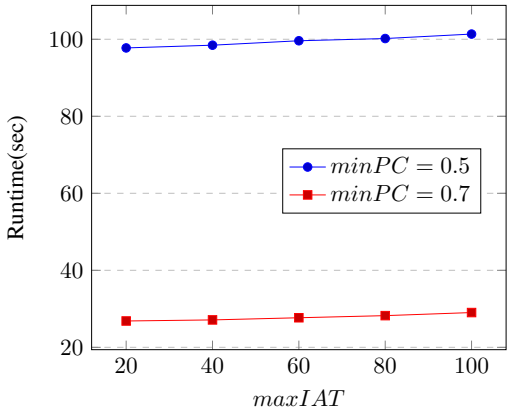
(a) T10I4D100K



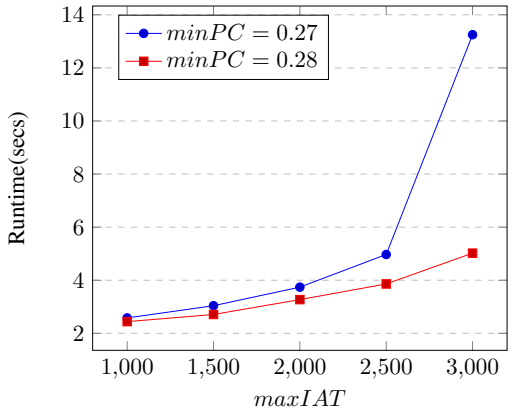
(b) T20I6D100K



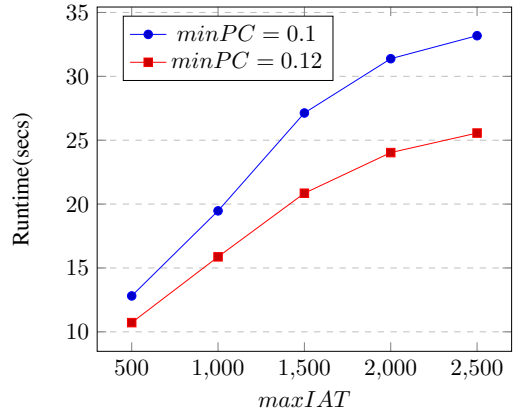
(c) Pollution



(d) Congestion

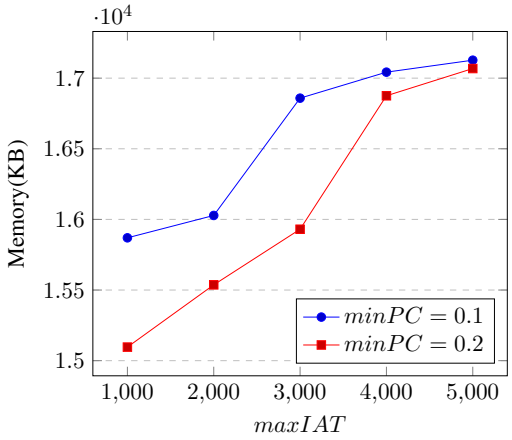


(e) BMS-WebView-1

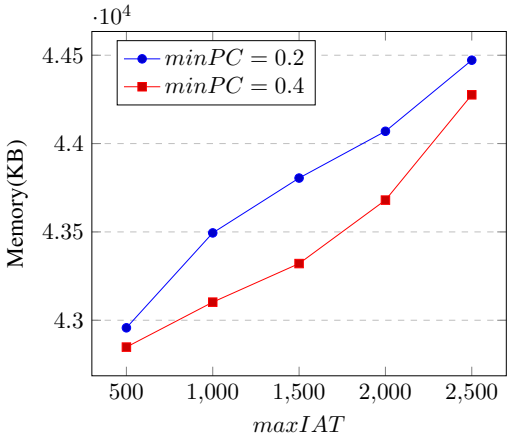


(f) BMS-WebView-2

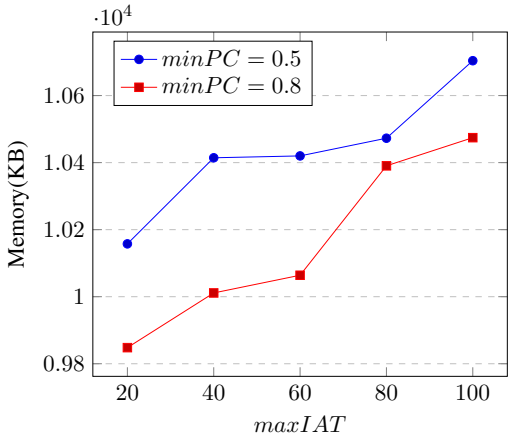
Figure 5.11: Runtime requirements of G3P-growth on varying $maxIAT$ and $minPC$



