Mining High-Utility Spatial Patterns in Utility Databases

What is High-Utility Spatial pattern mining?

High utility pattern mining aims to discover all the patterns with utility of pattern is no less than user-specified ***minimum utility*** threshold ***minutil*** and no distance between any two of its items should be no greater than user-specified ***maximum distance***.

Reference: R. Uday Kiran, Koji Zettsu, Masashi Toyoda, Philippe Fournier-Viger, P. Krishna Reddy, and Masaru Kitsuregawa. 2019. Discovering Spatial High Utility Itemsets in Spatiotemporal Databases. In Proceedings of the 31st International Conference on Scientific and Statistical Database Management (SSDBM '19). Association for Computing Machinery, New York, NY, USA, 49–60. https://doi.org/10.1145/3335783.3335789

What is a utility database?

A utility database is a collection of transaction, where each transaction contains a set of items and a positive integer called ***internal utility*** respectively. And each unique item in database is also associated with another positive number called ***external utility*** for each transaction.
A hypothetical utility database with items ***a, b, c, d, e, f and g*** and its ***internal utility*** is shown below at right side and items with its ***external utilities*** for each transaction is presented at left side.

Transactions	external utilities
(a,2) (b,3) (c,1) (g,1)	5 4 3 2
(b,3) (c,2) (d,3) (e,2)	5 2 9 3
(a,2) (b,1) (c,3) (d,4)	2 3 5 6
(a,3) (c,2) (d,1) (f,2)	1 3 4 6
(a,3) (b,1) (c,2) (d,1) (g,2)	2 5 3 6 1
(c,2) (d,2) (e,3) (f,1)	2 3 4 5
(a,2) (b,1) (c,1) (d,2)	5 4 3 2
(a,1) (e,2) (f,2)	4 8 3
(a,2) (b,2) (c,4) (d,2)	7 4 9 8
(b,3) (c,2) (d,2) (e,2)	5 9 10 24

Note: Duplicate items must not exist in a transaction.

What is the acceptable format of a utility databases in PAMI?

Each row in a utility database must contain only items, total sum of utilties and utility values. A sample transactional database, say sampleInputFile.txt, is provided below.

a b c g:7:2 3 1 1:5 4 3 2
b c d e:10:3 2 3 2:5 2 9 3
a b c d:10:2 1 3 4:2 3 5 6
a c d f:7:3 2 1 2:1 3 4 6
a b c d g:9:3 1 2 1 2:2 5 3 6 1
c d e f:8:2 2 3 1:2 3 4 5
a b c d:6:2 1 1 2:5 4 3 2
a e f:5:1 2 2:4 8 3
a b c d:10:2 2 4 2:7 4 9 8
b c d e:9:3 2 2 2:5 9 10 24

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 utility value exists in database
- Average utility exists in database
- Maximum utility 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.utilityDatabaseStats as stats <br>

obj = stats.utilityDatabaseStats('sampleInputFile.txt', ' ') <br>
obj.run() <br>
obj.printStats() <br>
```

What is the spatial (neighbour) database?

Spatial database contain the spatial (neighbourhood) information of items. It contains the items and its nearset neighbours satisfying the **maxDist** constraint.
A hypothetical spatial database containing items ***a, b, c, d, e, f and g*** and neighbours respectively is shown below.

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

What are the input parameters

The input parameters to a frequent pattern mining algorithm are:

- **Utility database**
  Acceptable formats:

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

- **Spatial database**
  Acceptable formats:

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

- **minUtil**
  specified in

  > - **count**

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

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

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

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into high utility spatial pattern folder.
- Enter into highUtilitySpatialPattern folder
- You will find folder like **basic**
- 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>` `<path to the neighbour file>` `<minUtil>` `<seperator>`

Example: python3 `HDSHUIM.py` `inputFile.txt` `outputFile.txt` `neighbourFile.txt` `$20$` `' '`

How to execute a high utility spatial 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.highUtilitySpatialPattern.basic.HDSHUIM as alg

iFile = 'sample_Input.txt'  #specify the input utility database <br>
minUtil = 20  #specify the minUtilvalue <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'utilityPatterns.txt'   #specify the output file name<br>
nFile = 'sampleNeighbourFile.txt'      #specify the neighbour file of dat

obj = alg.HDSHUIM(iFile, nFile, minUtil, seperator) #initialize the algor
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 utilityPatterns.txt file contains the following patterns (*format:* pattern:utility):!cat utilityPatterns.txt

In [3]:
```
!cat utilityPatterns.txt
```

```
a d : 22
a d c : 34
a c : 27
d c : 35
```

The dataframe containing the patterns is shown below:

In [4]:
```
df
```

Out[4]:

	Patterns	Support
0	a d	22
1	a d c	34
2	a c	27
3	d c	35