# MFHOI-Miner: An Efficient Method for Mining Maximal Frequent High-Occupancy Itemsets

Quan Van
Independent Researcher
vanhaminhquan2406@gmail.com

**Abstract**

Frequent itemset mining discovers itemsets that occur frequently in transactional databases, while maximal frequent itemset mining reduces output redundancy by preserving only frequent itemsets without frequent supersets. Occupancy-based pattern mining evaluates how completely an itemset occupies the transactions in which it appears. However, existing approaches optimize these objectives separately and do not directly discover patterns that are simultaneously frequent, high-occupancy, and maximal under an occupancy-dominance criterion. This paper introduces the problem of mining *Maximal Frequent High-Occupancy Itemsets* (MFHOIs). An itemset is an MFHOI if it is frequent, has average occupancy above a user-defined threshold, and is not dominated by any strict frequent high-occupancy superset with equal or higher average occupancy. We formalize weak and strong variants of MFHOI, prove their theoretical properties, analyze the non-monotonicity of average occupancy, and propose MFHOI-Miner, an exact vertical bitset-based depth-first algorithm. The proposed method uses support anti-monotonicity, safe occupancy upper bounds, and dominance-aware filtering to mine MFHOIs efficiently. We also present an experimental protocol for comparing MFHOI-Miner with frequent itemset mining, maximal frequent itemset mining, and high-occupancy itemset mining algorithms. The proposed framework provides a concise and occupancy-aware representation of dense frequent patterns.

**Keywords:** Frequent itemset mining; maximal frequent itemset mining; high-occupancy itemset mining; occupancy-based pattern mining; concise pattern representation; vertical bitset mining.

## 1 Introduction

Frequent itemset mining is one of the fundamental problems in data mining. Given a transaction database, the goal is to discover all itemsets whose support is no less than a user-defined minimum support threshold. The Apriori algorithm established the support-based view of frequent itemset mining and introduced the well-known downward closure property of support [**?**]. Later, FP-Growth improved scalability by mining frequent patterns without candidate generation using the FP-tree structure [**?**]. Eclat further popularized the vertical transaction-id representation, where support can be computed through intersections of transaction-id sets [**?**].

Although frequent itemset mining is useful, it often generates a very large number of patterns, especially when the minimum support threshold is low or the database is dense. To reduce redundancy, maximal frequent itemset mining reports only frequent itemsets that have no frequent supersets. Representative algorithms include MAFIA [**?**, **?**], GenMax [**?**], and FPmax* [**?**]. These algorithms show that maximality can substantially compress frequent itemset outputs.

However, frequency and maximality alone do not measure how much of a supporting transaction is represented by an itemset. A frequent itemset may occur in many transactions but occupy only a small portion of each transaction. To address this limitation, occupancy-based pattern mining evaluates the ratio between the size of an itemset and the size of the transactions in which it occurs. Occupancy has been used to measure the completeness or density of a pattern within its supporting transactions [**?**, **?**]. High-occupancy itemset mining was later proposed to directly discover itemsets with large average

occupancy [**?**]. Recent works have also studied transaction occupancy, fast high-occupancy itemset mining, and high-utility occupancy pattern mining [**?**, **?**, **?**, **?**].

Despite these advances, existing approaches do not directly address the following question:

> *Can we mine itemsets that are frequent, high-occupancy, and not redundant with respect to larger high-occupancy frequent supersets?*

To answer this question, we propose a new itemset pattern type called the *Maximal Frequent High-Occupancy Itemset* (MFHOI). An itemset is an MFHOI if it satisfies both frequency and high-occupancy requirements and is not dominated by any strict frequent high-occupancy superset with equal or higher average occupancy. This definition combines three important dimensions:

(1) **Frequency**: the itemset must appear in enough transactions.

(2) **Occupancy**: the itemset must occupy a large portion of its supporting transactions.

(3) **Dominance-based maximality**: the itemset must not be dominated by a larger frequent high-occupancy itemset.

This paper makes the following contributions:

- We introduce the problem of mining Maximal Frequent High-Occupancy Itemsets.
- We define weak and strong MFHOI variants using occupancy-dominance relations.
- We prove key theoretical properties, including support anti-monotonicity and average occupancy non-monotonicity.
- We propose safe occupancy upper bounds for pruning the search space.
- We design MFHOI-Miner, an exact vertical bitset-based algorithm for mining MFHOIs.
- We present a complete experimental methodology for evaluating MFHOI-Miner against frequent, maximal frequent, and occupancy-based itemset mining baselines.

# 2 Related Work

## 2.1 Frequent Itemset Mining

Frequent itemset mining was popularized by Agrawal and Srikant through the Apriori algorithm [**?**]. Apriori relies on the anti-monotonicity of support: if an itemset is infrequent, then all of its supersets are also infrequent. This property enables candidate pruning but still requires multiple database scans and can generate many candidates.

FP-Growth improves efficiency by compressing the database into an FP-tree and mining frequent patterns through recursive conditional pattern bases [**?**]. Eclat uses a vertical database layout, representing each itemset by the set of transaction identifiers containing it. Support is computed by intersecting transaction-id sets [**?**]. This vertical representation is especially useful for depth-first mining and is also suitable for bitset-based implementations.

### 2.2 Maximal Frequent Itemset Mining

Maximal frequent itemset mining aims to reduce output size by returning only frequent itemsets that have no frequent supersets. MAFIA integrates depth-first search with bitmap-based support counting and maximality pruning [**?**, **?**]. GenMax uses progressive focusing and efficient maximality checking to mine maximal frequent itemsets [**?**]. FPmax* uses FP-tree-based mining together with maximal frequent itemset trees for efficient maximality checking [**?**].

Maximal frequent itemset mining is related to MFHOI mining because both aim to produce concise itemset representations. However, the maximality condition is different. Maximal frequent itemsets are maximal only under support, while MFHOIs are maximal under an occupancy-dominance criterion among frequent high-occupancy itemsets.

### 2.3 Occupancy-Based Pattern Mining

Occupancy-based pattern mining evaluates the completeness of a pattern in the transactions where it appears. Tang et al. introduced occupancy into frequent pattern mining for high-quality pattern recommendation [**?**]. Zhang et al. later developed occupancy-based frequent pattern mining as a more systematic framework [**?**]. Deng proposed high-occupancy itemset mining to discover itemsets with high average occupancy [**?**]. Further improvements include transaction-occupancy-aware mining [**?**], equivalence-class and early-pruning techniques [**?**], and utility-occupancy pattern mining [**?**, **?**].

MFHOI mining differs from ordinary high-occupancy itemset mining because it also requires frequency and removes patterns that are dominated by larger frequent high-occupancy supersets.

## 3 Preliminaries

Let

$$\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$$

be a finite set of distinct items. A transaction database is denoted as

$$\mathcal{D} = \{T_1, T_2, \ldots, T_n\},$$

where each transaction $T_j$ is a non-empty subset of $\mathcal{I}$. An itemset is a non-empty subset $X \subseteq \mathcal{I}$. A transaction $T$ contains $X$ if $X \subseteq T$.

**Definition 1** (Supporting Transaction Set)**.** *The supporting transaction set of an itemset $X$ is*

$$\Gamma(X) = \{T \in \mathcal{D} \mid X \subseteq T\}.$$

**Definition 2** (Support)**.** *The support count of $X$ is*

$$\sup(X) = |\Gamma(X)|.$$

**Definition 3** (Frequent Itemset)**.** *Given a minimum support threshold $\sigma$, an itemset $X$ is frequent if*

$$\sup(X) \geq \sigma.$$

**Definition 4** (Transaction Occupancy)**.** *For an itemset $X$ and a transaction $T$ such that $X \subseteq T$, the occupancy of $X$ in $T$ is*

$$\mathrm{occ}(X, T) = \frac{|X|}{|T|}.$$

**Definition 5** (Average Occupancy)**.** *The average occupancy of $X$ over its supporting transactions is*

$$\mathrm{aocc}(X) = \frac{1}{\sup(X)} \sum_{T \in \Gamma(X)} \frac{|X|}{|T|}.$$

*Equivalently,*

$$\mathrm{aocc}(X) = |X| \cdot \frac{1}{\sup(X)} \sum_{T \in \Gamma(X)} \frac{1}{|T|}.$$

**Definition 6** (High-Occupancy Itemset)**.** *Given a minimum occupancy threshold $\beta$, an itemset $X$ is high-occupancy if*

$$\mathrm{aocc}(X) \geq \beta.$$

# 4 Maximal Frequent High-Occupancy Itemsets

## 4.1 Frequent High-Occupancy Itemsets

**Definition 7** (Frequent High-Occupancy Itemset)**.** *Given a transaction database $\mathcal{D}$, a minimum support threshold $\sigma$, and a minimum occupancy threshold $\beta$, an itemset $X$ is a Frequent High-Occupancy Itemset if*

$$\sup(X) \geq \sigma$$

*and*

$$\mathrm{aocc}(X) \geq \beta.$$

*The set of all frequent high-occupancy itemsets is*

$$\mathcal{FHO}(\mathcal{D}, \sigma, \beta) = \{X \subseteq \mathcal{I} \mid \sup(X) \geq \sigma \wedge \mathrm{aocc}(X) \geq \beta\}.$$

## 4.2 Weak MFHOI

**Definition 8** (Weak Maximal Frequent High-Occupancy Itemset)**.** *An itemset $X$ is a weak MFHOI if*

$$X \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta)$$

*and there does not exist an itemset $Y$ such that*

$$Y \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta),$$

$$X \subset Y,$$

*and*

$$\mathrm{aocc}(Y) > \mathrm{aocc}(X).$$

*The set of all weak MFHOIs is*

$$\mathcal{MFHO}^{>}(\mathcal{D}, \sigma, \beta) = \{X \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta) \mid \nexists Y \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta) : X \subset Y \wedge \mathrm{aocc}(Y) > \mathrm{aocc}(X)\}.$$

### 4.3 Strong MFHOI

**Definition 9** (Strong Maximal Frequent High-Occupancy Itemset)**.** *An itemset $X$ is a strong MFHOI if*

$$X \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta)$$

*and there does not exist an itemset $Y$ such that*

$$Y \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta),$$

$$X \subset Y,$$

*and*

$$\mathrm{aocc}(Y) \geq \mathrm{aocc}(X).$$

*The set of all strong MFHOIs is*

$$\mathcal{MFHO}^{\geq}(\mathcal{D}, \sigma, \beta) = \{X \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta) \mid \nexists Y \in \mathcal{FHO}(\mathcal{D}, \sigma, \beta) : X \subset Y \wedge \mathrm{aocc}(Y) \geq \mathrm{aocc}(X)\}.$$

**Remark 1.** *The strong variant is stricter than the weak variant because it removes an itemset even when a larger frequent high-occupancy superset has equal average occupancy.*

**Theorem 1.** *For any database $\mathcal{D}$, support threshold $\sigma$, and occupancy threshold $\beta$,*

$$\mathcal{MFHO}^{\geq}(\mathcal{D}, \sigma, \beta) \subseteq \mathcal{MFHO}^{>}(\mathcal{D}, \sigma, \beta) \subseteq \mathcal{FHO}(\mathcal{D}, \sigma, \beta).$$

*Proof.* Every weak or strong MFHOI must first be a member of $\mathcal{FHO}(\mathcal{D}, \sigma, \beta)$. Moreover, the strong dominance condition uses $\mathrm{aocc}(Y) \geq \mathrm{aocc}(X)$, while the weak condition uses only $\mathrm{aocc}(Y) > \mathrm{aocc}(X)$. Therefore, any itemset removed by the weak rule is also removed by the strong rule, and the strong output is a subset of the weak output. □

## 5 Theoretical Properties

### 5.1 Support Anti-Monotonicity

**Theorem 2** (Support Anti-Monotonicity)**.** *For any two itemsets $X$ and $Y$, if $X \subseteq Y$, then*

$$\sup(Y) \leq \sup(X).$$

*Proof.* If $X \subseteq Y$, then every transaction containing $Y$ also contains $X$. Therefore,

$$\Gamma(Y) \subseteq \Gamma(X).$$

Taking cardinalities gives

$$|\Gamma(Y)| \leq |\Gamma(X)|.$$

Thus,

$$\sup(Y) \leq \sup(X).$$

□

**Property 1** (Safe Support Pruning)**.** *If $X$ is infrequent, then every strict superset of $X$ is also infrequent. Thus, the subtree rooted at $X$ can be safely pruned.*

