Mining Relative High-Utility Patterns in Utility Databases

What is Relative High-Utility 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 ***utility ratio*** no greater than user-specified ***minimum utility ratio*** threshold ***minUR***. ***minUtil*** controls the minimum utility of patterns should have and ***minUR*** controls the minimum utility ratio of patterns.

Reference: R. U. Kiran, P. Pallikila, J. M. Luna, P. Fournier-Viger, M. Toyoda and P. K. Reddy, "Discovering Relative High Utility Itemsets in Very Large Transactional Databases Using Null-Invariant Measure," 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 252-262, doi: 10.1109/BigData52589.2021.9672064.

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***.
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 utility*** is presented at left side.

Transactions	Item	Profit
(a,2) (b,3) (c,1) (g,1)	a	4
(b,3) (c,2) (d,3) (e,2)	b	3
(a,2) (b,1) (c,3) (d,4)	c	6
(a,3) (c,2) (d,1) (f,2)	d	2
(a,3) (b,1) (c,2) (d,1) (g,2)	e	5
(c,2) (d,2) (e,3) (f,1)	f	2
(a,2) (b,1) (c,1) (d,2)	g	3
(a,1) (e,2) (f,2)		
(a,2) (b,2) (c,4) (d,2)		
(b,3) (c,2) (d,2) (e,2)		

Note: Duplicate items must not exist in a transaction.

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

Understanding the statistics 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 [5]:

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

```
Database size : 10
Number of items : 7
Minimum Transaction Size : 3
Average Transaction Size : 4.0
Maximum Transaction Size : 5
Minimum utility : 3
Average utility : 11.714285714285714
Maximum utility : 19
Standard Deviation Transaction Size : 0.4472135954999579
Variance : 0.2222222222222222
Sparsity : 0.42857142857142855
```

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'

- **minUtil**
  specified in

  > - **count**

- **minUR**
  specified in

  > - [0, 1]

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

How to store the output of a relative high utility pattern mining algorithm?

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

How to run the relative high utility pattern mining algorithms in a terminal?

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into relative high utility pattern folder.
- Enter into relativeHighUtilityPatterns folder.
- You will find folder like **basic**
- Enter into the basic folder and execute the following command on terminal.

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

Example: python3 `RHUIM.py` `inputFile.txt` `outputFile.txt` `20` `0.4` `' '`

How to execute a Relative High utility 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.relativeHighUtilityPatterns.basic.RHUIM as alg

iFile = 'sample_Input.txt'  #specify the input utility database <
minUtil = 20                #specify the minUtil value
minUR = 0.4                 #specify the minUR value
seperator = ' '               #specify the seperator. Default seperator i
oFile = 'relativePatterns.txt'   #specify the output file name

obj = alg.RHUIM(iFile, minUtil, minUR, seperator) #initialize the algorit
obj.startMine()                       #start the mining process
obj.savePatterns(oFile)               #store the patterns in file
df = obj.getPatternsAsDataFrame()     #Get the patterns discovered into a
obj.printStats()                      #Print the statistics of mining pro
```

The relativePatterns.txt file contains the following patterns (*format:* pattern:utility:utility ratio):!cat relativePatterns.txt

In [3]:

```
!cat relativePatterns.txt
```

```
e d c : 20:0.4444444444444444
a b d : 23:0.5
a b d c : 33:0.5076923076923077
a b c : 30:0.625
a d : 22:0.6875
a d c : 34:0.6666666666666666
a c : 27:0.7941176470588235
b d : 25:0.8064516129032258
b d c : 39:0.78
b c : 29:0.8787878787878788
d c : 35:0.9722222222222222
```

The dataframe containing the patterns is shown below:

In [4]:

```
df
```

Out[4]:

	Patterns	Utility:UtilityRatio
0	e d c	20:0.4444444444444444
1	a b d	23:0.5
2	a b d c	33:0.5076923076923077
3	a b c	30:0.625
4	a d	22:0.6875
5	a d c	34:0.6666666666666666
6	a c	27:0.7941176470588235
7	b d	25:0.8064516129032258
8	b d c	39:0.78
9	b c	29:0.8787878787878788
10	d c	35:0.9722222222222222