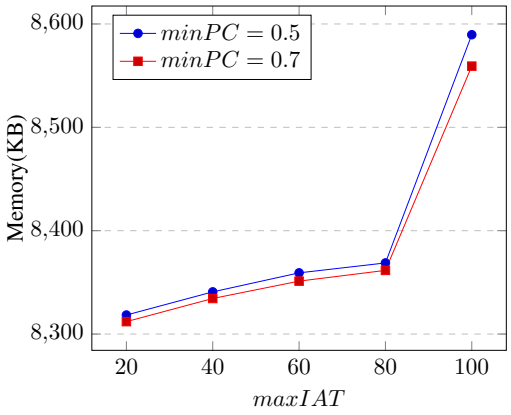
(a) T10I4D100K



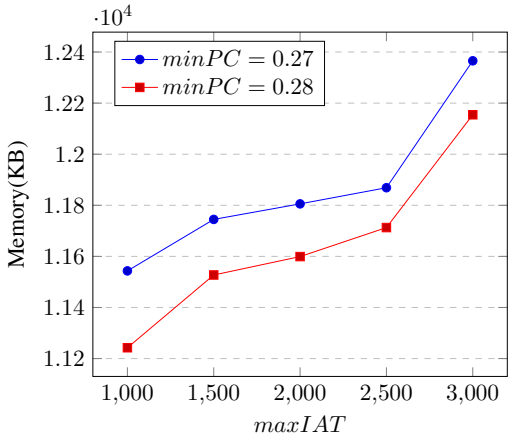
(b) T20I6D100K



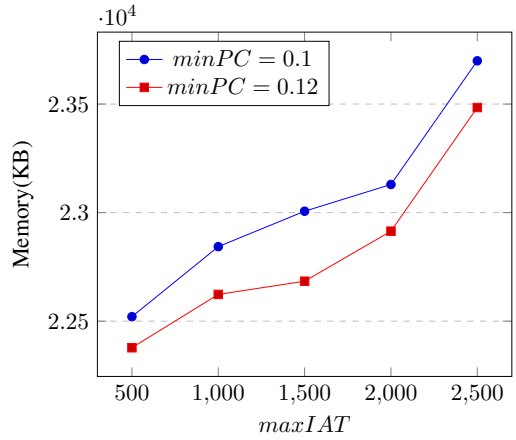
(c) Pollution



(d) Congestion



(e) BMS-WebView-1



(f) BMS-WebView-2

Figure 5.12: Memory used for the G3P-tree construction on varying $minPC$ and $maxIAT$

Fig. 5.10 represents the variations in the number of partial periodic patterns (3Ps) generated at different $minPC$ and $maxIAT$ values for different datasets. The following observations can be drawn from the sub-figures: (i) With the increase in $maxIAT$, there is an increase in the number of partial periodic patterns generated. It is because many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$. (ii) The increase in $minPC$ has a negative effect on the generation of patterns as many patterns fail to satisfy the increased $minPC$ values.

Fig. 5.11 represents the effect of varying $minPC$ and $maxIAT$ values on the runtime requirements of the G3P-growth algorithm for different datasets. The following observations can be drawn from the sub-figures: (i) With the increase in $maxIAT$, many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$ which increases the number of partial periodic patterns generated and leads to increase in the runtime requirements. It is primarily because the G3P-growth has to discover more patterns (ii) Increase in the $minPC$ value may decrease the runtime requirements of G3P-growth. It is because G3P-growth has to find fewer patterns.

Fig. 5.12 represents the changes in memory used for constructing G3P-tree and mining conditional patterns on varying $minPC$ and $maxIAT$ for different datasets. The following two observations can be drawn from these sub-figures: (i) increase in $minPC$ has minimal effect on the memory requirements of the G3P-tree. It is because G3P-tree is constructed by periodic items. (ii) An increase in $maxIAT$ may increase the memory requirements of G3P-Tree construction because many itemsets which appear sporadically satisfy the criteria of becoming periodic due to larger values of $maxIAT$, and in the mining of patterns, more conditional G3P-Trees would have to be created.

5.3 Scalability Experiment

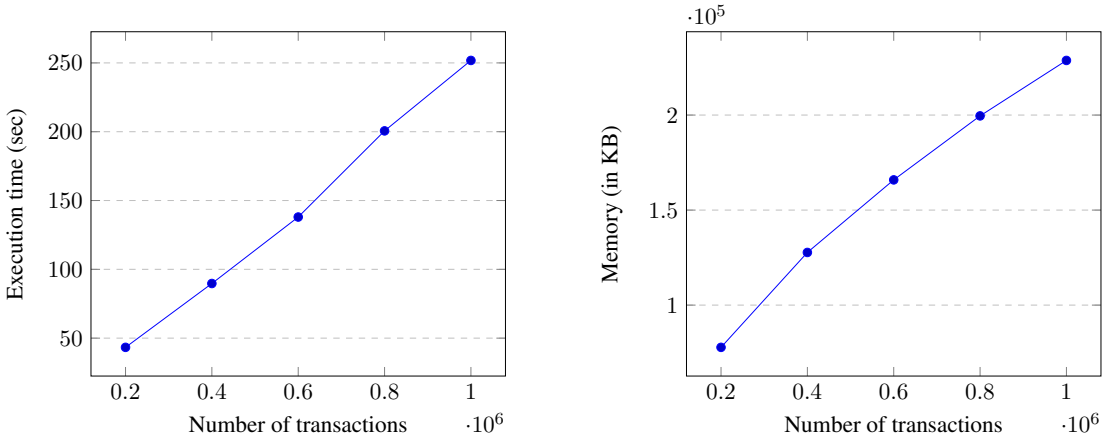
To test the scalability of the G3P-Growth algorithm, its performance and efficiency must be evaluated as the dataset size and complexity of the patterns increase. The primary goal of this test is to determine the algorithm's ability to handle larger and more complex datasets without significantly increasing the required resources while maintaining its speed.

The scalability test would involve generating or obtaining datasets of varying sizes and complexities and running the G3P-Growth algorithm on each dataset. The test would record execution time and memory usage, which would help to evaluate the algorithm's efficiency and suitability for handling larger and more complex datasets.

For this purpose, we chose the T10I4D1000K database. The database was divided into five portions, with 0.2 million transactions in each part. Then we investigated the performance of G3P-tree after accumulating each portion with previous parts. Fig. 5.13a, 5.13b respectively show the execution time of the algorithm and memory used for the construction of G3P-tree at $minPC = 0.1$, $minPS = 0.05\%$ and $maxIAT = 8000$. We deliberately set $minPC$ and $minPS$ to low values and $maxIAT$ to a

high value to assess the algorithm's maximum potential. It is evident from the almost linear increase in execution time and memory usage of the G3P-tree with the database size that the proposed G3P-algorithm is highly scalable, demonstrating its potential for managing large datasets.