### 5.2 Average Occupancy Non-Monotonicity

**Theorem 3.** *Average occupancy is neither monotone nor anti-monotone in the itemset lattice.*

*Proof.* Consider the database

$$T_1 = \{a, b\}, \quad T_2 = \{a, c, d, e, f\}, \quad T_3 = \{a, c, d, e, f\}.$$

For $X = \{a\}$,

$$\mathrm{aocc}(X) = \frac{1}{3}\left(\frac{1}{2} + \frac{1}{5} + \frac{1}{5}\right) = 0.3.$$

For $Y = \{a, b\}$, we have $X \subset Y$, and $Y$ occurs only in $T_1$. Thus,

$$\mathrm{aocc}(Y) = \frac{2}{2} = 1.$$

Therefore,

$$\mathrm{aocc}(Y) > \mathrm{aocc}(X),$$

showing that average occupancy is not anti-monotone. Conversely, extending an itemset can also decrease average occupancy when the extension appears in longer transactions or loses shorter supporting transactions. Hence average occupancy is not monotone either. □

**Property 2** (Unsafe Occupancy Pruning)**.** *The condition* $\mathrm{aocc}(X) < \beta$ *cannot be used to prune all supersets of* $X$*, because a strict superset* $Y$ *may still satisfy* $\mathrm{aocc}(Y) \geq \beta$*.*

### 5.3 Difference from Maximal Frequent Itemsets

**Definition 10** (Maximal Frequent Itemset)**.** *An itemset* $X$ *is a maximal frequent itemset if*

$$\sup(X) \geq \sigma$$

*and there does not exist* $Y$ *such that*

$$X \subset Y$$

*and*

$$\sup(Y) \geq \sigma.$$

**Theorem 4.** *An MFHOI is not necessarily a maximal frequent itemset, and a maximal frequent itemset is not necessarily an MFHOI.*

*Proof.* An MFHOI may have a frequent superset $Y$, as long as $Y$ is not high-occupancy or does not dominate $X$ by average occupancy. Hence an MFHOI is not necessarily a maximal frequent itemset.

Conversely, a maximal frequent itemset may fail to be high-occupancy, meaning $\mathrm{aocc}(X) < \beta$. Therefore, a maximal frequent itemset is not necessarily an MFHOI. □

# 6 Safe Occupancy Upper Bounds

Let a search node be denoted as

$$N = (P, E),$$

where $P$ is the current prefix itemset and $E$ is the set of possible extension items. Any descendant of $N$ has the form

$$Y = P \cup Z,$$

where $Z \subseteq E$.

For every transaction $T \in \Gamma(P)$, define the extension capacity of $T$ at node $N$ as

$$\mathrm{cap}_N(T) = |E \cap T|.$$

## 6.1 Support-Constrained Size Bound

Let

$$c_{(1)} \geq c_{(2)} \geq \dots$$

be the values of $\mathrm{cap}_N(T)$ sorted in descending order over all $T \in \Gamma(P)$.

**Theorem 5.** *For any descendant $Y = P \cup Z$ such that $\sup(Y) \geq \sigma$,*

$$|Y| \leq |P| + c_{(\sigma)}.$$

*Proof.* Let $d = |Z|$. Since $Y$ is supported by at least $\sigma$ transactions, there must be at least $\sigma$ transactions $T \in \Gamma(P)$ such that $Z \subseteq E \cap T$. Therefore, $d \leq |E \cap T|$ for at least $\sigma$ transactions. Hence $d \leq c_{(\sigma)}$, and

$$|Y| = |P| + d \leq |P| + c_{(\sigma)}.$$

□

Let

$$r(T) = \frac{1}{|T|},$$

and let

$$r_{(1)} \geq r_{(2)} \geq \dots$$

be the reciprocal transaction lengths sorted in descending order over $T \in \Gamma(P)$. Define

$$\overline{r}_N^{top\sigma} = \frac{1}{\sigma} \sum_{j=1}^{\sigma} r_{(j)}.$$

The first upper bound is

$$UB_1(N) = \big(|P| + c_{(\sigma)}\big) \cdot \overline{r}_N^{top\sigma}.$$

**Theorem 6.** *For every descendant $Y$ of node $N$ such that $\sup(Y) \geq \sigma$,*

$$\mathrm{aocc}(Y) \leq UB_1(N).$$

*Proof.* From the previous theorem,

$$|Y| \leq |P| + c_{(\sigma)}.$$

Also, $\Gamma(Y) \subseteq \Gamma(P)$, and $|\Gamma(Y)| \geq \sigma$. The largest possible average reciprocal transaction length over any supporting set of size at least $\sigma$ is bounded by the average of the top $\sigma$ reciprocal lengths in $\Gamma(P)$. Therefore,

$$\text{aocc}(Y) = |Y| \cdot \frac{1}{|\Gamma(Y)|} \sum_{T \in \Gamma(Y)} \frac{1}{|T|} \leq \big(|P| + c_{(\sigma)}\big) \cdot \overline{r}_N^{top\sigma}.$$

□

**Property 3** (UB1 Pruning)**.** *If*

$$UB_1(N) < \beta,$$

*then no descendant of $N$ can be a frequent high-occupancy itemset. Thus, node $N$ can be safely pruned.*

### 6.2 Top-$\sigma$ Occupancy Envelope Bound

For every $T \in \Gamma(P)$, define

$$u_N(T) = \frac{|P| + \text{cap}_N(T)}{|T|}.$$

Let

$$u_{(1)} \geq u_{(2)} \geq \dots$$

be the sorted values of $u_N(T)$ over all $T \in \Gamma(P)$. Define

$$UB_2(N) = \frac{1}{\sigma} \sum_{j=1}^{\sigma} u_{(j)}.$$

**Theorem 7.** *For every descendant $Y$ of node $N$ such that* $\sup(Y) \geq \sigma$*,*

$$\text{aocc}(Y) \leq UB_2(N).$$

*Proof.* For any transaction $T \in \Gamma(Y)$, the additional items in $Y \setminus P$ must be contained in $E \cap T$. Therefore,

$$|Y| \leq |P| + \text{cap}_N(T).$$

Thus,

$$\text{occ}(Y, T) = \frac{|Y|}{|T|} \leq \frac{|P| + \text{cap}_N(T)}{|T|} = u_N(T).$$

Since $\Gamma(Y) \subseteq \Gamma(P)$ and $|\Gamma(Y)| \geq \sigma$, the largest possible average of $u_N(T)$ over $\Gamma(Y)$ is bounded by the average of the top $\sigma$ values. Hence,

$$\text{aocc}(Y) \leq UB_2(N).$$

□

**Property 4** (UB2 Pruning)**.** *If*

$$UB_2(N) < \beta,$$

*then no descendant of $N$ can be a frequent high-occupancy itemset. Thus, node $N$ can be safely pruned.*

# 7 MFHOI-Miner

## 7.1 Vertical Bitset Representation

MFHOI-Miner uses a vertical database representation. For each item $i$, the algorithm stores a transaction-id set:

$$\mathrm{TID}(i) = \{tid(T) \mid i \in T\}.$$

For an itemset

$$X = \{i_1, i_2, \ldots, i_k\},$$

its transaction-id set is computed as

$$\mathrm{TID}(X) = \mathrm{TID}(i_1) \cap \mathrm{TID}(i_2) \cap \cdots \cap \mathrm{TID}(i_k).$$

Then,

$$\sup(X) = |\,\mathrm{TID}(X)|.$$

In the implementation, each transaction-id set is represented as a packed 64-bit bitset, so intersection is performed using bitwise AND operations and support is computed using population count instructions.

## 7.2 Average Occupancy Computation

Let $L(t)$ be the length of transaction $t$. Precompute

$$w(t) = \frac{1}{L(t)}.$$

Then for any itemset $X$,

$$\mathrm{aocc}(X) = |X| \cdot \frac{1}{|\,\mathrm{TID}(X)|} \sum_{t \in \mathrm{TID}(X)} w(t).$$

This makes average occupancy computation efficient after obtaining the supporting transaction-id set.

## 7.3 Mining Strategy

MFHOI-Miner has two exact modes:

(1) **Exact two-phase mode**: mine all frequent high-occupancy itemsets, then apply weak or strong dominance filtering.

(2) **Direct MFHOI mode**: mine itemsets using support pruning, occupancy upper bounds, and dominance-aware branch pruning.

The two-phase mode is used as the correctness baseline. The direct mode is the optimized version.

### 7.4 Algorithm

**Algorithm 1** MFHOI-Miner

**Require:** Transaction database $\mathcal{D}$, minimum support $\sigma$, minimum occupancy $\beta$, variant $v \in \{weak, strong\}$
**Ensure:** Set of MFHOIs

Build vertical bitset database $V$
Precompute transaction lengths $L(t)$ and inverse lengths $w(t) = 1/L(t)$
Remove all globally infrequent 1-items
Sort remaining items by increasing support
$FHO \leftarrow \emptyset$
DFS-FHO($\emptyset$, all transaction IDs, sorted items, $FHO$)
**if** $v = weak$ **then**
    $Result \leftarrow$ WEAK-DOMINANCE-FILTER($FHO$)
**else**
    $Result \leftarrow$ STRONG-DOMINANCE-FILTER($FHO$)
**end if**
**return** $Result$

**Algorithm 2** DFS-FHO

**Require:** Prefix $P$, prefix tidset $TID_P$, extension items $E$, output list $FHO$

**for** each item $i \in E$ **do**
    $X \leftarrow P \cup \{i\}$
    $TID_X \leftarrow TID_P \cap TID(i)$
    $s \leftarrow |TID_X|$
    **if** $s < \sigma$ **then**
        **continue**
    **end if**
    $a \leftarrow$ AVERAGEOCCUPANCY$(X, TID_X)$
    **if** $a \geq \beta$ **then**
        Add $(X, s, a)$ to $FHO$
    **end if**
    $E' \leftarrow$ items after $i$ in $E$
    **if** $E' \neq \emptyset$ **then**
        Compute $UB_1(X, E')$
        **if** $UB_1(X, E') < \beta$ **then**
            **continue**
        **end if**
        Compute $UB_2(X, E')$
        **if** $UB_2(X, E') < \beta$ **then**
            **continue**
        **end if**
        DFS-FHO($X, TID_X, E', FHO$)
    **end if**
**end for**

**Algorithm 3** AVERAGEOCCUPANCY

**Require:** Itemset $X$, transaction-id set $TID_X$

$sum \leftarrow 0$
**for** each transaction id $t \in TID_X$ **do**
  $sum \leftarrow sum + 1/L(t)$
**end for**
**return** $|X| \cdot sum/|TID_X|$

**Algorithm 4** STRONG-DOMINANCE-FILTER

**Require:** List $FHO$ of frequent high-occupancy itemsets
**Ensure:** Strong MFHOI set

Sort $FHO$ by decreasing itemset length
$Processed \leftarrow \emptyset$
$Output \leftarrow \emptyset$
**for** each pattern $X \in FHO$ **do**
  $dominated \leftarrow false$
  **for** each pattern $Y \in Processed$ **do**
    **if** $X \subset Y$ and $\mathrm{aocc}(Y) \geq \mathrm{aocc}(X)$ **then**
      $dominated \leftarrow true$
      **break**
    **end if**
  **end for**
  **if** $dominated = false$ **then**
    Add $X$ to $Output$
  **end if**
  Add $X$ to $Processed$
**end for**
**return** $Output$

**Remark 2.** *The $Processed$ list must contain all previously processed FHO itemsets, including those that are themselves dominated. A dominated FHO itemset may still dominate a smaller itemset.*

# 8 Correctness Analysis

**Theorem 8.** *Algorithm* **??** *enumerates all frequent itemsets that are not safely pruned by support or valid occupancy upper bounds.*

