Mining Geo Referenced Periodic-Frequent Patterns in Temporal Databases

What is geo referenced periodic-frequent pattern mining?

Geo Referenced 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 **distance** between two items is no less than **maximum distance** (**maxDist**). The **minSup** controls the minimum number of transactions that a pattern must appear in a database and the **maxPer** controls the maximum time interval within which a pattern must reappear in the database.

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

What is the spatial database?

Spatial database contain the spatial (neighbourhood) information of items. It contains the items and its nearset neighbours satisfying the **maxDist** constraint.

Items	neighbours
a	b, c, d
b	a, e, g
c	a, d
d	a, c
e	b, f
f	e, g
g	b, f

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 [ ]:

```
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 spatial pattern mining algorithm are:

- **Temporal 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'

- **Neighbour database**
  Acceptable formats:

  > - String : E.g., 'NeighbourDatabase.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 geo referenced periodic frequent pattern mining algorithm?

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

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

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into geo referenced periodic frequent pattern folder.
- Enter into geoReferencedPeriodicFrequentPattern folder
- Enter into a specific folder of your choice and execute the following command on terminal.

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

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

How to implement the GPFPMiner algorithm by importing PAMI package

- 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.geoReferencedPeriodicFrequentPattern.GPFPMiner as alg

iFile = 'sampleInputFile.txt'  #specify the input temporal database <br>
nFile = 'sampleNeighbourFile.txt'  #specify the input neighbour database
minSup = 5  #specify the minSupvalue <br>
maxPer = 3  #specify the maxPer value <br>
seperator = ' '  #specify the seperator. Default seperator is tab space. <
oFile = 'Patterns.txt'   #specify the output file name<br>

obj = alg.GPFPMiner(iFile, nFile, minSup, maxPer, 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 Patterns.txt file contains the following patterns (*format:* pattern:support:periodicity):!cat Patterns.txt

In [3]:

```
!cat Patterns.txt
```

```
d c a : 5: 3
c d : 8: 2
c a : 6: 2
c : 9: 2
d a : 5: 3
d : 8: 2
b a : 5: 2
b : 7: 2
a : 7: 2
```

The dataframe containing the patterns is shown below:

In [4]:

```
df
```

Out[4]:

	Patterns	Support	Period
0	(d, c, a)	5	3
1	(c, d)	8	2
2	(c, a)	6	2
3	(c,)	9	2
4	(d, a)	5	3
5	(d,)	8	2
6	(b, a)	5	2
7	(b,)	7	2
8	(a,)	7	2