Overall, the scalability test indicates that the G3P-Growth algorithm is suitable for handling larger and more complex datasets, making it a useful tool for mining partial periodic patterns in non-uniform temporal databases.



(a) Execution Time

(b) Memory

Figure 5.13: Scalability Results for G3P-Growth algorithm

5.4 Case Studies

We showcase the potential applications of partial periodic mining in non-uniform temporal databases through two case studies.

The first case study focuses on using the G3P-Growth algorithm to improve traffic safety during disastrous situations, while the second case study demonstrates the identification of highly polluted locations using the same algorithm. The results of these studies highlight the usefulness of the G3P-Growth algorithm in discovering partial periodic patterns and its potential for use in data-driven decision-making in various industries, including transportation and environmental management. The case studies are explained in detail in the following subsections.

5.4.1 Case study 1: Improving traffic safety during disastrous situations

This case study focuses on investigating the use of the G3P-Growth algorithm in improving traffic safety during disastrous situations. To demonstrate this application, the study examines a real-life event, Typhoon Nangka, that occurred in Kobe, Japan, in July 2015. The typhoon resulted in heavy

rainfall, flooding, and severe traffic congestion, highlighting the need for effective disaster management strategies to minimize the impact of such incidents.

Description of the case study

The dataset used for this case study consisted of traffic congestion data in Kobe, Japan, during the typhoon season of 2015. The congestion database was provided by JARTIC for Kobe, Japan and consisted of road segment identifiers (or sensor ids) with more than 300 meters of observed congestion. The data was collected at 5-minute intervals. To facilitate the identification of patterns in the data and improve its interpretability, we divided the congestion data into hourly intervals. The dataset also included hourly precipitation data obtained from the Japan Meteorological Agency. The combination of these two datasets allowed us to analyze the impact of the typhoon on traffic congestion in the city and develop strategies for managing it effectively.

We used the G3P-Growth algorithm to identify partial periodic patterns in each interval. The intuition behind this approach is that there might be certain patterns in the data that can be used to identify road segments with heavy traffic during the typhoon. By identifying these patterns, we can pinpoint the areas that require traffic monitoring and diversion.

The G3P-Growth algorithm suits better for the use case over the 3P-algorithm because there might be certain road segments that face congestion rarely and could be missed by the 3P-algorithm. Missing such patterns could result in unsafe circumstances for some of the areas. To identify the patterns, we used the following parameters $minPS = 15\%$, $minPC = 0.8\ (= 80\%)$, and $maxIAT = 5$ hours. These parameters were chosen based on our understanding of the traffic congestion problem and the available data. Using low $minPS$ and high $minPC$ values enables us to find patterns containing both frequent and rare items. Once the patterns were identified, we interpolated them onto hourly X-Rain data to pinpoint the road segments that require traffic monitoring and diversion.

Results

Fig. 5.13 illustrates the interpolation of patterns on precipitation data at six different hourly intervals.

The interpolated data suggests that areas with heavy traffic congestion, as identified by the G3P-Growth algorithm, could be considered high-risk during typhoon events, particularly if the typhoon is approaching near the spatial locations where the patterns were observed. This information can be used by local authorities, transportation agencies, and emergency services to identify areas that require traffic monitoring and diversion, as well as alerting pedestrians and suggesting police patrol routes.

Furthermore, as the precipitation shifts over different locations, the areas that could be declared unsafe in the beginning become safe once the precipitation goes away, while previously safe areas may

