Mining Periodic Frequent Patterns in Uncertain Temporal Databases

What is periodic-frequent pattern mining?

Periodic-Frequent pattern mining aims to discover all interesting patterns in a transactional database that have **support** no less than the user-specified **minimum support** (**minSup**) constraint and **periodicity** no greater than user-specified **maximum period** (**maxPer**). The **minSup** controls the minimum number of transactions that a pattern must appear in a database and **maxPer** controls the maximum interval time a pattern must reappear.

What is a uncertain temporal database?

A temporal database is a collection of transactions at different timestamps, where each transaction contains a timestamp and a set of items with their respective uncertain value.
A hypothetical transactional database containing the items ***a, b, c, d, e, f, and g*** as shown below.

TS	Transactions
1	a(0.4) b(0.5) c(0.2) g(0.1)
2	b(0.2) c(0.3) d(0.4) e(0.2)
3	a(0.3) b(0.1) c(0.3) d(0.4)
4	a(0.2) c(0.6) d(0.2) f(0.1)
5	a(0.3) b(0.2) c(0.4) d(0.5) g(0.3)
6	c(0.2) d(0.7) e(0.34) f(0.2)
7	a(0.6) b(0.4) c(0.3) d(0.2)
8	a(0.2) e(0.2) f(0.2)
9	a(0.1) b(0.3) c(0.2) d(0.4)
10	b(0.3) c(0.2) d(0.1) e(0.6)

Note: Duplicate items must not exist in a transaction.

What is acceptable format of a uncertain temporal databases in PAMI?

Each row in a transactional database must contain timestamp and items with their respective uncertain values. A sample transactional database, say sampleInputFile.txt, is provided below.

1 a(0.4) b(0.5) c(0.2) g(0.1)
2 b(0.2) c(0.3) d(0.4) e(0.2)
3 a(0.3) b(0.1) c(0.3) d(0.4)
4 a(0.2) c(0.6) d(0.2) f(0.1)
5 a(0.3) b(0.2) c(0.4) d(0.5) g(0.3)
6 c(0.2) d(0.7) e(0.34) f(0.2)
7 a(0.6) b(0.4) c(0.3) d(0.2)
8 a(0.2) e(0.2) f(0.2)
9 a(0.1) b(0.3) c(0.2) d(0.4)
10 b(0.3) c(0.2) d(0.1) e(0.6)

What are the input parameters?

The input parameters to a periodic frequent pattern mining algorithm are:

- **Transactional database**
  Acceptable formats:

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

- **minSup**
  specified in

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

- **maxPer**
  specified in

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

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

How to store the output of a uncertain periodic frequent pattern mining algorithm?

The patterns discovered by a uncertain periodic frequent pattern mining algorithm can be saved in file or dataframe.

How to run the uncertain periodic-frequent pattern mining algorithms in a terminal?

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into uncertain periodic frequent pattern folder.
- Enter into uncertainPeriodicFrequentPattern folder
- Enter into a specific folder and execute the following command on terminal.

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

Example: python3 `UPFPGrowth.py` `inputFile.txt` `outputFile.txt` `0.05` `4` `' '`

How to execute a uncertain 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.uncertainPeriodicFrequentPattern.basic.UPFPGrowth as alg

iFile = 'samplefuncertain.txt'  #specify the input transactional database
minSup = 0.8  #specify the minSupvalue <br>
maxPer = 4    #specify the maxPer value <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'periodicFrequentPatterns.txt'   #specify the output file name<br

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

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

In [ ]:
```
!cat periodicFrequentPatterns.txt
```

The output in file format:

e :1.3399999999999999:4
b :2.0:2
a :2.0999999999999996:2
c :2.7:2
d :2.9000000000000004:2
c d :0.8600000000000001:2

The dataframe containing the patterns is shown below:

In [ ]:
```
df
```

	Patterns	Support	Periodicity
0	e	1.34	4
1	b	2.00	2
2	a	2.09	2
3	c	2.70	2
4	d	2.90	2
5	c d	0.86	2