About Code

Aurora

November 5, 2016

1 How to Run

Run util.TransactionAnalysis, input path of testcase file with standard input stream, and number of possible partitions will be printed with stantard output stream.

The testcase file should be structured as following:

```
# input
[name of party] [amount of money]
# output
[name of party] [amount of money]
```

Code will ignore any empty line. For lines begin with "#", if it contains "input", then all lines below will be considered as input parties, until reaching a line which begins with "#" and contains "output".

2 Code Structure

Packages and classes:

- `element`:
  - Node: storing party name (string) and party value (double)
  - Subset: set of nodes, which is subset of whole input set or output set of nodes
  - SubsetPair: pair of subsets (inputSubset, outputSubset), which satisfying $0 \leq$ sum of input - sum of output $\leq c$
  - SubsetPairGroup: group of subset pairs. Is a partition if all nodes in input set and output set are covered by all input subset and output subset exactly once
- `util`:
  - TransactionAnalysis: main class, almost all logics are implemented here
  - DoubleArithmetics: functional class providing methods to do arithmeticso on doubles without lost of precision

Process:

- Read in from file, construct original transaction with n input nodes and m output nodes
- Ignore small inputs, remove replicates
- Compute 2^n subsets of input nodes and 2^m subsets of output nodes

- Sort all subsets (including input and output subsets) by ascending order. For each input subset, find all output subsets which can form valid pair with it, i.e. find all $outputSubset_{j_1}, ..., outputSubset_{j_k}$ S.T. $sum(inputSubset_i) - sum(outputSubset_j) \in [0, c]$ for each input subset i. Add all t valid pairs into a list.
- Calculate all subset of this pair list, generates all 2^t possible groups of pairs. Check each group one by one.
- Return all groups of subset pairs which are valid partitions.