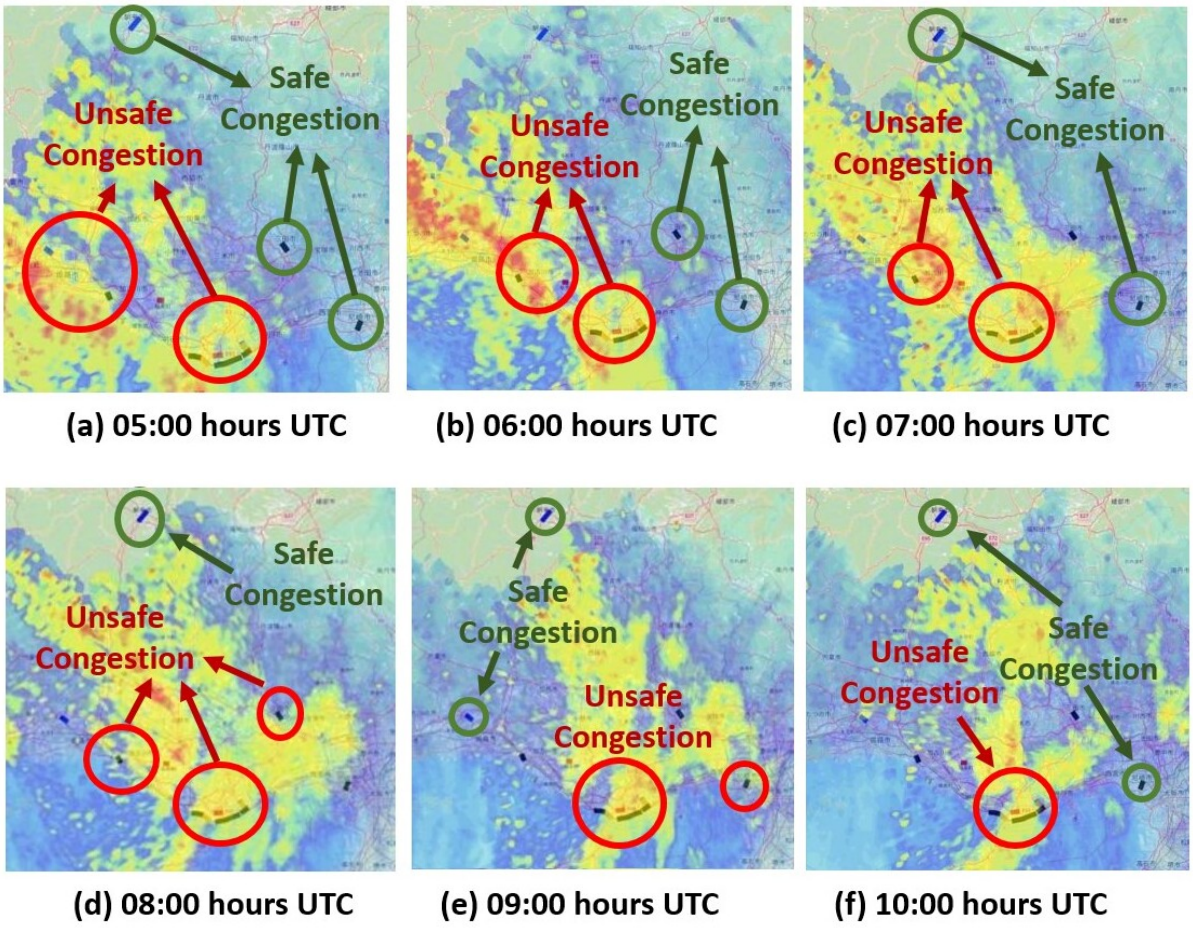


Figure 5.13: Patterns generated by segmenting the congestion data into hourly intervals. Precipitation data of Typhoon Nangka is overlaid at hourly intervals.

become unsafe. This highlights the dynamic nature of the situation and the need for real-time monitoring and decision-making

The case study showcases the potential application of the G3P-Growth algorithm in disaster management and transportation planning, emphasizing the usefulness of partial periodic mining in non-uniform temporal databases.

5.4.2 Case study 2: Identifying highly polluted locations

Air pollution is a major global concern that poses a serious threat to public health. In this case study, we employed the G3P-Growth algorithm to analyze the Pollution database of Japan, with the aim of identifying highly polluted locations with unsafe levels of PM2.5. The results of this study are of critical importance as they can provide policymakers with valuable insights into the spatial distribution of air pollution. By using the proposed algorithm, we aimed to gain a deeper understanding of air pollution

in Japan and to provide actionable information that can help to promote a healthier environment for everyone, particularly in areas such as harbours or bay regions that are particularly vulnerable to high levels of pollution.

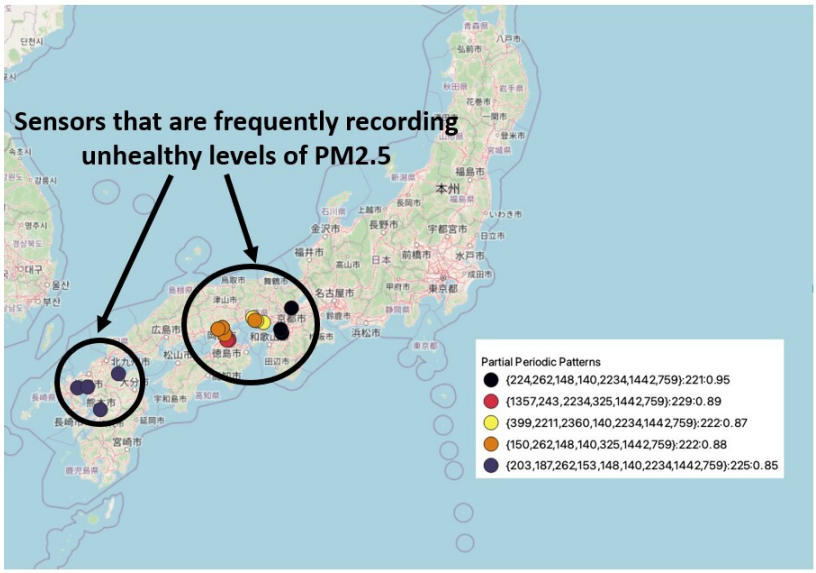
Description of the case study

The dataset used in this case study was the Pollution database of Japan, downloaded from the Atmospheric Environmental Regional Observation System (AEROS) website. The AEROS is a monitoring system for air pollutants managed by the Ministry of the Environment, Japan. The database consists of sensor identifiers whose PM2.5 values are equal to or greater than 16 $\mu g/m^3$. The data collection period for this study is from 1-April-2018 to 30-April-2018, and the data was collected at hourly intervals.

The primary objective of applying the G3P-Growth algorithm to the Pollution database was to identify the locations in Japan where there are frequent occurrences of unsafe levels of PM2.5. It is possible that some locations report high pollution levels together due to the same underlying reasons. By analyzing the patterns obtained from the G3P algorithm, we can identify such locations and explore the possible reasons for pollution triggers. The patterns obtained contain sensor IDs that appear frequently and rarely in the database, providing a comprehensive evaluation of the country's air pollution.

To apply the G3P-Growth algorithm, we used the following parameter values: $minPS = 40\%$, $minPC = 0.5$ $(= 50\%)$, and $maxIAT = 24$ hours. These values were chosen based on previous studies and the specific characteristics of the Pollution database. The G3P-Growth algorithm is preferred over the 3P-Growth algorithm for this particular use case because the Pollution database is non-uniform, and the 3P algorithm would not be able to discover all the patterns from the database. On the other hand, the G3P algorithm is capable of handling non-uniform databases and can identify patterns containing both rare and frequent items. Therefore, the G3P-Growth algorithm is more suitable for identifying partial periodic patterns that could be missed by other algorithms.

