Mining Partial Periodic-Frequent Patterns in Temporal Databases

What is partial periodic-frequent pattern mining?

Partial Periodic-Frequent pattern mining aims to discover all interesting patterns in a temporal database that have **support** no less than the user-specified **minimum support** (**minSup**) constraint, **periodicity** no greater than user-specified **maximum periodicity** (**maxPer**) constraint and **periodic ratio** no less than user-specified **minimum periodic ratio (minPR)**. The **minSup** controls the minimum number of transactions that a pattern must appear in a database, **maxPer** controls the maximum time interval within which a pattern must reappear in the database and the **minPR** controls the minimum periodic ratio which is the proportion of cyclic repititions of a pattern in database.

Research paper: R. Uday Kiran, J.N. Venkatesh, Masashi Toyoda, Masaru Kitsuregawa, P. Krishna Reddy, Discovering partial periodic-frequent patterns in a transactional database, Journal of Systems and Software, Volume 125, 2017, Pages 170-182, ISSN 0164-1212,https://doi.org/10.1016/j.jss.2016.11.035.

What is a temporal database?

A temporal database is a collection of transactions at a particular timestamp, where each transaction contains a timestamp and a set of items.
A hypothetical temporal database containing the items ***a, b, c, d, e, f, and g*** as shown below

TS	Transactions
1	a b c g
2	b c d e
3	a b c d
4	a c d f
5	a b c d g
6	c d e f
7	a b c d
8	a e f
9	a b c d
10	b c d e

Note: Duplicate items must not exist in a transaction.

Acceptable format of temporal databases in PAMI

Each row in a temporal database must contain timestamp and items.

1 a b c g
2 b c d e
3 a b c d
4 a c d f
5 a b c d g
6 c d e f
7 a b c d
8 a e f
9 a b c d
10 b c d e

Understanding the statisctics of database

To understand about the database. The below code will give the detail about the transactional database.

- Total number of transactions (Database size)
- Total number of unique items in database
- Minimum lenth of transaction that existed in database
- Average length of all transactions that exists in database
- Maximum length of transaction that existed in database
- Minimum periodicity exists in database
- Average periodicity exists in database
- Maximum periodicity exists in database
- Standard deviation of transaction length
- Variance in transaction length
- Sparsity of database

The below sample code prints the statistical details of a database.

In [20]:
```
import PAMI.extras.dbStats.temporalDatabaseStats as stats

obj = stats.temporalDatabaseStats('sampleInputFile.txt', ' ')
obj.run()
obj.printStats()
```

```
Database size : 10
Number of items : 7
Minimum Transaction Size : 3
Average Transaction Size : 4.0
Maximum Transaction Size : 5
Minimum period : 1
Average period : 1.0
Maximum period : 1
Standard Deviation Transaction Size : 0.4472135954999579
Variance : 0.2222222222222222
Sparsity : 0.42857142857142855
```

What is the input to partial periodic-frequent pattern mining algorithms

Algorithms to mine the partial periodic-frequent patterns requires temporal database, minSup and maxPer (specified by user).

- Temporal database is accepted following formats:

  > - String : E.g., 'temporalDatabase.txt'
  > - URL : E.g., https://u-aizu.ac.jp/~udayrage/datasets/transactionalDatabases/transactional_T10
  > - DataFrame. Please note that dataframe must contain the header titled 'TS' and 'Transactions'

- minSup should be mentioned in

  > - **count (beween 0 to length of database)**
  > - [0, 1]

- maxPer should be mentioned in

  > - **count (beween 0 to length of database)**
  > - [0, 1]

- minPR should be mentioned in

  > - [0, 1]

- seperator
  default seperator is '\t' (tab space)

How to run the partial periodic-frequent pattern algorithm in terminal

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into partial periodic frequent pattern folder.
- Enter into partialPeriodicFrequentPattern folder
- You will another folder like **basic**
- Enter and execute the following command on terminal.

syntax: python3 algorithmName.py `<path to the input file>` `<path to the output file>` `<minSup>` `<maxPer>` `<minPR>` `<seperator>`

Example: python3 `GPFGrowth.py` `inputFile.txt` `outputFile.txt` `3` `4` `0.5` `' '`

How to execute a partial periodic-frequent pattern mining algorithm in a Jupyter Notebook?

- Install the PAMI package from the PYPI repository by executing the following command: **pip3 install PAMI**
- Run the below sample code by making necessary changes

In [ ]:
```
import PAMI.partialPeriodicFrequentPattern.basic.PPF_DFS as alg

iFile = 'sampleInputFile.txt'  #specify the input transactional database
minSup = 5                     #specify the minSup value
maxPer = 3                     #specify the maxPer value
minPR = 0.4                   #specify the minSup value
seperator = ' '                #specify the seperator. Default seperator
oFile = 'periodicPatterns.txt'   #specify the output file name

obj = alg.PPF_DFS(iFile, minSup, maxPer, minPR, seperator) #initialize th
obj.startMine()                          #start the mining process
obj.savePatterns(oFile)                  #store the patterns in file
df = obj.getPatternsAsDataFrame()        #Get the patterns discovered into a
obj.printStats()                         #Print the statistics of mining pro
```

The periodicPatterns.txt file contains the following patterns (*format:* pattern:support:periodicity):!cat periodicPatterns.txt

In [25]:
```
!cat periodicPatterns.txt
```

```
('d', 'c', 'b'):[6, 1.0]
('d', 'c', 'a'):[5, 1.0]
('c', 'd'):[8, 1.0]
('b', 'c', 'a'):[5, 1.0]
('c', 'b'):[7, 1.0]
('c', 'a'):[6, 1.0]
('c',):[9, 1.0]
('d', 'b'):[6, 1.0]
('d', 'a'):[5, 1.0]
('d',):[8, 1.0]
('b', 'a'):[5, 1.0]
('b',):[7, 1.0]
('a',):[7, 1.0]
```

The dataframe containing the patterns is shown below:

In [26]:
```
df
```

Out[26]:

	Patterns	Support	Periodicity
0	(d, c, b)	6	1.0
1	(d, c, a)	5	1.0
2	(c, d)	8	1.0
3	(b, c, a)	5	1.0
4	(c, b)	7	1.0
5	(c, a)	6	1.0
6	(c,)	9	1.0
7	(d, b)	6	1.0
8	(d, a)	5	1.0
9	(d,)	8	1.0
10	(b, a)	5	1.0
11	(b,)	7	1.0
12	(a,)	7	1.0