COL761
Report

August 20, 2023

1 Intuition

We know the time complexity of FP growth algorithm depends on threshold support. It can take too much time if we choose small threshold support. To find the optimal threshold support we need to develop some heuristic. Also if the size of dataset reduces we can run the FP growth with small threshold support. So, we have divided our dataset in multiple blocks.

2 Heuristic

The process begins by selecting an initial threshold support for a block of transactions. The next step involves running the FP-growth algorithm using this chosen support threshold. Instead of instantly deciding, we introduce a waiting period. If, during this time, the algorithm successfully identifies a pattern, we consider this specific threshold support as valid. This approach allows us to establish a practical and effective support threshold that caters to the dynamic nature of pattern discovery.

3 Algorithm

After find the optimal threshold support. We first sort the transactions based on the number of items they contain. This strategic arrangement ensures a structured approach to handling data. Subsequently, the sorted transactions are divided into distinct blocks, a process that enhances the manageability of the dataset for subsequent operations.

In each block of transactions, the FP-growth algorithm is harnessed to extract frequent patterns. By analyzing patterns within smaller blocks, the algorithm streamlines its focus, optimizing the identification of recurring itemsets. The outcome of this phase is a collection of frequent patterns obtained from different transaction blocks. The FP-growth will give frequent patterns based on their size, this will prioritize the larger size pattern.

Patterns that appear in transactions more than once are targeted for replacement with compressed representations. This approach ensures that patterns that hold repetitive significance contribute to the compressed dataset.

It's important to note that only patterns with multiple appearances are replaced, as this strategic decision takes into consideration the subsequent addition of an extra integer in the mapping, which needs to be justified by the pattern's recurrence. By following this step-by-step process, the data compression approach efficiently combines transaction sorting, block division, FP-growth mining, and targeted pattern replacement to achieve a compact yet meaningful compressed dataset, coupled

with a mapping that retains essential information.

Algorithm 1: Data Compression using FP-growth

```
Data: List of Transactions: transactions
Result: Compressed Transactions: compressed_transactions, Mapping: mapping
Sort transactions by number of items;
Divide sorted transactions into blocks;
foreach block in blocks do
    Build FP-tree from block using FP-growth algorithm;
    Mine frequent patterns from FP-tree in order of their length;
    foreach frequent pattern in frequent patterns do
        if frequent pattern appears more than once in transactions then
            Generate compressed code for frequent pattern;
            Update mapping with compressed code;
foreach transaction in transactions do
    Compressed transaction ← Replace frequent patterns with compressed codes using
     mapping;
    Append Compressed transaction to compressed_transactions;
```

4 Analysis

Table 1: Compression Ratio for Different Files

S. No	**File name**	**Compression ratio** $(1 - (\frac{size_{compressed}}{size_{decompressed}})$	**Time taken (minutes)**
1	D_small.dat	0.6211	2
2	D_medium.dat	0.2234	14
3	D_medium2.dat	0.024	5
4	D_large.dat	0.0723	42