*Proof.* The DFS procedure explores the itemset lattice according to a fixed total item order. Each itemset has exactly one ordered construction path. Support pruning is safe by support anti-monotonicity. $UB_1$ and $UB_2$ pruning are safe by their upper-bound theorems. Therefore, no frequent high-occupancy itemset is removed by the pruning rules. □

**Theorem 9.** *Every itemset inserted into $FHO$ by Algorithm* **??** *is a frequent high-occupancy itemset.*

*Proof.* An itemset $X$ is inserted into $FHO$ only if its support satisfies

$$\sup(X) \geq \sigma$$

and its average occupancy satisfies

$$\mathrm{aocc}(X) \geq \beta.$$

These are exactly the conditions of a frequent high-occupancy itemset. □

**Theorem 10.** *Algorithm* **??** *returns exactly the set of strong MFHOIs.*

*Proof.* The input $FHO$ contains all frequent high-occupancy itemsets. The algorithm sorts them by decreasing length, so any strict superset of a pattern $X$ is processed before $X$. For each $X$, the algorithm checks whether there exists a previously processed $Y$ such that $X \subset Y$ and $\mathrm{aocc}(Y) \geq \mathrm{aocc}(X)$. This is exactly the strong dominance condition. If such $Y$ exists, $X$ is not a strong MFHOI. Otherwise, $X$ is retained. Therefore, the output is exactly $\mathcal{MFHO}^{\geq}(\mathcal{D}, \sigma, \beta)$. □

# 9 Complexity Analysis

Let $n = |\mathcal{D}|$ be the number of transactions, $m = |\mathcal{I}|$ be the number of distinct items, and $m_f$ be the number of frequent 1-items after support filtering. Let

$$W = \left\lceil \frac{n}{64} \right\rceil$$

be the number of 64-bit words required to represent a vertical bitset.

In the worst case, itemset mining remains exponential because the number of possible itemsets is $2^{m_f} - 1$. For each candidate extension, bitset intersection costs

$$O(W)$$

word operations. Support counting also costs

$$O(W)$$

using population count instructions.

Average occupancy computation requires scanning the set bits of the resulting tidset. If the support of $X$ is $s$, this costs

$$O(s).$$

Thus, the enumeration phase has worst-case cost

$$O\left(2^{m_f} \cdot W + \sum_X \sup(X)\right).$$

If $q = |\mathcal{FHO}|$, naive dominance filtering costs

$$O(q^2 \cdot c),$$

where $c$ is the cost of subset checking. If itemsets are represented as bitsets, subset checking costs

$$O\left(\left\lceil \frac{m_f}{64} \right\rceil\right).$$

The vertical database requires

$$O(m_f W)$$

machine words. The FHO pattern list requires

$$O(q\bar{k}),$$

where $\bar{k}$ is the average itemset length.

# 10 Experimental Design

Since MFHOI is a newly defined itemset pattern type, no existing algorithm mines exactly the same output. Therefore, MFHOI-Miner is compared against representative algorithms from its parent families: frequent itemset mining, maximal frequent itemset mining, and high-occupancy itemset mining. The goal of the evaluation is not to show that MFHOI is always faster than classical frequent itemset mining, because the algorithms solve different output tasks. The goal is to verify whether the proposed strong dominance rule gives a compact frequent high-occupancy representation, whether it is stricter than weak MFHOI, and whether its output is different from maximal frequent itemset mining.

## 10.1 Datasets

The experiments used real transaction datasets already present in the project and synthetic transaction datasets generated for the MFHOI study. No toy or hand-written example dataset was used. Table **??** reports the dataset statistics measured from the input files used in the benchmark.

Table 1: Dataset statistics used in the MFHOI Q1 benchmark.

| Dataset | Transactions | Distinct Items | Avg. Length | Max Length |
|---|---|---|---|---|
| T10I4D100K | 100000 | 870 | 10.10 | 29 |
| Accidents | 340183 | 468 | 33.81 | 51 |
| Chess | 3196 | 75 | 37.00 | 37 |
| Connect | 67557 | 129 | 43.00 | 43 |
| FoodmartFIM | 4141 | 1559 | 4.42 | 14 |
| Mushrooms | 8416 | 119 | 23.00 | 23 |
| Pumsb | 49046 | 2113 | 74.00 | 74 |
| Retail | 88162 | 101546 | 10.31 | 76 |
| SYN-CORRELATED | 10000 | 500 | 15.84 | 46 |
| SYN-DENSE | 10000 | 200 | 51.95 | 59 |
| SYN-LONG | 10000 | 500 | 90.74 | 99 |
| SYN-SPARSE | 10000 | 1000 | 9.96 | 10 |
| T20I6D100K | 99922 | 893 | 19.90 | 47 |

## 10.2 Threshold Settings

The support threshold was tested with four values:

$$minsup\_ratio \in \{0.10, 0.05, 0.02, 0.01\}.$$

For each dataset, the support count was computed as

$$\sigma = \lceil minsup\_ratio \cdot |\mathcal{D}| \rceil.$$

The occupancy threshold was tested with

$$\beta \in \{0.20, 0.40, 0.60, 0.80\}.$$

The Apriori, FPmax, and MFI-baseline runs use only minimum support, because their target definitions do not include occupancy. The FHO, weak MFHOI, strong MFHOI, and HEP runs use both minimum support and minimum occupancy. To avoid out-of-memory failures on dense datasets, the benchmark used a per-run memory cap of 1024 MB, a wall-time protection threshold, and a maximum retained pattern limit of 20000 patterns. A run marked `LIMITED` therefore reports a protected bounded

measurement rather than an uncapped exhaustive measurement. This protection is important for reproducibility on commodity hardware and prevents dense datasets such as Chess, Connect, Mushrooms, Pumsb, SYN-DENSE, and SYN-LONG from exhausting system memory.

# 11 Visualization-Based Experimental Analysis

To improve the interpretability of the experimental results, we provide both aggregate-level and dataset-level visualizations. The aggregate figures summarize the global behavior of all algorithms across all datasets and threshold settings. The per-dataset figures show runtime, peak memory usage, output disk usage, number of discovered itemsets, and number of generated candidates.

## 11.1 Aggregate-Level Results

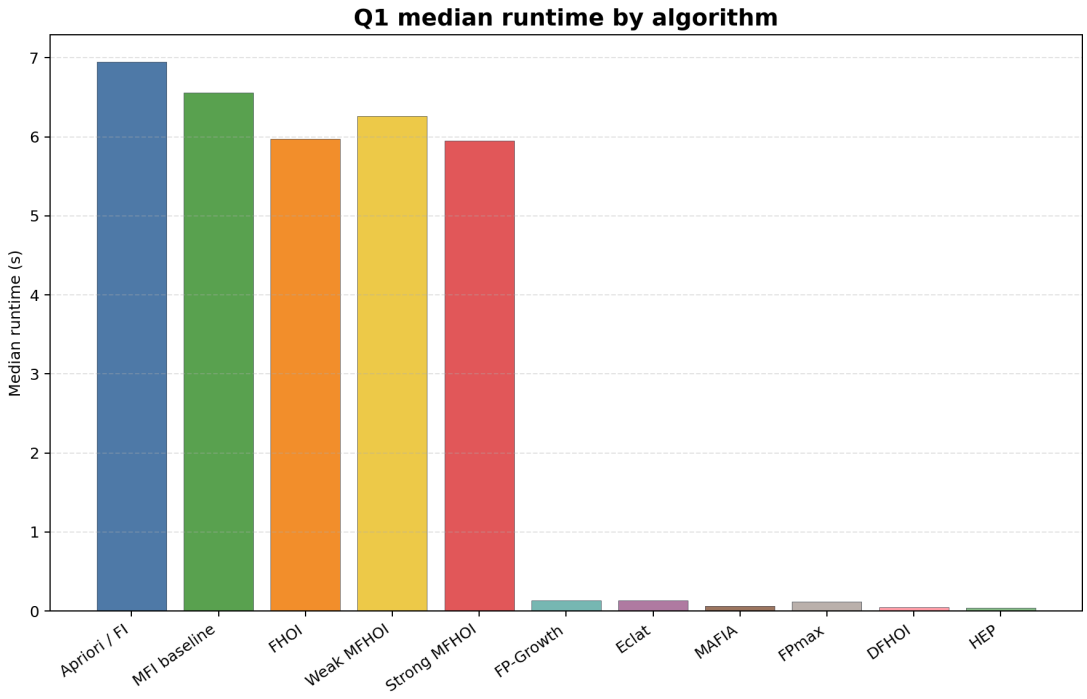



Figure 1: Aggregate median runtime by algorithm over all datasets and threshold configurations.

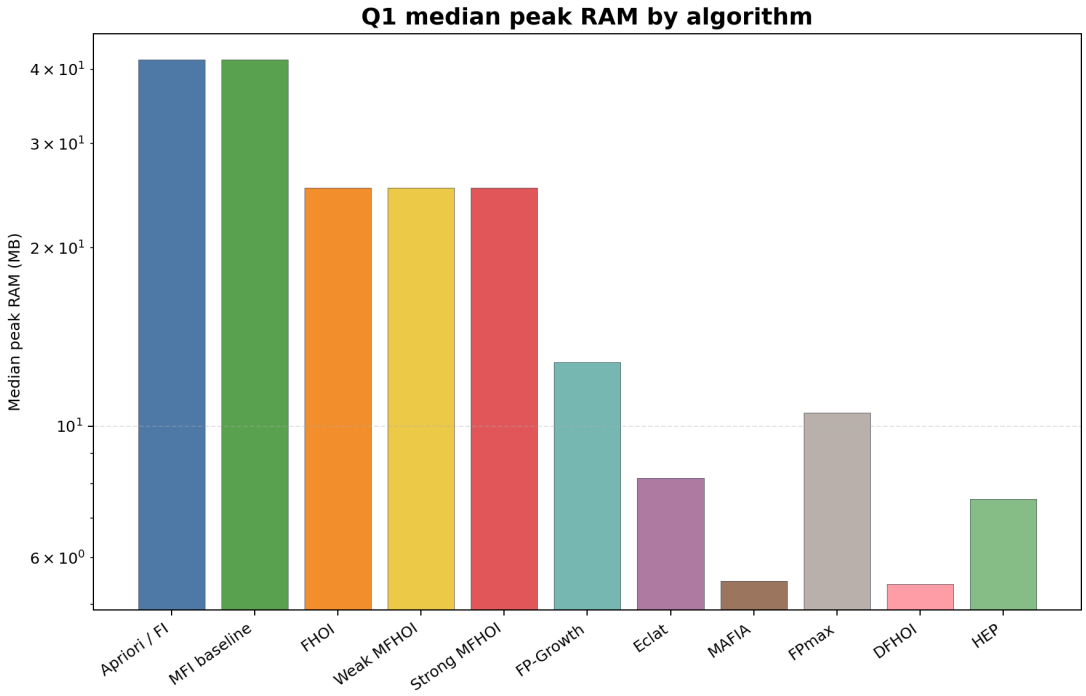



Figure 2: Aggregate median peak RAM usage by algorithm over all datasets and threshold configurations.

