LCIM: Mining Low Cost High Utility Itemsets

M. Saqib Nawaz, Philippe Fournier-Viger, Naji Alhusaini, Yulin He, Youxi Wu

Abstract

Abstract.This paper presents the LCIM algorithm, a novel approach in data mining for identifying high utility itemsets with low associated costs. This method is particularly impactful in areas like e-learning, where it's essential to maximize the benefits while minimizing the costs, such as time and effort. LCIM integrates a novel lower bound on average cost, called Average Cost Bound (ACB), to effectively reduce the search space, demonstrating efficiency and the ability to reveal interesting patterns in data.

Keywords: Data Mining, Utility Itemsets, Cost Efficiency, LCIM Algorithm.

1 Introduction

Data mining and pattern recognition in large datasets have increasingly become vital in extracting valuable information and insights. One of the significant challenges in this domain is identifying high utility itemsets, especially considering the associated costs. Traditional methods primarily focus on the benefits or utility of patterns, often overlooking the cost aspect (e.g., time, effort, or financial resources consumed). This paper introduces a novel approach to address this gap, proposing the LCIM (Low Cost Itemset Miner) algorithm. LCIM aims to discover patterns that not only offer high utility but also entail lower costs, making it particularly beneficial in e-learning environments where the optimization of both learning outcomes (utility) and invested resources (cost) is crucial.

2 Related works

2.1 Frequent Itemset MIning

Frequent Itemset Mining (FIM) algorithms, such as Apriori, FP-Growth, and Eclat, operate on transaction databases to identify frequently occurring sets of items, or itemsets. These algorithms are efficient because they do not need to examine every possible itemset combination. They leverage the anti-monotonicity property of support, which states that the occurrence frequency (support) of any itemset cannot exceed that of its subsets. This principle allows these algorithms to effectively prune the search space, thereby identifying frequent itemsets more efficiently.

2.2 High Utility Itemset Mining

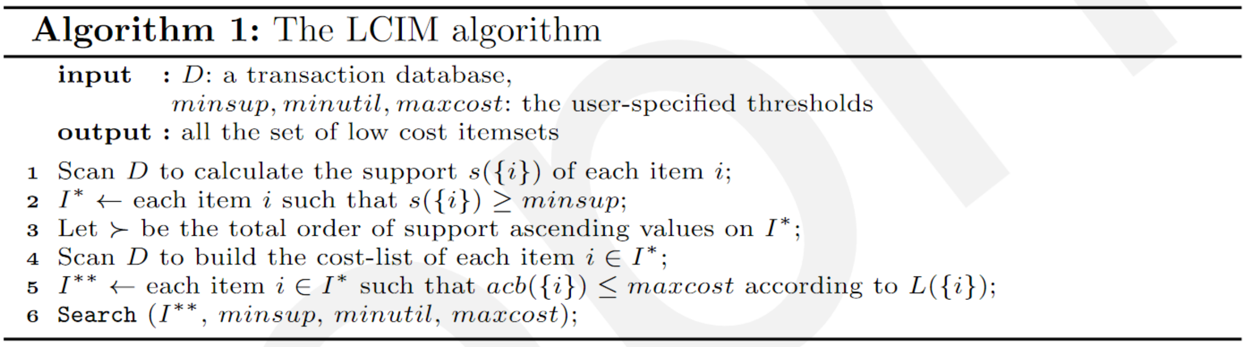
High-Utility Itemset Mining (HUIM) extends the concept of Frequent Itemset Mining (FIM) to account for the utility or importance of items in transactions, rather than just their frequency. Unlike

FIM, which only considers the occurrence frequency of itemsets, HUIM incorporates a utility value for each item, reflecting its significance in a transaction (like profit). The goal is to identify itemsets with high overall utility. This task is more complex than FIM as most utility functions do not exhibit anti-monotonicity, a property fundamental to FIM's efficiency. HUIM algorithms, therefore, use different strategies, including various search methods and data formats, along with anti-monotonic upper bounds on utility to manage the search space. Unlike previous studies, the integration of cost into HUIM is yet to be explored extensively.

3 Method

3.1 Overview

Figure 1:



Algorithm 1: The LCIM algorithm

```
input  : $D$: a transaction database,
         $minsup, minutil, maxcost$: the user-specified thresholds
output : all the set of low cost itemsets
1 Scan $D$ to calculate the support $s(\{i\})$ of each item $i$;
2 $I^* \leftarrow$ each item $i$ such that $s(\{i\}) \geq minsup$;
3 Let $\succ$ be the total order of support ascending values on $I^*$;
4 Scan $D$ to build the cost-list of each item $i \in I^*$;
5 $I^{**} \leftarrow$ each item $i \in I^*$ such that $acb(\{i\}) \leq maxcost$ according to $L(\{i\})$;
6 Search ($I^{**}$, $minsup$, $minutil$, $maxcost$);
```

Figure 1. Overview of the method

3.2 Search space exploration and pruning properties

Definition 2 (Extension). Two itemsets X and Y can be joined together to obtain a new itemset $Z = X \cup Y$ if all items in X and Y are the same except the last one according to .(like the order in the alphabet) The itemset Z is then said to be an extension of X and Y .