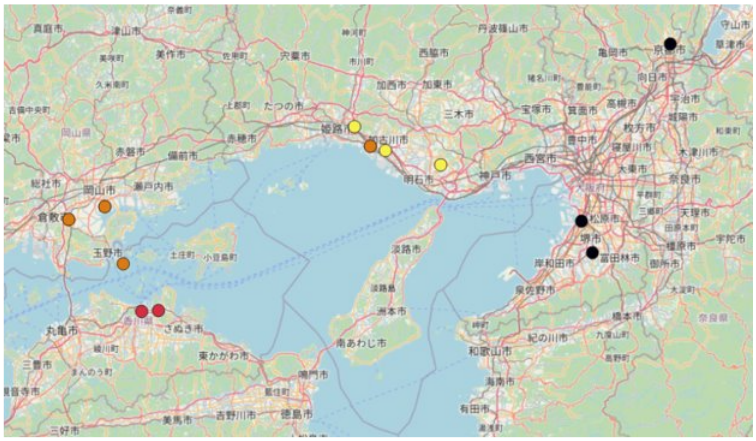
Results



A) Locations with unhealthy levels of PM2.5 frequently

Figure 5.14: Spatial location of sensors which measured highest levels of PM2.5 at regular intervals across Japan

B) Fukuoka area



C) Okayama, Kobe, Osaka area

Figure 5.14: A closer view of the identified pollution clusters in (A), displaying the specific locations with high levels of PM2.5 in the highlighted areas across Japan.

Table 5.5: Some of the interesting patterns generated from pollution database

Pattern	PS_1	PS_2	PC
181, 598, 242, 433	354	0.49	0.89
1391, 1378, 1377, 1266	354	0.49	0.89
1092, 1356, 1337, 1266	353	0.49	0.88
2340, 2289, 2041, 1988	352	0.49	0.90
1568, 1563, 1645, 1591	352	0.49	0.88

Table 5.5 presents several patterns extracted from the air pollution database. The *periodic-support* of each pattern in count and percentage is represented by 'PS_1' and 'PS_2' in the table, respectively. Fig 5.14 shows the spatial distribution of the sensors that were identified from the extracted patterns, revealing that individuals residing in Okayama, Kobe, Osaka, and Fukuoka were frequently exposed to hazardous levels of PM2.5.

The case study highlights the potential application of the G3P-Growth algorithm in urban planning. The algorithm's ability to analyze non-uniform data and identify frequent and rare patterns provides policymakers with valuable insights into the distribution of air pollution. By utilizing the spatial distribution of sensors, policymakers can develop effective pollution control regulations in areas such as harbors or bay regions to protect public health.

5.5 Summary

In this chapter, we performed multiple experiments to demonstrate that the proposed algorithm is highly scalable and efficient, making it a practicable algorithm for real-world datasets. We also showcase two case studies that illustrate the potential practical applications of our proposed solution in real-world scenarios. In the next chapter, we provide summary and conclusions.

Chapter 6

Conclusion and Future work

In this chapter, we provide the summary, conclusion and discuss future work.

6.1 Summary

Temporal databases are specialized databases designed to store and manage time-varying data, capturing the evolution of data over time. They provide a rich foundation for understanding how data changes over time and enable the discovery of patterns, trends, and dependencies. To discover patterns from temporal databases, pattern mining emerges as the key data mining technique. Pattern mining involves the extraction of valuable insights from large volumes of temporal data, aiding in identifying recurring behaviours, correlations, and associations. However, effectively mining patterns from these databases presents significant challenges. Existing models for mining 3Ps (partial periodic patterns) from temporal databases encounter the rare item problem, which involves either missing the patterns containing rare items or producing too many patterns, most of which may be uninteresting to the user. This problem hampers the extraction of meaningful patterns and associations in non-uniform databases.

To address this challenge, we proposed a novel null-invariant temporal measure, the *periodic confidence*, to discover partial periodic patterns in non-uniform temporal databases. The proposed measure considers the frequency of the items in the database and the occurrence of each item in the periods. The proposed model could effectively identify patterns containing frequent and rare items by using periodic confidence, thereby overcoming the rare item problem. Moreover, the proposed measure satisfies both *null-invariant* and *convertible anti-monotonic* properties. The null-invariant property allows our model to disclose genuine correlations without influencing the object co-absence in the database. The *convertible anti-monotonic property* facilitates our model to effectively reduce the enormous search space, allowing it to handle big data and make the proposed model practicable on large real-world databases. We also proposed an efficient depth-first search algorithm, called Generalized Partial Periodic Pattern-growth (G3P-Growth), to discover partial periodic patterns in non-uniform temporal databases. The

algorithm exploits the concept of "irregularity pruning" to eliminate the irregular occurrence information of items, thereby reducing the search space and improving the algorithm's efficiency.

In our experimental evaluation, we thoroughly assessed the efficiency and scalability of the proposed algorithm using a combination of synthetic and real-world datasets. The experiments included two synthetic databases, namely T10I4D100K and T20I6D100K, and four real-world datasets: Pollution, Congestion, BMS-WebView-1, and BMS-WebView-2.