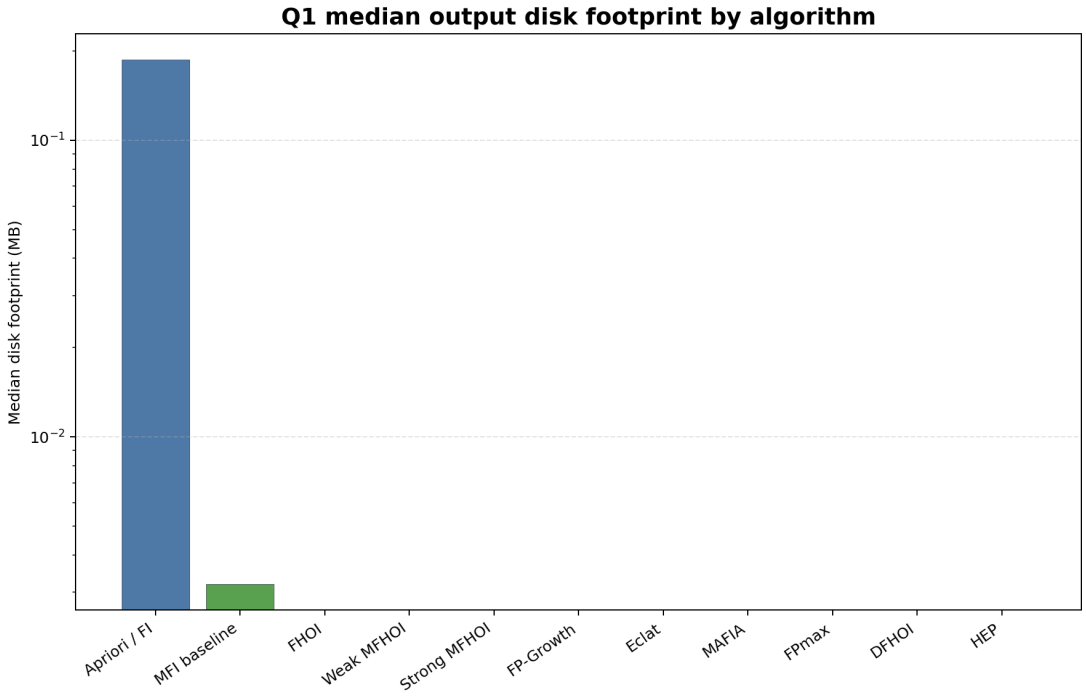



Figure 3: Aggregate median output disk usage by algorithm over all datasets and threshold configurations.

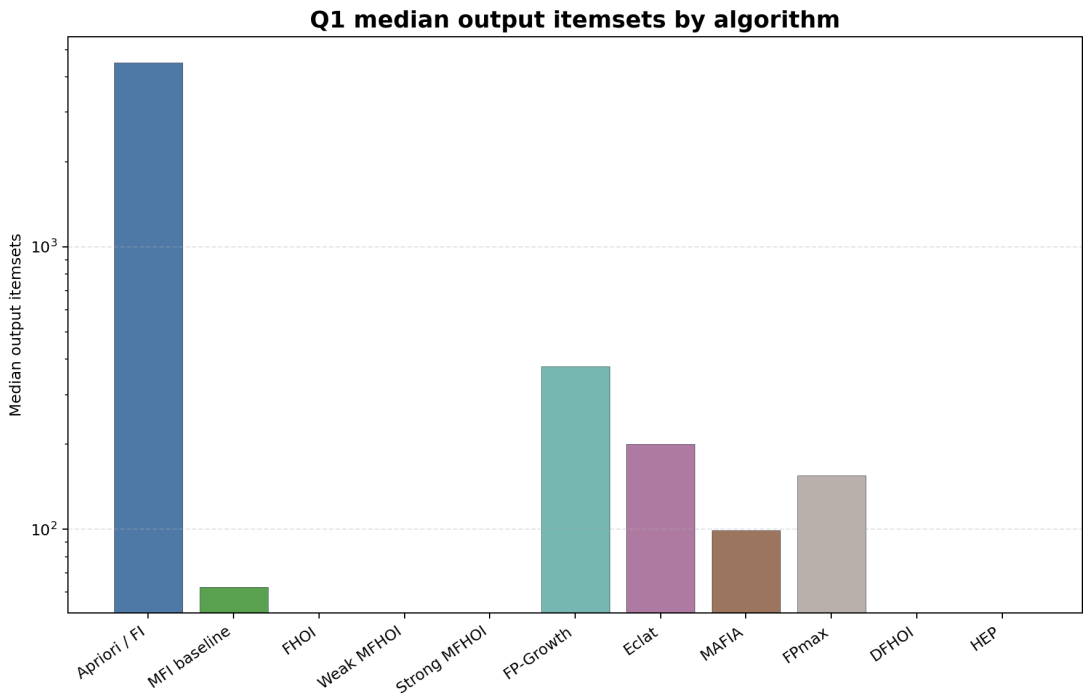



Figure 4: Aggregate median number of output itemsets by algorithm. This figure highlights the compactness of MFHOI-based methods compared with ordinary frequent and high-occupancy itemset mining methods.

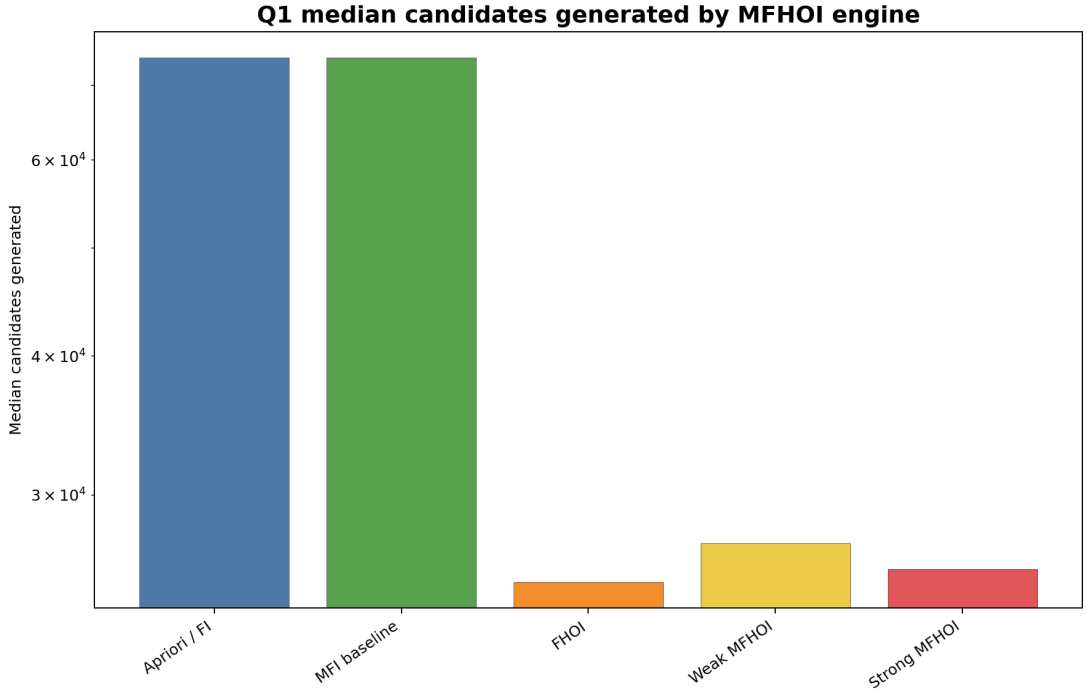



Figure 5: Aggregate median number of generated candidates by algorithm. Candidate generation indicates search-space size and pruning effectiveness.

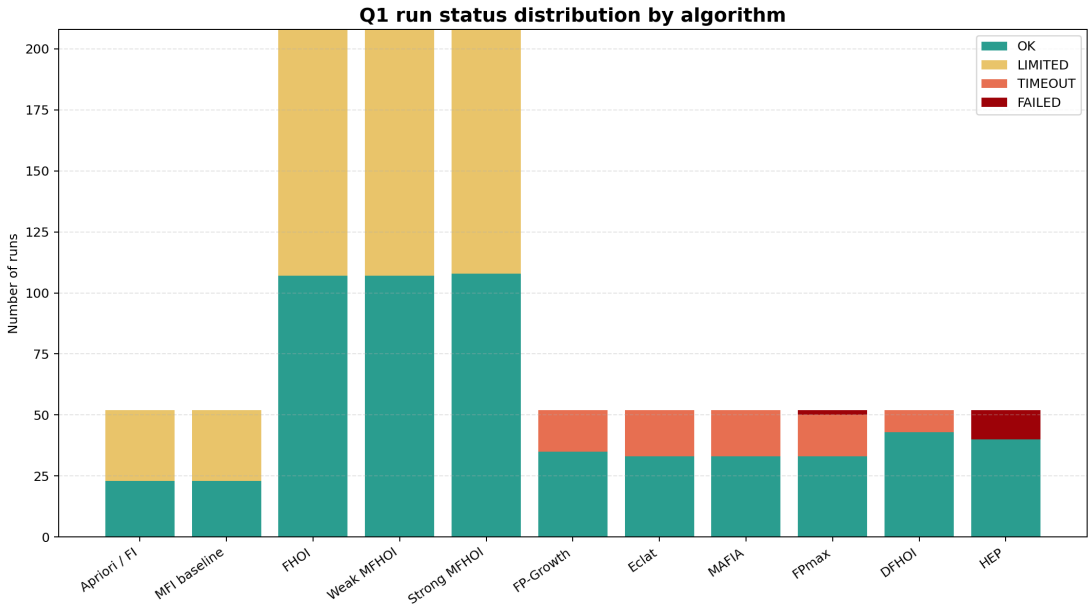



Figure 6: Execution status distribution by algorithm, including successful, limited, timeout, and failed runs.

## 11.2 MFHOI-Family Ablation Results

The MFHOI-family figures compare FHO-Miner, Weak MFHOI-Miner, and Strong MFHOI-Miner. This comparison is the key ablation study because it directly measures the effect of the proposed occupancy-dominance filtering.

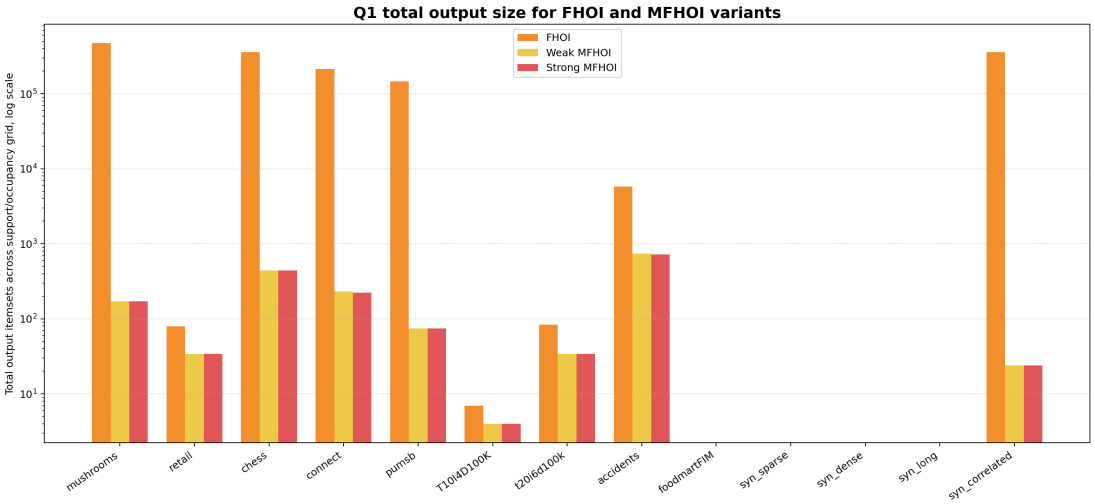



Figure 7: Total number of itemsets produced by FHO-Miner, Weak MFHOI-Miner, and Strong MFHOI-Miner on each dataset.

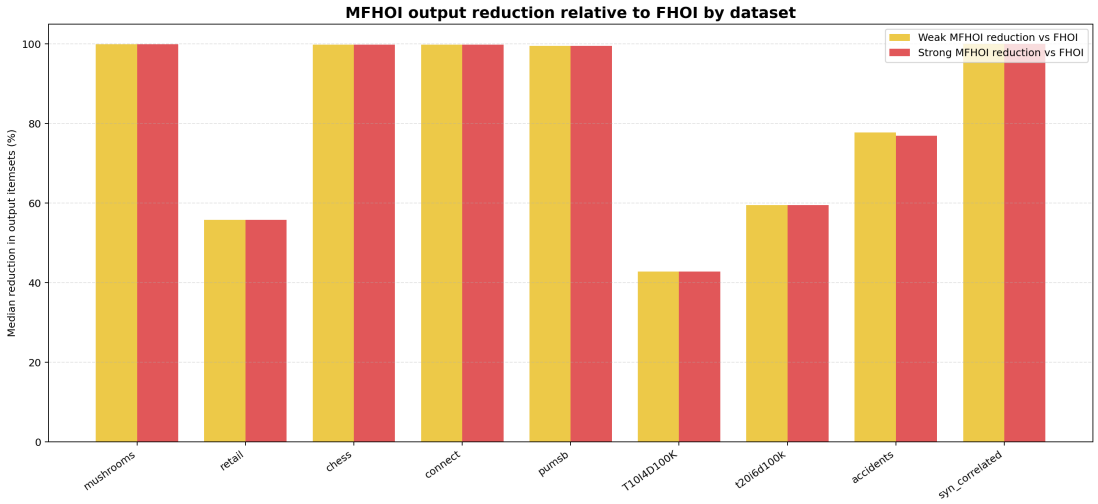



Figure 8: Reduction ratio of MFHOI variants compared with FHO-Miner. Higher values indicate stronger output compression.