Property 1 (Support pruning). For any two itemsets X ∈ Y , s(X) ≥ s(Y).(is proven before in the itemset mining topic, like the support of a is always bigger than its superset(like a,b,a,b,c,etc.).)

Definition 3 (Lower bound on the cost).

1. The sequence of cost values of X is the unique sequence: $A(X) = (a_i)_{i=1}^{N}\ where\ a_i = c(X, T_i')$

2. The K largest cost values of X is the sequence:$A(X)^{(K)} = (c_i)_{i=1}^{K}$

3. The average cost bound (ACB) of X is defined as $acb(X) = \frac{\sum_{c_i \in A(X)^{minsup}} c_i}{s(X)}$.

For instance, let minsup = 1 and X = b, c. The sequence of cost values of X is A(X) = ⟨11, 11, 6⟩. Then, sort(A(X)) = ⟨6, 11, 11⟩, and $A(X)^{(1)}$ = ⟨6⟩. The average cost bound of X is acb(X) = 6/3 = 2.If minsup = 2, $A(X)^{(2)}$ = ⟨6, 11⟩, and acb(X) = $\frac{(6+11)}{3}$= $\frac{17}{2}$.

Property 2 (lower bound of the ACB on the average cost). For any itemset X, the average cost bound of X is a lower bound on the average cost of X. In other words, acb(X) $\geq$ ac(X).

Property 3 (anti-monotonicity of the ACB). For any two itemsets X $\subseteq$ Y , then acb(X) $\leq$ acb(Y).

Property 4 (Search space pruning using the ACB). For an itemset X, if acb(X) ¿ maxcost, X and its supersets are not low cost itemsets.

3.3 The cost-list data structure

Another key consideration for the design of an efficient algorithm is how to efficiently calculate the utility and cost values of itemsets and also the ACB lower bound for reducing the search space. For this purpose, the designed LCIM algorithm relies on a novel data structure called the cost-list.

Definition 4 (cost-list). The cost-list of an itemset X is a tuple L(X) = (utility, cost, tids, costs) that stores information in four fields. The field utility contains u(X). The field cost stores c(X). The field tids stores g(X), while costs stores A(X). In the following, the notation L(X).field refers to the value of the field field in L(X).

The cost-lists of an itemset X is useful as it contains all the required information about it. The cost-list of X allows to directly obtain its support as s(X) = ($|L(X).tids|$), its average utility as au(X) = L(X).utility/s(X), and its average cost as ac(X) = L(X).utility/s(X). Moreover, the ACB lower bound can be calculated by finding the minsup smallest values in L(X).costs.

Let there be two itemsets X and Y that are joined to obtain an extension Z = X $\cup$ Y . The cost-list L(Z) is derived directly from the cost-lists L(X) and L(Y) as follows. The field L(Z).costs is obtained by merging the cost values corresponding to the same transactions in L(X).cost and L(Y).costs. The field L(Z).tids = L(X).tids $\cap$ L(Y).tids. The field L(Z).cost is the sum of values in L(Z).costs. The field L(Z).utility is calculated as the sum of utility values for transactions in L(Z).tids.

4 Implementation details

4.1 Comparing with the released source codes

In the existing codebase, I implemented the following modifications:

I utilized the Random class to generate datasets of varying sizes. This approach was designed to test and compare the performance of the LCIM (Large-scale Complex Information Management) algorithm against traditional algorithms under different data scales.

Through this analysis, I observed a noteworthy trend: while traditional algorithms may exhibit higher efficiency with smaller data volumes, the LCIM algorithm significantly outperforms them as the dataset reaches a larger scale. This finding underscores the scalability and efficiency of LCIM in handling extensive data sets, a critical factor in modern data processing and analysis.

A key observation from my research is the performance differential between traditional algorithms and LCIM (Large-scale Complex Information Management) based on dataset size. Specifically, within a dataset containing approximately 500 entries, traditional algorithms frequently outperform LCIM. However, a pivotal shift occurs beyond this threshold: for datasets larger than 500 entries, LCIM consistently demonstrates superior performance. This trend highlights LCIM's robust scalability and efficiency, particularly in processing and managing large volumes of data. Such insights are critical for optimizing algorithm selection based on dataset size, ensuring maximum efficiency in data processing workflows.