We examined the effects of changing certain parameters to evaluate the performance of the proposed model. We focused on three parameters: minPC, minPS, and maxIAT. By adjusting these parameters individually, while keeping others constant, we could see how the algorithm responded. When we increased minPC and minPS while keeping maxIAT constant, we noticed a decrease in the number of identified patterns, as well as the algorithm's runtime and memory requirements. This was because the search space became smaller with higher minPC and minPS values. One varying maxIAT and minPIS, the increase in maxIAT led to more patterns, longer runtime, and higher memory requirements due to the exploration of patterns with longer inter-arrival times and larger search space, while the increase in minPS resulted in fewer patterns, shorter runtime, and lower memory requirements. On Varying minPC and maxIAT while keeping minPS constant showed that increasing maxIAT resulted in more patterns, longer runtime, and higher memory requirements while increasing minPC led to fewer patterns, shorter runtime, and lower memory requirements.

To evaluate the scalability of the experiments synthetic database (T10I4D1000K) containing 1 Million transactions is used. To study, we varied the number of transactions to consider different database sizes. We observe a linear increase in runtime and memory requirements of the algorithm with the increase in the size of the database. This observation highlights the efficiency and scalability of our proposed model, demonstrating its ability to handle larger datasets without compromising performance.

We presented two case studies to showcase the practical applications of our proposed model in real-world scenarios. The first case study aimed to enhance traffic safety during disastrous situations. We applied the G3P-Growth algorithm to data from Typhoon Nangka in Kobe, Japan. This enabled us to pinpoint specific road segments that required immediate traffic monitoring and diversion for improved safety measures. The second case study focused on identifying highly polluted locations using the G3P-Growth algorithm. We utilized the Pollution database of Japan to identify areas with consistently high levels of PM2.5, which is a harmful air pollutant. By analyzing the generated patterns, we obtained valuable insights into areas that are frequently exposed to significant levels of PM2.5. This information is crucial for implementing effective pollution control regulations and strategies.

6.2 Conclusions

The conclusions are as follows

- We have proposed a new pruning measure called periodic-confidence. Through experimental results, we conclude that the proposed model facilitates the extraction of 3Ps from large datasets.
- Through case studies, we have demonstrated the utility of the proposed model in deploying real applications. We conclude that there is a scope for the proposed approach in deploying similar kinds of applications.

6.3 Future Work

The proposed model could be extended to handle non-static databases, such as incremental databases and data streams - Incremental databases involve continuous updates and additions to the existing data. By extending the proposed model to handle incremental databases, it would be possible to mine 3Ps in real-time or near real-time scenarios efficiently. This would enable the identification of recurring patterns and associations as new data arrives, providing timely insights for decision-making processes. While data streams, on the other hand, involve a continuous flow of data with limited storage capacity. Extending the proposed model to handle data streams would allow for pattern mining in dynamic environments where data is processed in real-time and discarded after a certain period. This would require designing algorithms and data structures that can effectively handle the high velocity and potentially infinite nature of data streams.

Related Publications

1. Kiran, R.U., Chhabra, V., Chennupati, S., Reddy, P.K., Dao, M.S., Zettsu, K.: "Relative Period-Confidence: A Null-invariant Measure to Discover Partial Periodic Patterns in Non-Uniform Temporal Databases". In: *International Journal of Data Science and Analytics (JDSA)*. (In Progress, Accepted with revisions)

2. Kiran, R.U., Chhabra, V., Chennupati, S., Reddy, P.K., Dao, M.S., Zettsu, K.: "A novel null-invariant temporal measure to discover partial periodic patterns in non-uniform temporal databases". In: *Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part I, pp. 569–577*. Springer, Berlin, Heidelberg (2022)

Other Publications (Not related to the thesis)

1. Chhabra, V., Kiran, R.U., Xiao, J., Reddy, P.K., Avtar, R.: "A Spatiotemporal Image Fusion Method for Predicting High-Resolution Satellite Images". In: *Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings, pp. 470–481*. Springer, Berlin, Heidelberg (2022)

2. Chhabra, V., Kiran, R.U., Xiao, J., Reddy, P.K., Avtar, R.: "A Novel Parallel Spatiotemporal Image Fusion Method for Predicting High-Resolution Satellite Images". In: *Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 36th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2023 (Accepted)*

Bibliography

[1] KristofMoris. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, June 2011.

[2] Manpreet Kaur and Shivani Kang. Market Basket Analysis: Identify the Changing Trends of Market Data Using Association Rule Mining. *Procedia Comput. Sci.*, 85:78–85, January 2016.

[3] Stefan Naulaerts, Pieter Meysman, Wout Bittremieux, Trung Nghia Vu, Wim Vanden Berghe, Bart Goethals, and Kris Laukens. A primer to frequent itemset mining for bioinformatics. *Briefings Bioinf.*, 16(2):216–231, March 2015.

[4] Puteri N. E. Nohuddin, Rob Christley, Frans Coenen, Yogesh Patel, Christian Setzkorn, and Shane Williams. Frequent Pattern Trend Analysis in Social Networks. In *Advanced Data Mining and Applications*, pages 358–369. Springer, 2010.

[5] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

[6] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Database Theory — ICDT'99*, pages 398–416. Springer, January 1999.

[7] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 2001–02. IEEE.

[8] Carl H. Mooney and John F. Roddick. Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.*, 45(2):1–39, March 2013.

[9] Carson Kai-Sang Leung. Uncertain Frequent Pattern Mining. In *Frequent Pattern Mining*, pages 339–367. Springer, August 2014.

[10] Jerry Chun-Wei Lin, Ting Li, Philippe Fournier-Viger, and Tzung-Pei Hong. A fast Algorithm for mining fuzzy frequent itemsets. *J. Intell. Fuzzy Syst.*, 29(6):2373–2379, January 2015.

[11] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Tin Truong-Chi, and Roger Nkambou. A Survey of High Utility Itemset Mining. In *High-Utility Pattern Mining: Theory, Algorithms and Applications*, pages 1–45. Springer, January 2019.