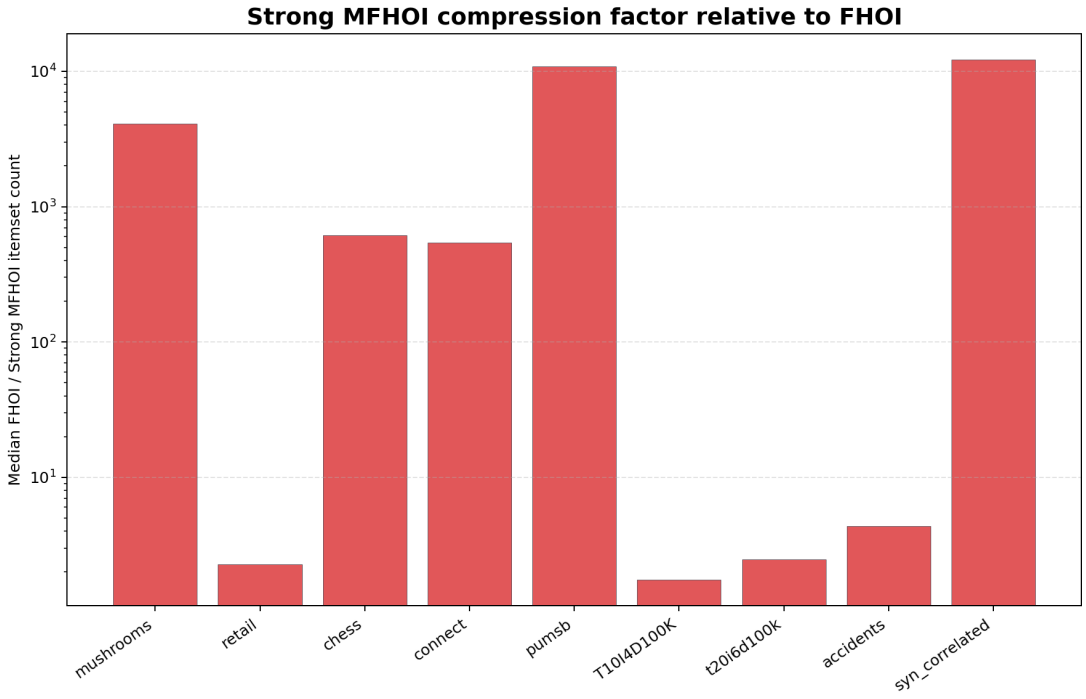



Figure 9: Compression factor achieved by Strong MFHOI-Miner relative to FHO-Miner on each dataset.

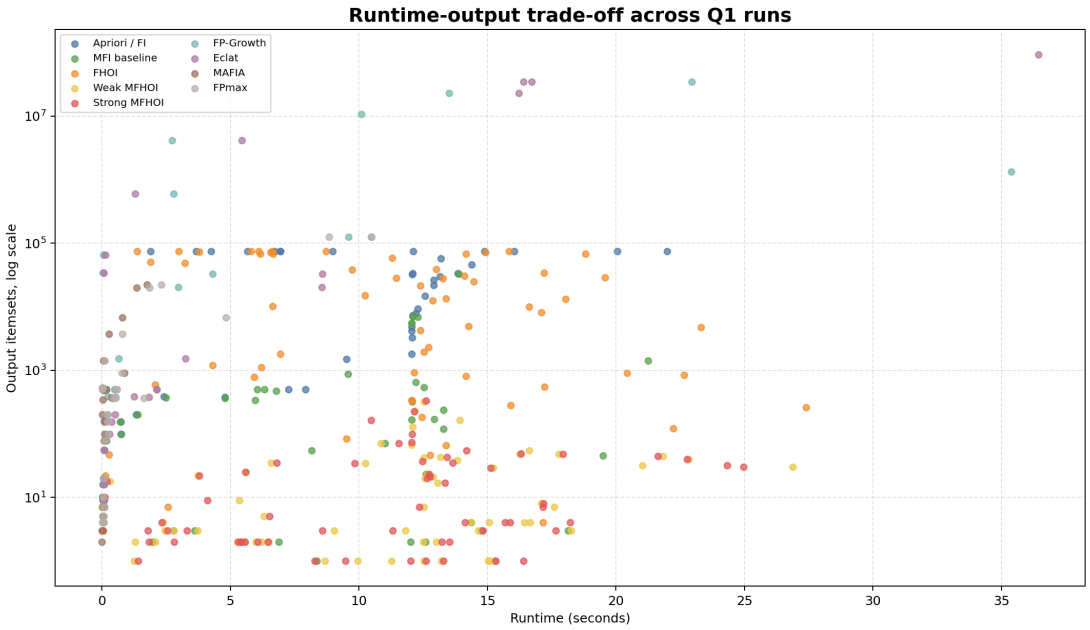
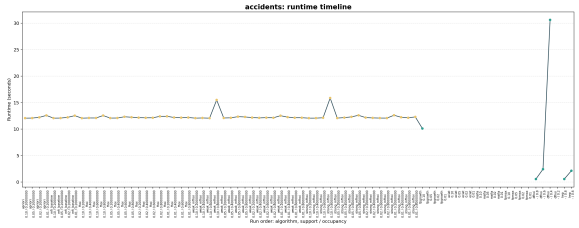


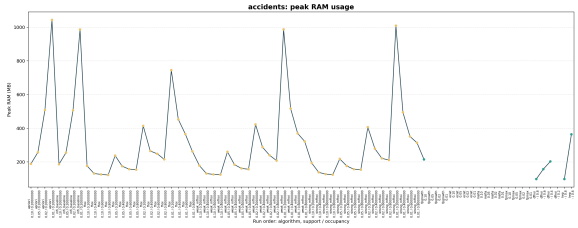

Figure 10: Runtime-output trade-off of all algorithms. Algorithms closer to the lower-left region require less runtime and produce fewer output itemsets.

## 11.3 Per-Dataset Results

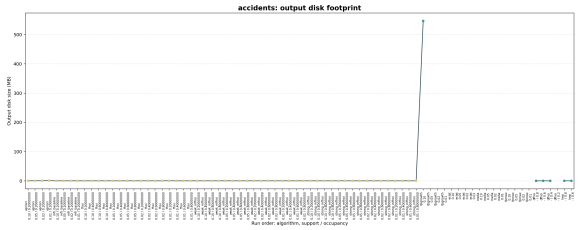
For each dataset, five figures are shown: runtime, RAM usage, disk usage, number of output itemsets, and number of generated candidates. These figures visualize how each algorithm behaves under different support and occupancy thresholds.



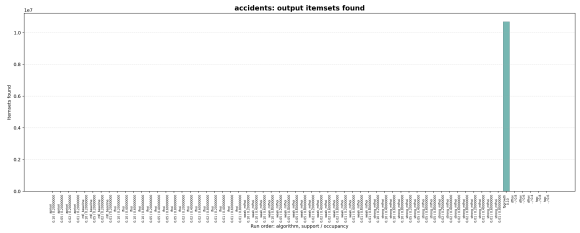

(a) Runtime



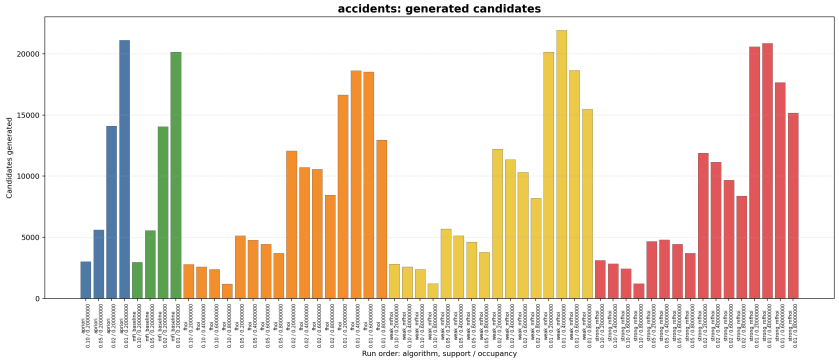

(b) Peak RAM




(c) Disk usage

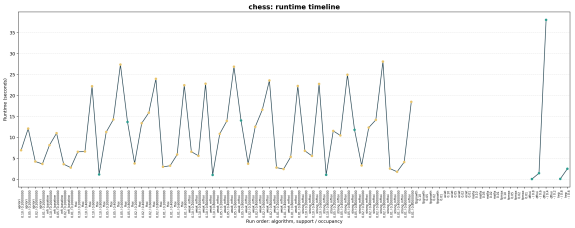



(d) Output itemsets

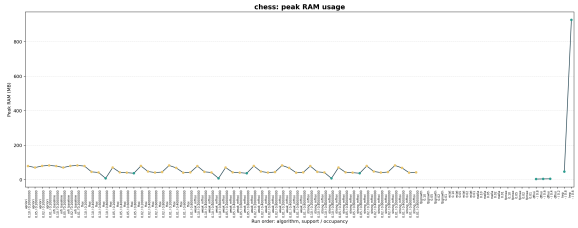



(e) Generated candidates

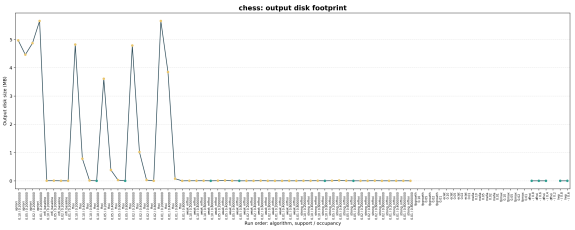
Figure 11: Per-dataset visualization results on the `Accidents` dataset.



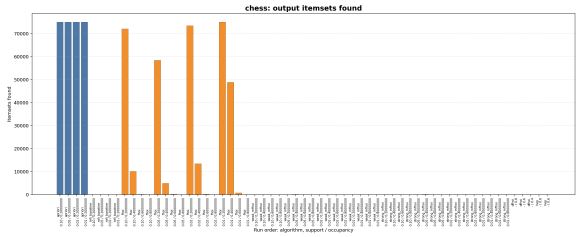

(a) Runtime



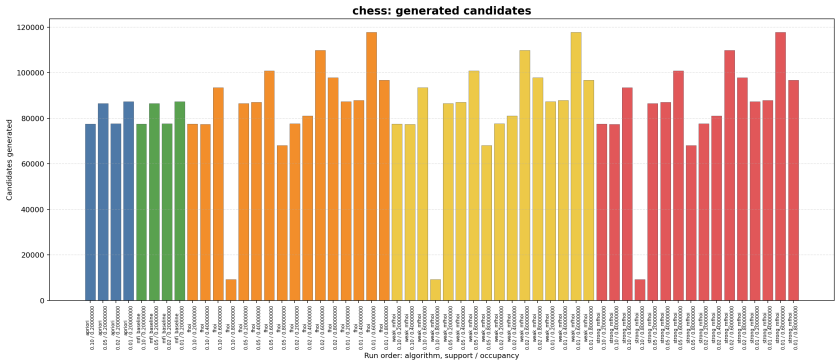

(b) Peak RAM




(c) Disk usage

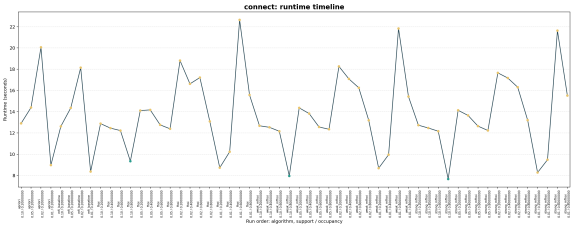



(d) Output itemsets

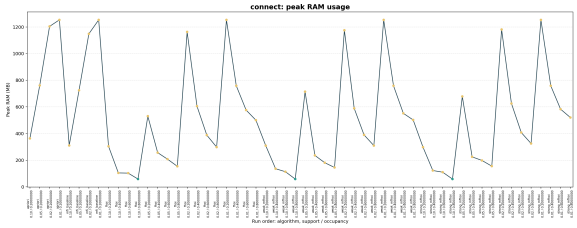



(e) Generated candidates

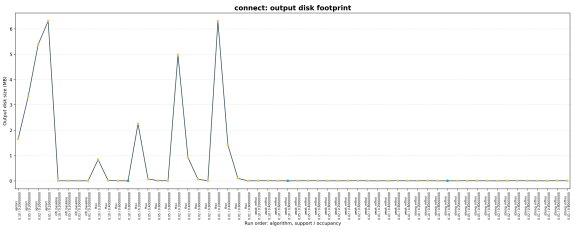
Figure 12: Per-dataset visualization results on the `Chess` dataset.



