Mining Frequent Itemsets with Bit Strings and Trie

Nuansri Denwattana

Supervisor: Dr. Janusz R Getta

University of Wollongong

NSW. 2522 Australia

nd22@uow.edu.au

Introduction of Mining Association Rules

- An association rule is an implication that determines the co-occurrence of objects in a large set of so called transactions, e.g. customer baskets.
- A formal specification of association rules is as follows:

  given a set of transactions $\{t_1, t_2, \ldots, t_n\}$, where a transaction t_i is a set of items $\{i_{i1}, i_{i2}, \ldots, i_{im}\}$

  an association rule is an expression **$A \Rightarrow B$**, where A and B are set of items.

  The *support* for an itemset is defined as a fraction of all transactions that includes $A \cup B$.

  The *confidence* of a rule $A \Rightarrow B$ is defined as $support(A \cup B)/support(A)$.
- We accept a rule as true if its support and confidence exceeds given threshold values.
- There are two main steps in mining association rules.

1. Find all sets of items (itemsets) that have transaction support above a threshold, called minimum support. Itemsets with minimum support are called **frequent itemsets**.
2. Use the frequent itemsets to generate association rules.

Most existing algorithms focused on the first step because it requires a great deal of computation, memory, and I/O, and has a significant impact on the overall performance.

A Parameterized Algorithm

- The efficiency of mining frequent itemsets can be improved in three different ways

1. Conceptual improvement: it can be an improvement of already existing algorithms or to discover new and more efficient algorithms.
2. Implementation technique improvement: as the size of the database is very large, it is important to develop appropriate structures capable of high compression as well as supporting fast frequent itemset generation.
3. Using more advanced hardware and the features offered by such hardware

- This research aims to achieve the performance improvements through invention of a new algorithm and its efficient implementation.
- For the conceptual improvement, we have already proposed a parameterized mining frequent itemsets algorithm. The main idea of the algorithm is to guess candidate itemsets in each level of an itemsets lattice by using information from the thorough analysis of data during the first scan. Instead of considering candidate itemsets level by level, the algorithm analyses candidate itemsets through *n* levels in *p* passes, where *p* is less than *n*.

A Data Structure Trie of Bit Strings

- The current work concerns implementation of the itemset discoverer using the technique based on bit strings and Trie data structure to represent transactions from the input database.
- Each node (except root) contains the following attributes:

  1. bitmap string of a set of items: m bits
  2. transaction's frequency
  3. a pointer to a child node
  4. a pointer to a sibling node

- A transaction is represented as a path from a root node to a certain node in the trie structure.
- The nodes are used to store a set of bitmap strings of itemsets (m bits) and transaction frequencies.
- The counter represents a number of repetition transactions.
- The depth of our trie is equal to an integer value of a number of frequent 1-items in L_1 divided by m.

An Example of the Trie

- Consider the following input data set:

  {{1,2,3}, {3,4,5,6}, {1,2,3,4,5,6,7,8}, {1,2,3,5,6}, {1,2,3,5,6,7,8}}.

- The first scan with a minimum support 40%,

  produces set L_1 = {1,2,3,4,5,6,7,8}.

- The trie of bit strings is then created as shown in Figure 1.

