Mining periodic correlated patterns in a temporal

What is periodic correlated pattern mining?

Periodic correlated pattern mining aims to discover all the interesting patterns using **support, all confidence, periodicity and periodic all confidence**, that have **support** no less than the user-specified **minimum support (minSup)**, **all confidence** no less than **minimum all confidence (minAllConf)**, **periodicity** no greater than **maximum periodicity (maxPer)** and **periodic all confidence** no greater than **maximum period all confidence maxPerAllConf**

Reference: Venkatesh, J.N., Uday Kiran, R., Krishna Reddy, P., Kitsuregawa, M. (2018). Discovering Periodic-Correlated Patterns in Temporal Databases. In: Hameurlain, A., Wagner, R., Hartmann, S., Ma, H. (eds) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVIII. Lecture Notes in Computer Science(), vol 11250. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-58384-5_6

What is a temporal database?

A temporal database is an unordered collection of transactions. A temporal represents a pair constituting of temporal-timestamp and a set of items.
A hypothetical temporal database containing the items ***a, b, c, d, e, f, and g*** and its timestamp 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. A sample transactional 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()
```

What are the input parameters?

The input parameters to a periodic 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]

- **minAllConf**
  specified in

  > - [0, 1]

- **maxPer**
  specified in

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

- **maxPerAllConf**
  specified in

  > - [0, 1]

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

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

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

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

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into periodic correlated pattern folder.
- Enter into periodicCorrelatedPattern folder
- Enter into specific folder execute the following command on terminal.

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

Example: python3 `EPCPGrowth` `inputFile.txt` `outputFile.txt` `4` `0.5` `3` `0.4` `' '`

How to execute a periodic correlated 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.periodicCorrelatedPattern.EPCPGrowth as alg

iFile = 'sampleInputFile.txt'  #specify the input temporal database <br>
minSup = 4  #specify the minSupvalue <br>
minAllConf = 0.6
maxPer = 4   #specify the maxPervalue <br>
maxPerAllConf = 1.5    #specify the maxPerAllConfValue <br>
seperator = ' '  #specify the seperator. Default seperator is tab space. <
oFile = 'periodicCorrelatedPatterns.txt'    #specify the output file name<

obj = alg.EPCPGrowth(iFile, minSup, minAllConf, maxPer, maxPerAllConf, se
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 correlatedPeriodicPatterns.txt file contains the following patterns (*format:* pattern:support:lability):!cat periodicCorrelatedPatterns.txt

In [7]:

```
!cat periodicCorrelatedPatterns.txt
```

```
e :4:4:1:1
a :7:2:1:1
a b :5:2:0.7142857142857143:1.0
a d :5:3:0.625:1.5
a c :6:2:0.6666666666666666:1.0
b :7:2:1:1
b d :6:2:0.75:1.0
b d c :6:2:0.6666666666666666:1.0
b c :7:2:0.7777777777777778:1.0
d :8:2:1:1
d c :8:2:0.8888888888888888:1.0
c :9:2:1:1
```

The dataframe containing the patterns is shown below:

In [8]:
```
df
```

Out[8]:

	Patterns	Support	allConf	Periodicity	maxPerAllConf
0	e	4	4	1.000000	1.0
1	a	7	2	1.000000	1.0
2	a b	5	2	0.714286	1.0
3	a d	5	3	0.625000	1.5
4	a c	6	2	0.666667	1.0
5	b	7	2	1.000000	1.0
6	b d	6	2	0.750000	1.0
7	b d c	6	2	0.666667	1.0
8	b c	7	2	0.777778	1.0
9	d	8	2	1.000000	1.0
10	d c	8	2	0.888889	1.0
11	c	9	2	1.000000	1.0