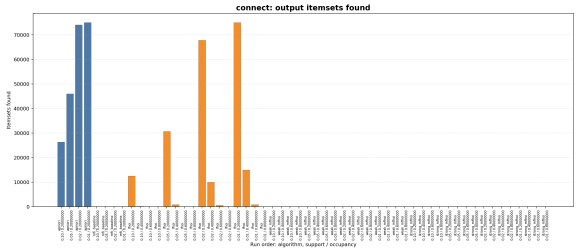

(a) Runtime



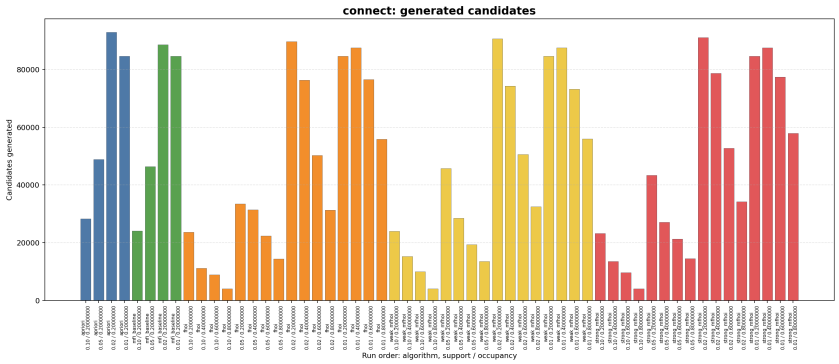

(b) Peak RAM




(c) Disk usage

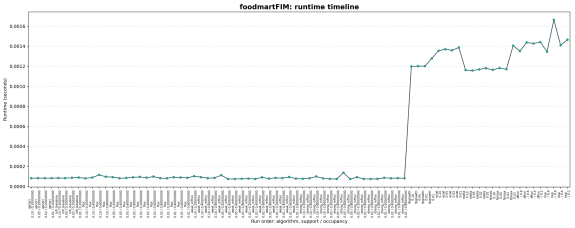



(d) Output itemsets

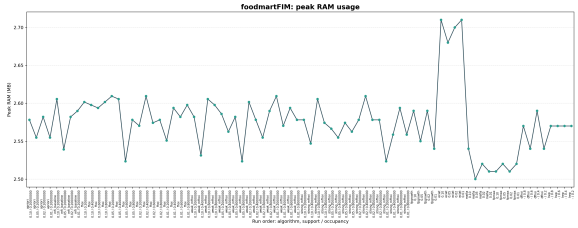



(e) Generated candidates

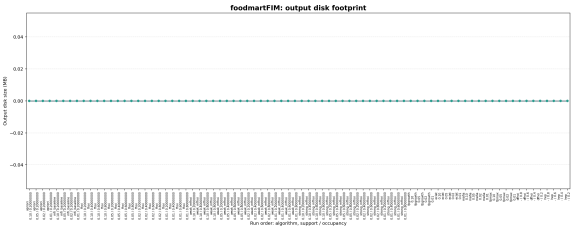
Figure 13: Per-dataset visualization results on the `Connect` dataset.



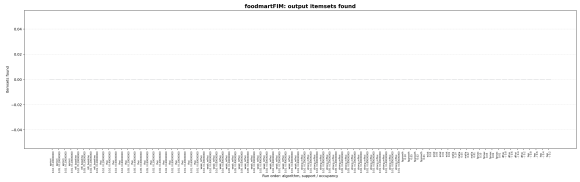

(a) Runtime



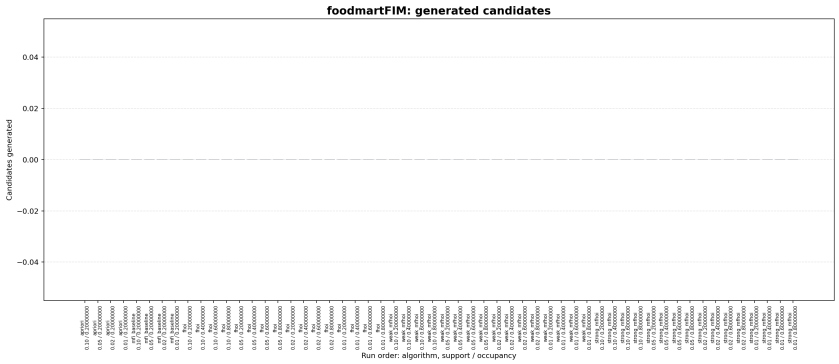

(b) Peak RAM




(c) Disk usage

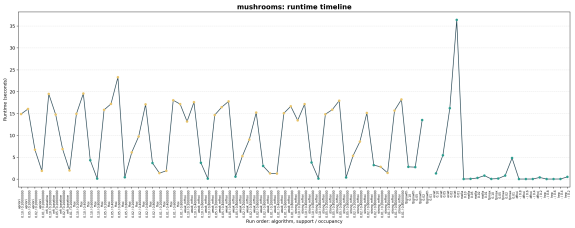



(d) Output itemsets

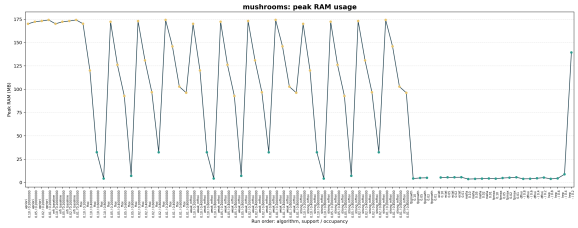



(e) Generated candidates

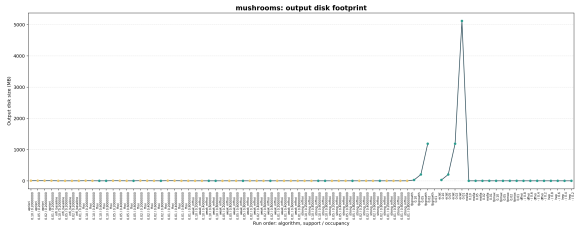
Figure 14: Per-dataset visualization results on the `FoodmartFIM` dataset.



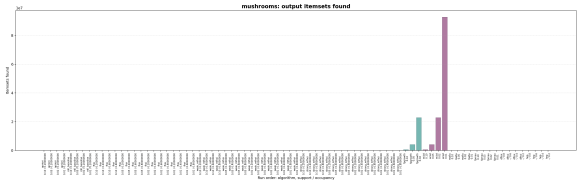

(a) Runtime



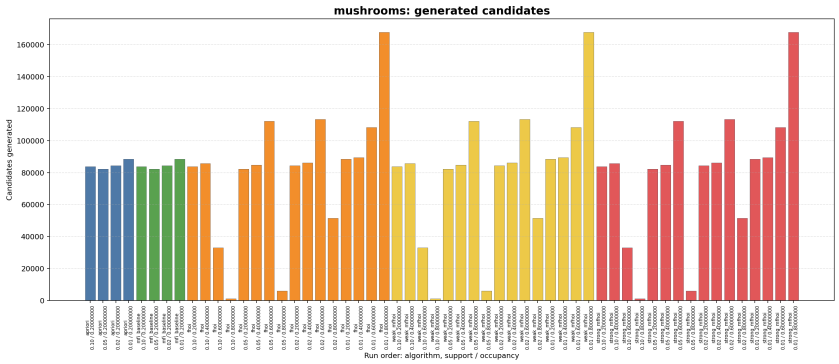

(b) Peak RAM




(c) Disk usage

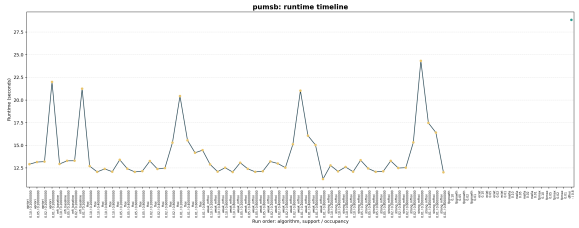



(d) Output itemsets

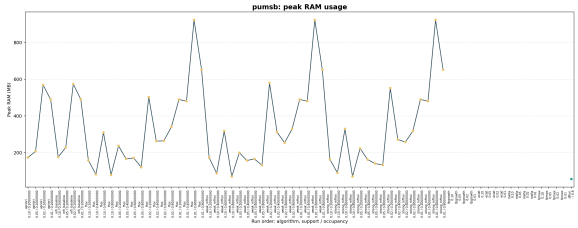



(e) Generated candidates

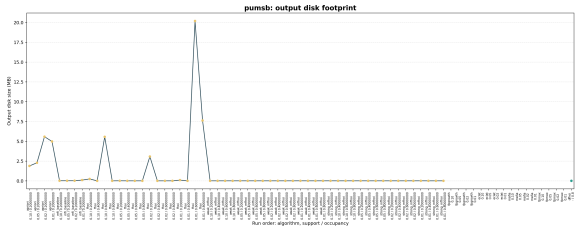
Figure 15: Per-dataset visualization results on the `Mushrooms` dataset.



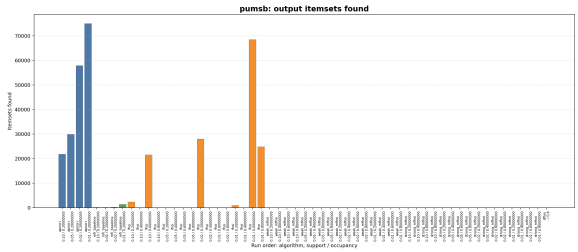

(a) Runtime



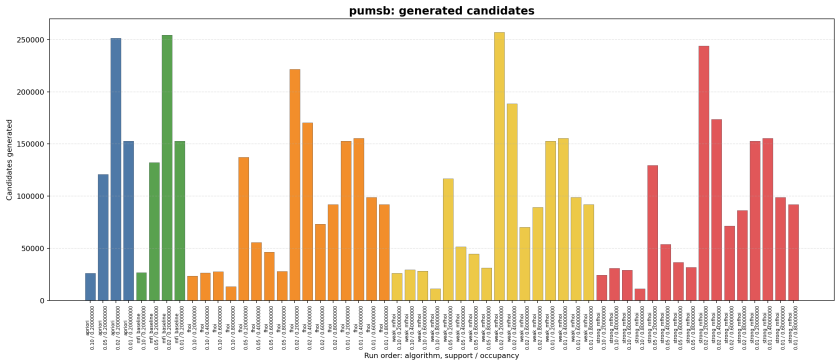

(b) Peak RAM




(c) Disk usage

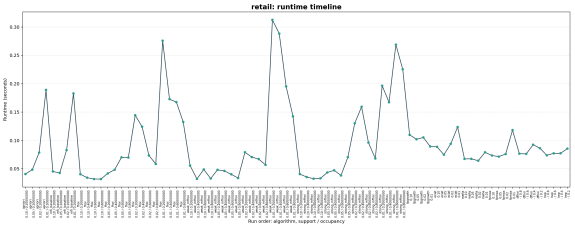



(d) Output itemsets

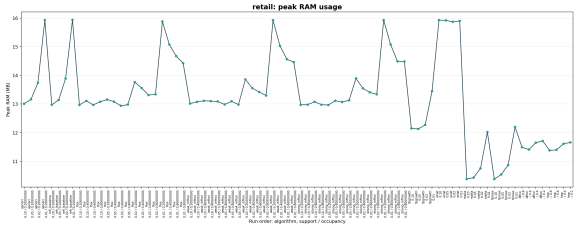



(e) Generated candidates

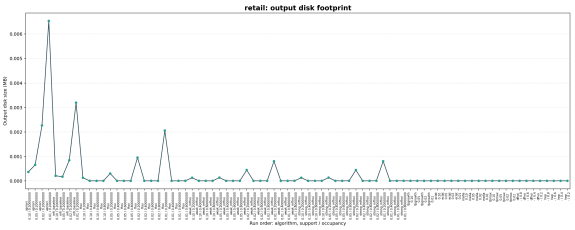
Figure 16: Per-dataset visualization results on the PUMSB dataset.