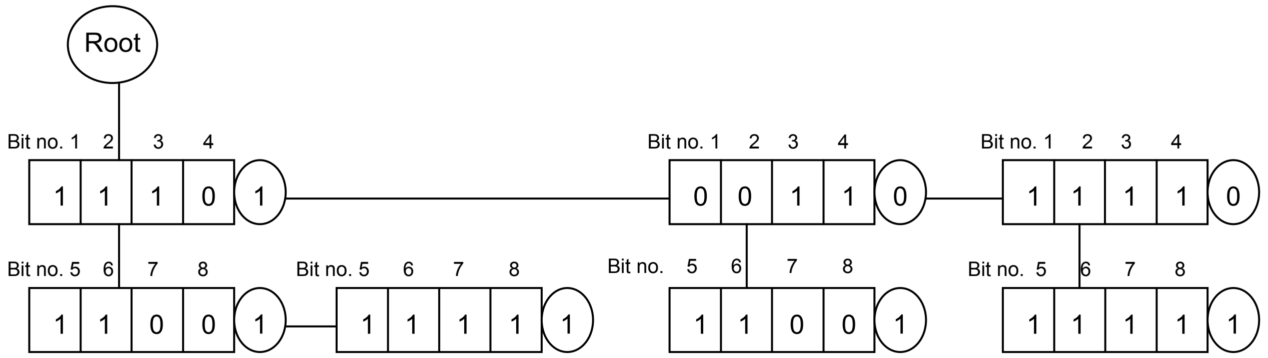

Figure 1: The trie of bit strings

Finding Frequent Itemsets

- The first pass through an input data set finds all frequent 1-items (L_1) and their frequencies.
- In the second pass, the trie of bit strings illustrated in Figure 1 is constructed in the following way.
- At the beginning, the root of the trie structure is created.
- When the first transaction is read, the frequent items in the transaction are encoded into bitmap strings. Every *m* bits forms a group and the first *m* bits are kept in the node under the root. The subsequent bitmap strings are stored in child nodes. At the end of transaction, the frequency of the node represented the last group of bitmap strings increases by one.
- The next transaction is then read. If the first *m* bits of itemsets is different from the first *m* bits in the previous transaction, these *m* bits are put in a sibling node of the node under the root and subsequent bits are put in the child nodes of these *m* bits. These steps are repeated until all transactions are read.
- When a trie structure is ready, a parameterized algorithm is applied to mine frequent itemsets from the trie of bit strings.

Experimental Results

We conducted a number of experiments on large data sets to evaluate the performance of the new technique as shown in Figures 2-5.

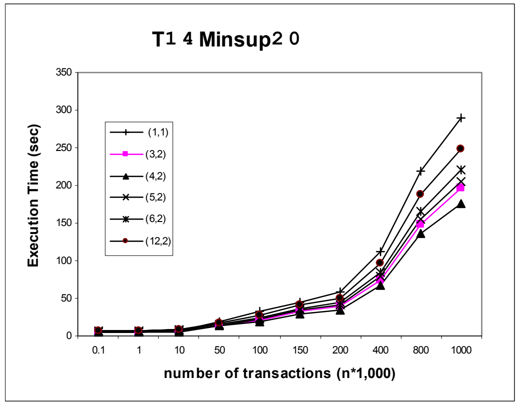

Figure 2: Scalability with number of transactions

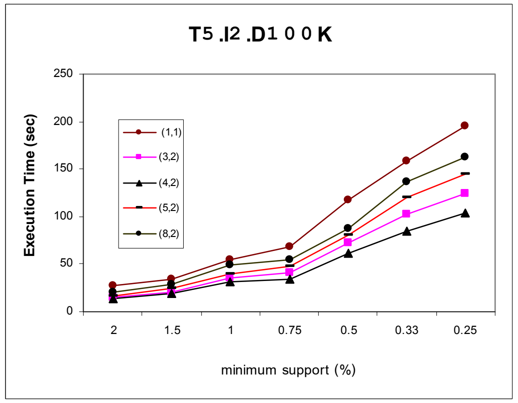

Figure 3: Scalability with threshold (minimum support)

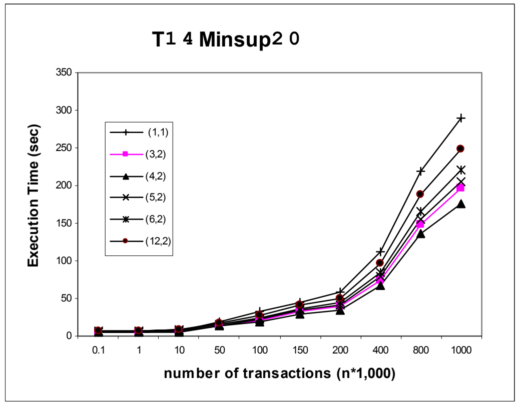

Figure 4: Scalability with parameter ***n***

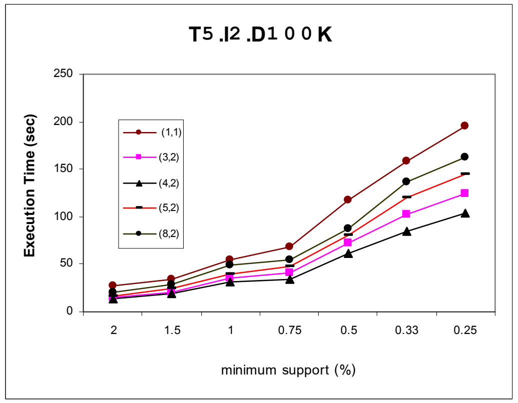

Figure 5: Scalability with parameter ***p***