[12] Carson Kai-Sang Leung and Syed Khairuzzaman Tanbeer. PUF-Tree: A Compact Tree Structure for Frequent Pattern Mining of Uncertain Data. In *Advances in Knowledge Discovery and Data Mining*, pages 13–25. Springer, 2013.

[13] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proceedings 14th International Conference on Data Engineering*, pages 412–421. IEEE, February 1998.

[14] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Discovering Periodic-Frequent Patterns in Transactional Databases. In *Advances in Knowledge Discovery and Data Mining*, pages 242–253. Springer, 2009.

[15] R. Uday Kiran and P. Krishna Reddy. Towards Efficient Mining of Periodic-Frequent Patterns in Transactional Databases. In *Database and Expert Systems Applications*, pages 194–208. Springer, 2010.

[16] Akshat Surana, R. Uday Kiran, and P. Krishna Reddy. An efficient approach to mine periodic-frequent patterns in transactional databases. In *PAKDD'11: Proceedings of the 15th international conference on New Frontiers in Applied Data Mining*, pages 254–266. Springer-Verlag, May 2011.

[17] R. Uday Kiran and P. Krishna Reddy. An Alternative Interestingness Measure for Mining Periodic-Frequent Patterns. In *Database Systems for Advanced Applications*, pages 183–192. Springer, 2011.

[18] Md. Mamunur Rashid, Md. Rezaul Karim, Byeong-Soo Jeong, and Ho-Jin Choi. Efficient Mining Regularly Frequent Patterns in Transactional Databases. In *Database Systems for Advanced Applications*, pages 258–271. Springer, 2012.

[19] Vincent Mwintieru Nofong. Discovering Productive Periodic Frequent Patterns in Transactional Databases. *Ann. Data. Sci.*, 3(3):235–249, September 2016.

[20] R. Uday Kiran, Haichuan Shang, Masashi Toyoda, and Masaru Kitsuregawa. Discovering Partial Periodic Itemsets in Temporal Databases. In *SSDBM '17: Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–6. Association for Computing Machinery, June 2017.

[21] Philippe Fournier-Viger, Peng Yang, Jerry Chun-Wei Lin, and Rage Uday Kiran. Discovering Stable Periodic-Frequent Patterns in Transactional Data. In *Advances and Trends in Artificial Intelligence. From Theory to Practice*, pages 230–244. Springer, June 2019.

[22] R. Uday Kiran, C. Saideep, Penugonda Ravikumar, Koji Zettsu, Masashi Toyoda, Masaru Kitsuregawa, and P. Krishna Reddy. Discovering Fuzzy Periodic-Frequent Patterns in Quantitative Temporal Databases. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, July 2020.

[23] Penugonda Ravikumar, R. Uday Kiran, Palla Likhitha, T. Chandrasekhar, Yutaka Watanobe, and Koji Zettsu. Discovering Geo-referenced Periodic-Frequent Patterns in Geo-referenced Time Series Databases. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, October 2022.

[24] Pamalla Veena, Penugonda Ravikumar, Kundai Kwangwari, R. Uday Kiran, Kazuo Goda, Yutaka Watanobe, and Koji Zettsu. Discovering Fuzzy Geo-referenced Periodic-Frequent Patterns in Geo-referenced Time Series Databases. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, July 2022.

[25] J. M. Luna, Philippe Fournier-Viger, and S. Ventura. Frequent itemset mining: A 25 years review. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery*, 2019.

[26] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, January 2004.

[27] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(3):372–390, May 2000.

[28] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 163–170. IEEE, November 2001.

[29] Jiawei Han, Jianyong Wang, Ying Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 211–218. IEEE, December 2002.

[30] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Discovering Periodic-Frequent Patterns in Transactional Databases. In *Advances in Knowledge Discovery and Data Mining*, pages 242–253. Springer, 2009.

[31] P. Gowtham Srinivas, P. Krishna Reddy, A. V. Trinath, S. Bhargav, and R. Uday Kiran. Mining coverage patterns from transactional databases. volume 45, pages 423–439. Springer, December 2015.

[32] Hong Yao, Howard J. Hamilton, and Cory J. Butz. *A Foundational Approach to Mining Itemset Utilities from Databases*, pages 482–486.

[33] Komate Amphawan, Philippe Lenca, and Athasit Surarerks. Mining Top-K Periodic-Frequent Pattern from Transactional Databases without Support Threshold. In *Advances in Information Technology*, pages 18–29. Springer, 2009.

[34] R. Uday Kiran, Masaru Kitsuregawa, and P. Krishna Reddy. Efficient discovery of periodic-frequent patterns in very large databases. *Journal of Systems and Software*, 112:110–121, February 2016.

[35] J. N. Venkatesh, R. Uday Kiran, P. Krishna Reddy, and Masaru Kitsuregawa. Discovering Periodic-Correlated Patterns in Temporal Databases. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVIII: Special Issue on Database- and Expert-Systems Applications*, pages 146–172. Springer, November 2018.

[36] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: generalizing association rules to correlations. In *ACM SIGMOD Record*, volume 26, pages 265–276. Association for Computing Machinery, June 1997.

[37] E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Trans. Knowl. Data Eng.*, 15(1):57–69, January 2003.

[38] Sangkyum Kim, Marina Barsky, and Jiawei Han. Efficient Mining of Top Correlated Patterns Based on Null-Invariant Measures. In *Machine Learning and Knowledge Discovery in Databases*, pages 177–192. Springer, 2011.

[39] Hyunyoon Yun, Danshim Ha, Buhyun Hwang, and Keun Ho Ryu. Mining association rules on significant rare data using relative support. *Journal of Systems and Software*, 67(3):181–191, September 2003.

