Mining Stable Periodic Patterns in Temporal Databases

What is stable periodic pattern mining?

Stable periodic pattern mining aims to dicover all interesting patterns in a temporal database using three contraints **miimum support**, **maximum period** and **maximum lability**, that have **support** no less than the user-specified **minimum support** (**minSup**) constraint and **lability** no greater than **maximum lability** (**maxLa**).

Reference: Fournier-Viger, P., Yang, P., Lin, J. C.-W., Kiran, U. (2019). Discovering Stable Periodic-Frequent Patterns in Transactional Data. Proc. 32nd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA AIE 2019), Springer LNAI, pp. 230-244

What is a temporal database?

A temporal database is an ordered collection of transactions. A transaction represents a pair constituting of timestamp and a set of items.
A hypothetical temporal database containing the items ***a, b, c, d, e, f, and g*** is 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 within a transaction.

What is the acceptable format of a temporal database in PAMI?

Each row in a temporal database must contain timestamp and items. The stable periodic frequent pattern mining algorithms considers the timestamp to calculate the periodicity. A sample temporal database, say sampleInputFile.txt, is provided below.

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 statistics of a temporal database

The performance of a pattern mining algorithm primarily depends on the satistical nature of a database. Thus it is important to know the following details of a database:

- Total number of transactions (Database size)
- Total number of unique items in database
- Minimum lenth of transaction that exists in database
- Average length of all transactions that exists in database
- Maximum length of transaction that exists in database
- Minimum periodicity that exists in database
- Average periodicity hat exists in database
- Maximum periodicity that 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 [ ]:

```
import PAMI.extras.dbStats.temporalDatabaseStats as stats
obj = stats.temporalDatabaseStats('sampleInputFile.txt', ' ')
obj.run()
obj.printStats()
```

The input parameters to a frequent pattern mining algorithm are:

- **Temporal database**

  Acceptable formats:

  > - String : E.g., 'transactionalDatabase.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]

- **maxLa**

  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 stable periodic frequent pattern mining algorithm?

The patterns discovered by a stable periodic frequent pattern mining algorithm can be saved into a file or a data frame.

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

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into stable periodic frequent pattern folder.
- Enter into stablePeriodicFrequentPattern folder
- You will find different types of folders like **basic, topk**
- Enter into **basic** folder execute the following command on terminal.

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

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

How to execute a stable 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 [ ]:

```python
import PAMI.stablePeriodicFrequentPattern.basic.SPPGrowth as alg

iFile = 'sampleInputFile.txt'  #specify the input temporal database <br>
minSup = 5  #specify the minSupvalue <br>
maxPer = 3   #specify the maxPervalue <br>
maxLa = 3    #specify the minLavalue <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'stablePatterns.txt'   #specify the output file name<br>

obj = alg.SPPGrowth(iFile, minSup, maxPer, maxLa, seperator) #initialize
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 stablePatterns.txt file contains the following patterns (*format:* pattern:support:lability):!cat stablePatterns.txt

In [3]:

```
!cat stablePatterns.txt
```

```
a :7:0
a b :5:0
a b c :5:0
a d :5:0
a d c :5:0
a c :6:0
b :7:0
b d :6:0
b d c :6:0
b c :7:0
d :8:0
d c :8:0
c :9:0
```

The dataframe containing the patterns is shown below:

In [4]:

```
df
```

Out[4]:

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