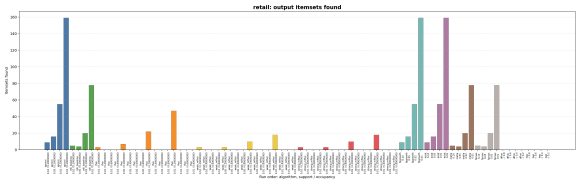

(a) Runtime



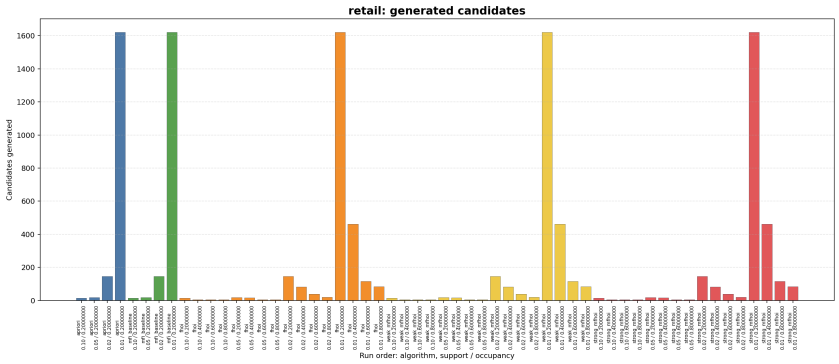

(b) Peak RAM




(c) Disk usage

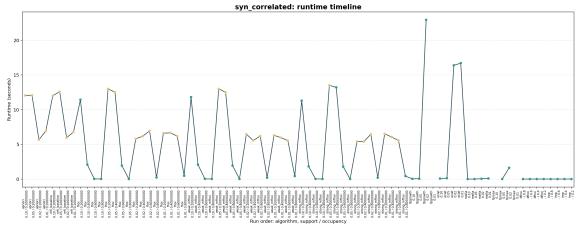



(d) Output itemsets

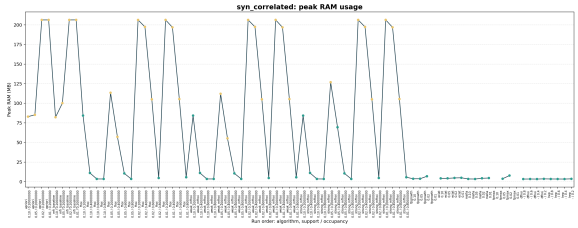



(e) Generated candidates

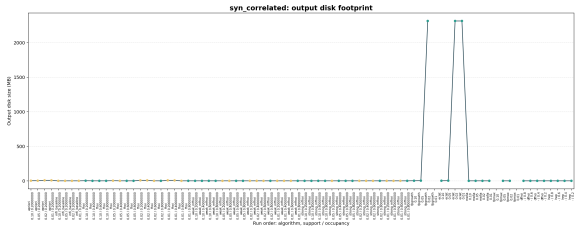
Figure 17: Per-dataset visualization results on the `Retail` dataset.



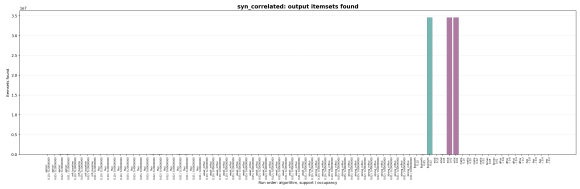

(a) Runtime



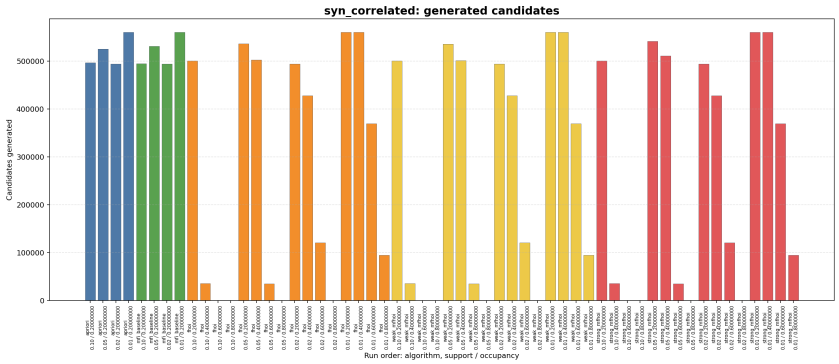

(b) Peak RAM




(c) Disk usage

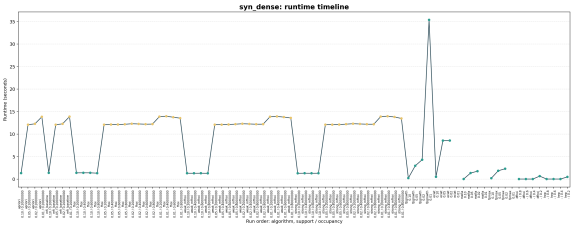



(d) Output itemsets

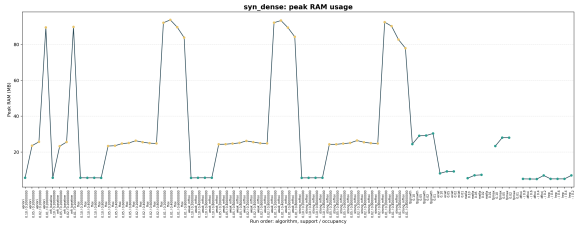



(e) Generated candidates

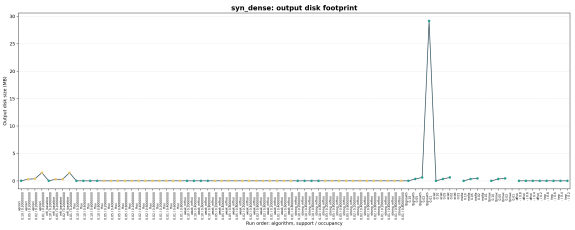
Figure 18: Per-dataset visualization results on the SYN-CORRELATED dataset.



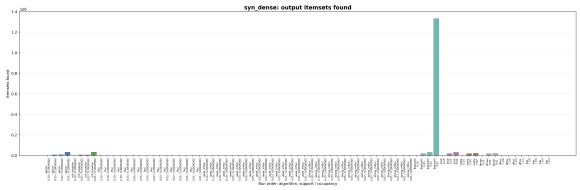

(a) Runtime



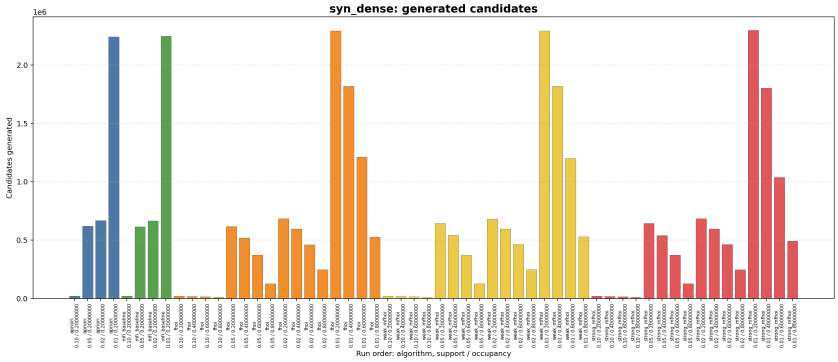

(b) Peak RAM




(c) Disk usage

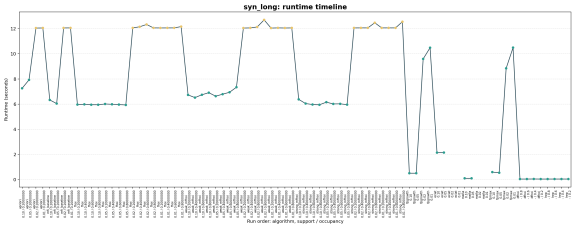



(d) Output itemsets

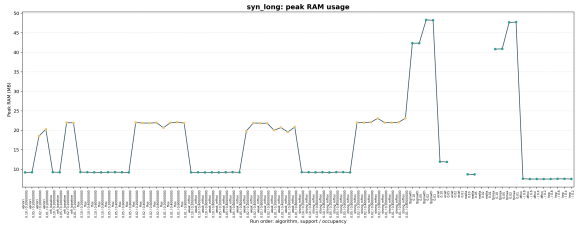



(e) Generated candidates

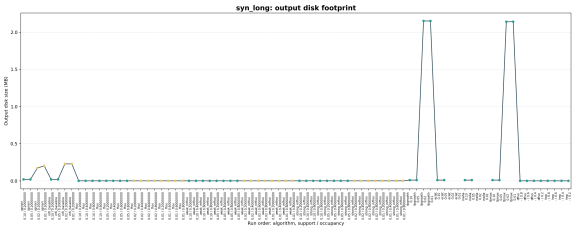
Figure 19: Per-dataset visualization results on the SYN−DENSE dataset.



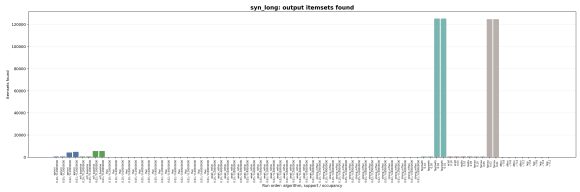

(a) Runtime



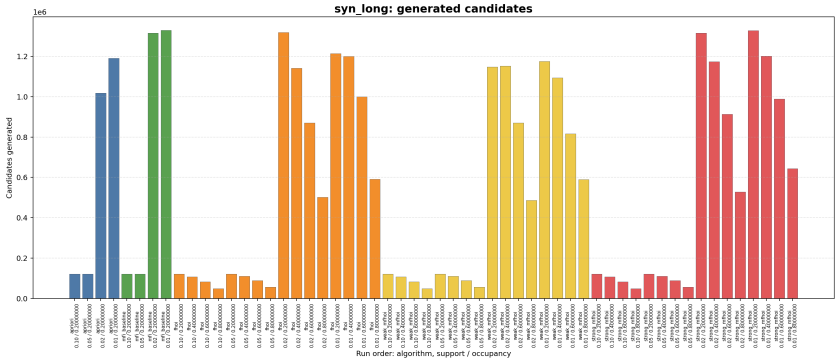

(b) Peak RAM




(c) Disk usage

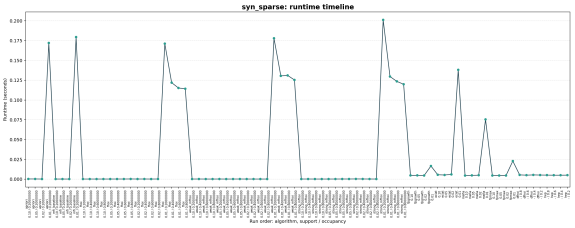



(d) Output itemsets

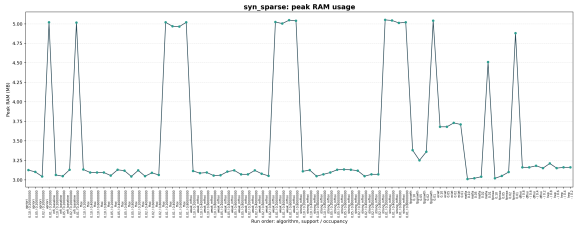



(e) Generated candidates

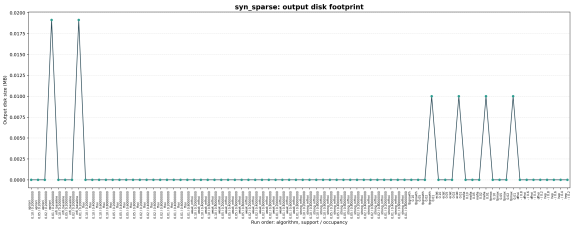
Figure 20: Per-dataset visualization results on the SYN−LONG dataset.