4.2 Experimental environment setup

The experimental setup for evaluating the LCIM algorithm was designed to test its efficiency and ability to uncover meaningful patterns. The setup included various benchmark datasets like Mushroom, Accidents, and E-Learning. These datasets varied in size, complexity, and nature (some with synthetic values and others with real-world data). The performance of LCIM was compared with a baseline version of the algorithm and optimizations. The experiments were conducted on a specific hardware setup, ensuring a controlled environment for accurate results.

4.3 Main contributions

The paper introduced the novel problem of low-cost high utility itemset mining, focusing on finding patterns that balance high utility with low cost. A new algorithm, LCIM (Low Cost Itemset Miner), was developed for efficient mining, incorporating a unique Average Cost Bound (ACB) for search space reduction. The LCIM algorithm also featured a novel cost-list data structure for efficient utility and cost calculations. The research provided empirical evidence of LCIM's efficiency through extensive experiments across various datasets. The results demonstrated that LCIM could uncover interesting and meaningful patterns, particularly in e-learning data.

5 Results and analysis

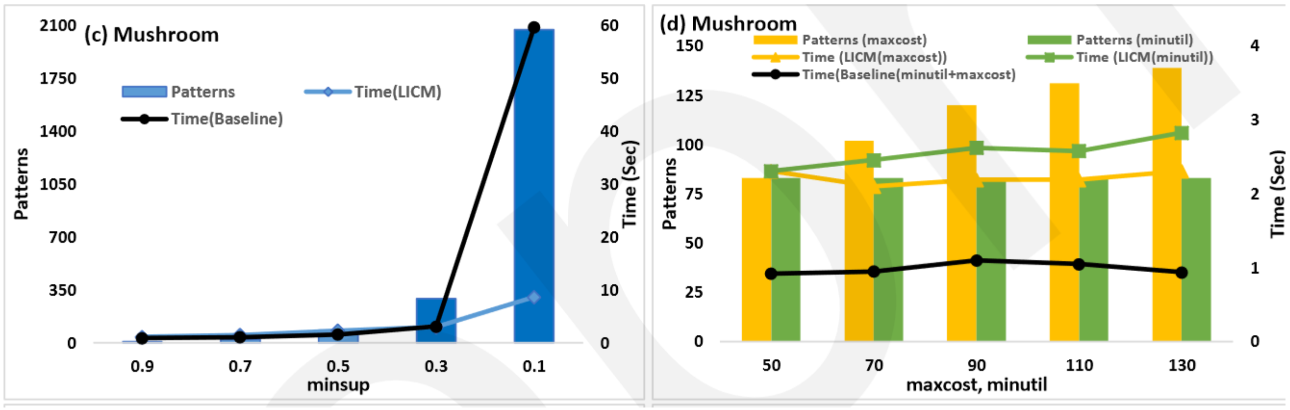
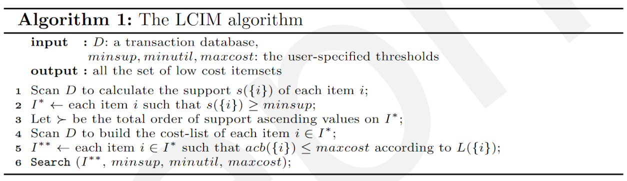


Figure 2. Experimental results



```
Algorithm 1: The LCIM algorithm
  input  : $D$: a transaction database,
           $minsup, minutil, maxcost$: the user-specified thresholds
  output : all the set of low cost itemsets
1 Scan $D$ to calculate the support $s(\{i\})$ of each item $i$;
2 $I^* \leftarrow$ each item $i$ such that $s(\{i\}) \geq minsup$;
3 Let $\succ$ be the total order of support ascending values on $I^*$;
4 Scan $D$ to build the cost-list of each item $i \in I^*$;
5 $I^{**} \leftarrow$ each item $i \in I^*$ such that $acb(\{i\}) \leq maxcost$ according to $L(\{i\})$;
6 Search ($I^{**}$, $minsup$, $minutil$, $maxcost$);
```

Figure 3. Running time

6 Conclusion and future work

This paper presented a novel problem of low cost high utility itemset mining (finding patterns that have a high average utility but a low average-cost). An efficient algorithm named LCIM (Low Cost Itemset Miner) was presented to solve this problem efficiently. It introduces a lower bound on the average cost called Average Cost Bound (ACB) to reduce the search space, and a cost-list data structure. Experiments have shown that LCIM is efficient and can find interesting patterns in e-learning data. In future work, alternative ways of integrating utility and cost will be studied. [1].

References

[1] M. Saqib Nawaz, Philippe Fournier-Viger, Naji Alhusaini, Yulin He, Youxi Wu, and Debdatta Bhattacharya. Lcim: Mining low cost high utility itemsets. In *Multi-Disciplinary Trends in Artificial Intelligence: 15th International Conference, MIWAI 2022, Virtual Event, November 17–19, 2022, Proceedings*, page 73–85, 2022.