[40] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[41] D. W. Cheung, Jiawei Han, V. T. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: an incremental updating technique. In *Proceedings of the 12th International Conference on Data Engineering*, pages 106–114. IEEE, February 1996.

[42] R. Uday Kiran, P. Likhitha, Minh-Son Dao, Koji Zettsu, and Ji Zhang. Discovering Periodic-Frequent Patterns in Uncertain Temporal Databases. In *Neural Information Processing*, pages 710–718. Springer, December 2021.

[43] Keith C. C. Chan and Wai-Ho Au. Mining fuzzy association rules. In *CIKM '97: Proceedings of the 6th International Conference on Information and Knowledge Management*, pages 209–215. Association for Computing Machinery, January 1997.

[44] Joong Hyuk Chang and Won Suk Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 487–492. Association for Computing Machinery, August 2003.

[45] Zhongmei Zhou, Zhaohui Wu, Chunshan Wang, and Yi Feng. Mining Both Associated and Correlated Patterns. In *Computational Science – ICCS 2006*, pages 468–475. Springer, 2006.

[46] Sangkyum Kim, Marina Barsky, and Jiawei Han. Efficient mining of top correlated patterns based on null-invariant measures. In *ECML PKDD'11: Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II*, pages 177–192. Springer-Verlag, September 2011.

[47] Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. Using association rules for product assortment decisions: a case study. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 254–260. Association for Computing Machinery, August 1999.

[48] R. Uday Kiran, Sourabh Shrivastava, Philippe Fournier-Viger, Koji Zettsu, Masashi Toyoda, and Masaru Kitsuregawa. Discovering Frequent Spatial Patterns in Very Large Spatiotemporal Databases. In *SIGSPATIAL '20: Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 445–448. Association for Computing Machinery, November 2020.

[49] Hung Tran-The and Koji Zettsu. Discovering co-occurrence patterns of heterogeneous events from unevenly-distributed spatiotemporal data. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1006–1011. IEEE, December 2017.

[50] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoret. Comput. Sci.*, 312(1):3–15, January 2004.

[51] R. Uday Kiran and Masaru Kitsuregawa. Novel Techniques to Reduce Search Space in Periodic-Frequent Pattern Mining. In *Database Systems for Advanced Applications*, pages 377–391. Springer, 2014.

[52] J. N. Venkatesh, R. Uday Kiran, P. Krishna Reddy, and Masaru Kitsuregawa. Discovering Periodic-Frequent Patterns in Transactional Databases Using All-Confidence and Periodic-All-Confidence. In *DEXA 2016: Proceedings, Part I, 27th International Conference on Database and Expert Systems Applications - Volume 9827*, pages 55–70. Springer-Verlag, September 2016.

[53] T. Yashwanth Reddy, R. Uday Kiran, Masashi Toyoda, P. Krishna Reddy, and Masaru Kitsuregawa. Discovering Partial Periodic High Utility Itemsets in Temporal Databases. In *Database and Expert Systems Applications*, pages 351–361. Springer, August 2019.

[54] R. Uday Kiran, C. Saideep, Koji Zettsu, Masashi Toyoda, Masaru Kitsuregawa, and P. Krishna Reddy. Discovering Partial Periodic Spatial Patterns in Spatiotemporal Databases. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 233–238. IEEE, December 2019.

[55] C. Saideep, R. Uday Kiran, Koji Zettsu, Cheng-Wei Wu, P. Krishna Reddy, Masashi Toyoda, and Masaru Kitsuregawa. Parallel Mining of Partial Periodic Itemsets in Big Data. In *Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices*, pages 807–819. Springer, September 2020.

[56] P. Likitha, P. Veena, R. Uday Kiran, Yukata Watanobe, and Koji Zettsu. Discovering Maximal Partial Periodic Patterns in Very Large Temporal Databases. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1460–1469. IEEE, December 2021.

[57] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *KDD '99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 337–341. Association for Computing Machinery, August 1999.

[58] Ya-Han Hu and Yen-Liang Chen. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. *Decision Support Systems*, 42(1):1–24, October 2006.

[59] R. Uday Kiran and P. Krishna Reddy. Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. pages 11–20, March 2011.

[60] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *KDD '02: Proceedings of the 8TH ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–41. Association for Computing Machinery, July 2002.

[61] Akshat Surana, R. Uday Kiran, and P. Krishna Reddy. Selecting a right interestingness measure for rare association rules. In P. Sreenivasa Kumar, Srinivasan Parthasarathy, and Shantanu Godbole, editors, *Proceedings of the 16th International Conference on Management of Data, 2010, Nagpur, India*, page 115. Allied Publishers, 2010.

[62] R. Uday Kiran, Haichuan Shang, Masashi Toyoda, and Masaru Kitsuregawa. Discovering Partial Periodic Itemsets in Temporal Databases. In *SSDBM '17: Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–6. Association for Computing Machinery, June 2017.

[63] Jian Pei and Jiawei Han. Can we push more constraints into frequent pattern mining? In *KDD '00: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, pages 350–354. Association for Computing Machinery, August 2000.

[64] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann Publishers Inc., September 1994.

[65] Pami - gthreepgrowth. `https://github.com/udayRage/PAMI/blob/main/PAMI/partialPeriodicPattern/basic/GThreePGrowth.py`, April 2023. [Online; accessed 18. Apr. 2023].

[66] Air pollution data of aeros. `https://soramame.env.go.jp/download`, February 2023. [Online; accessed 18. Apr. 2023].

[67] SIGKDD : KDD Cup 2000 : Online retailer website clickstream analysis. `https://kdd.org/kdd-cup/view/kdd-cup-2000`, April 2023. [Online; accessed 18. Apr. 2023].