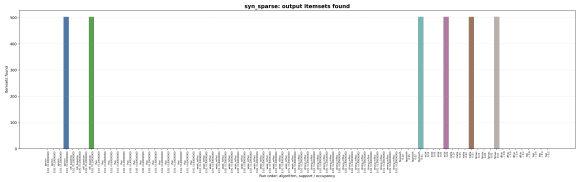

(a) Runtime



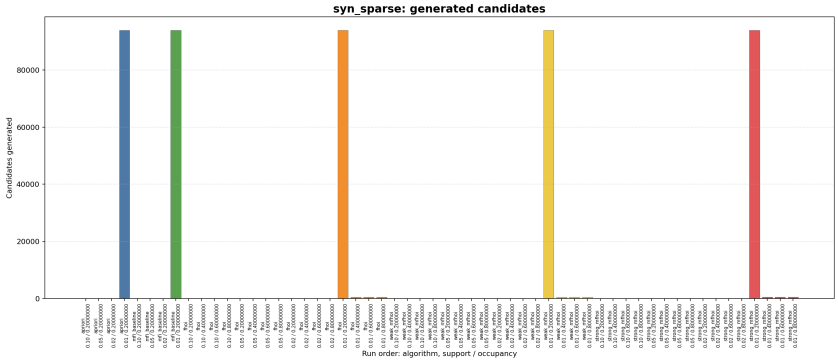

(b) Peak RAM




(c) Disk usage

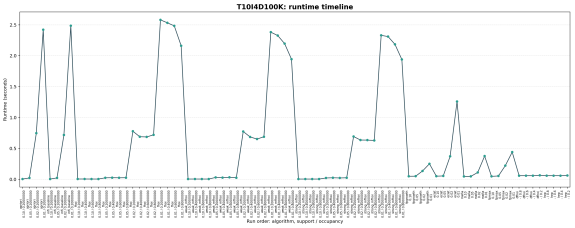



(d) Output itemsets

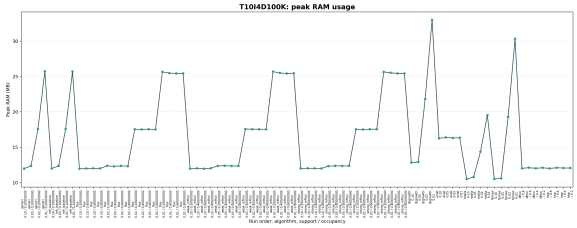



(e) Generated candidates

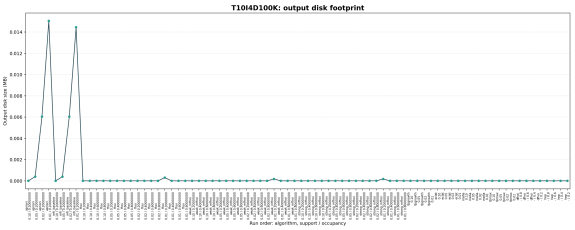
Figure 21: Per-dataset visualization results on the SYN-SPARSE dataset.



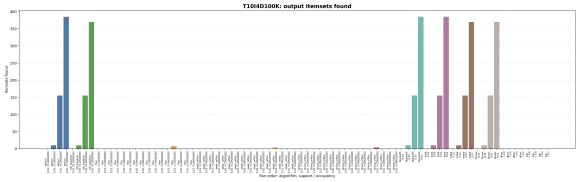

(a) Runtime



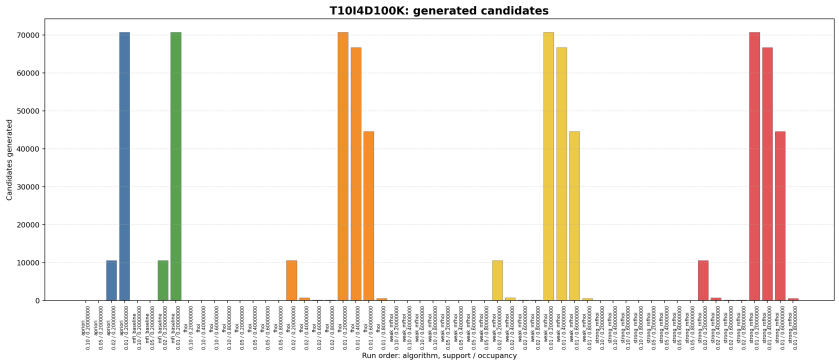

(b) Peak RAM




(c) Disk usage

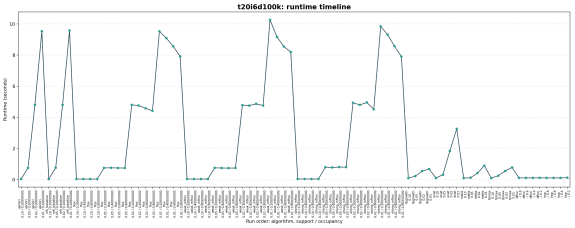



(d) Output itemsets

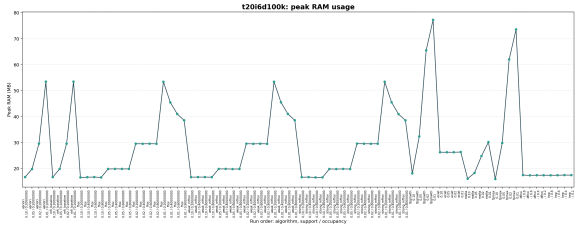



(e) Generated candidates

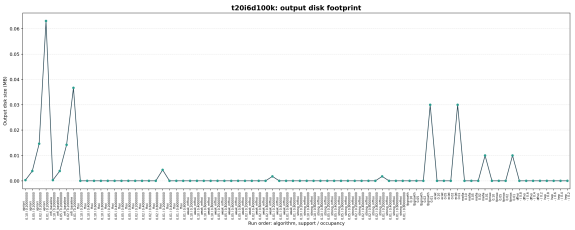
Figure 22: Per-dataset visualization results on the `T10I4D100K` dataset.



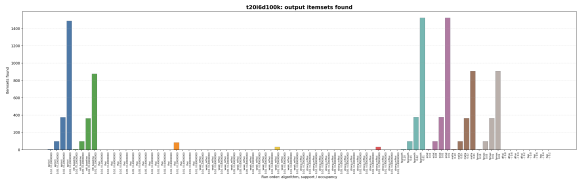

(a) Runtime



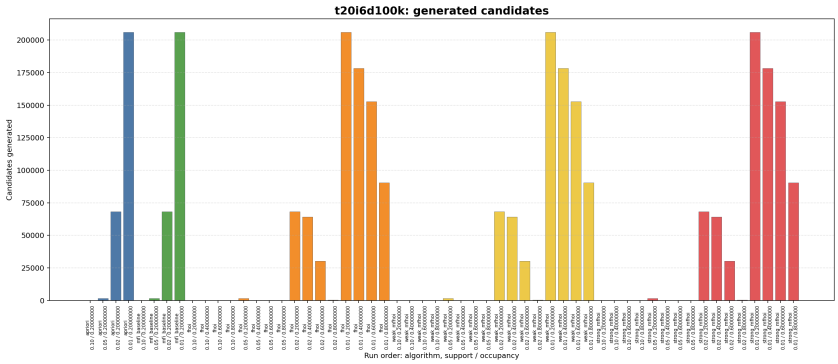

(b) Peak RAM




(c) Disk usage




(d) Output itemsets




(e) Generated candidates

Figure 23: Per-dataset visualization results on the `T20I6D100K` dataset.

### 11.4 Summary of Visualization Findings

The visualization results show that MFHOI-Miner behaves as a concise representation method for frequent high-occupancy patterns. Across dense and long-transaction datasets, FHO-Miner often reports many frequent high-occupancy itemsets, while Weak MFHOI-Miner and Strong MFHOI-Miner retain only the non-dominated representative patterns.

The runtime and RAM figures show that MFHOI-Miner introduces an additional dominance-checking cost compared with FHO-Miner. However, this overhead is acceptable because the final output size is substantially reduced. The disk-usage figures further confirm that MFHOI-Miner produces much smaller output files than methods that enumerate all frequent or all frequent high-occupancy itemsets.

Overall, the visualization results support the main claim of this paper: Strong MFHOI-Miner provides a compact, interpretable, and occupancy-aware representation of dense frequent itemsets.

## 12 Conclusion

This paper introduced MFHOI-Miner, an efficient method for mining Maximal Frequent High-Occupancy Itemsets. MFHOIs combine three requirements: frequency, high average occupancy, and dominance-based maximality. We formalized weak and strong MFHOI variants, proved their theoretical properties, showed that average occupancy is not anti-monotone, and proposed safe occupancy upper bounds for

pruning. MFHOI-Miner uses a vertical bitset-based depth-first strategy to mine frequent high-occupancy itemsets and then applies exact dominance filtering to obtain weak or strong MFHOIs.

The proposed framework provides a concise, occupancy-aware representation of dense frequent patterns. It is especially useful when ordinary frequent itemset mining produces too many patterns and when standard maximal frequent itemsets ignore occupancy quality. Future work will focus on tighter upper bounds, faster dominance filtering, direct online MFHOI mining, and extensive empirical evaluation on real and synthetic benchmark datasets.

# References


[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[2] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.

[3] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[4] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 443–452, 2001.

[5] Doug Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. MAFIA: A maximal frequent itemset algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1490–1504, 2005.

[6] Karam Gouda and Mohammed J. Zaki. GenMax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery*, 11(3):223–242, 2005.

[7] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using FP-trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1347–1362, 2005.

[8] Linpeng Tang, Lei Zhang, Ping Luo, and Min Wang. Incorporating occupancy into frequent pattern mining for high quality pattern recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 75–84, 2012.

[9] Lei Zhang, Ping Luo, Linpeng Tang, Enhong Chen, Qi Liu, Min Wang, and Hui Xiong. Occupancy-based frequent pattern mining. *ACM Transactions on Knowledge Discovery from Data*, 10(2):1–33, 2015.

[10] Zhi-Hong Deng. Mining high occupancy itemsets. *Future Generation Computer Systems*, 102:222–229, 2020.

[11] Subrata Datta, Kalyani Mali, and Udit Ghosh. High occupancy itemset mining with consideration of transaction occupancy. *Arabian Journal for Science and Engineering*, 47:2061–2075, 2022.

[12] Loan T. T. Nguyen, Thang Mai, Giao-Huy Pham, Unil Yun, and Bay Vo. An efficient method for mining high occupancy itemsets based on equivalence class and early pruning. *Knowledge-Based Systems*, 267:110441, 2023.

[13] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, and Philip S. Yu. HUOPM: High-utility occupancy pattern mining. *IEEE Transactions on Cybernetics*, 50(3):1195–1208, 2020.

[14] Hai Duong, Huy Pham, Tin Truong, and Philippe Fournier-Viger. Efficient algorithms to mine concise representations of frequent high utility occupancy patterns. *Applied Intelligence*, 54(5):4012–4042, 2024.

[15] Bart Goethals. Frequent Itemset Mining Implementations Repository. 2003.

[16] UCI Machine Learning Repository. Mushroom dataset. 1981.