

Mining maximal high occupancy itemsets with a size-coupled upper bound

Quan Van

vanhaminhquan2406@gmail.com

Independent Researcher <https://orcid.org/0009-0002-8222-0076>

Research Article

Keywords: Pattern mining, Occupancy, High occupancy itemsets, Maximal itemsets, Closed itemsets, Upper bounds

Posted Date: October 20th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11385702/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Mining maximal high occupancy itemsets with a size-coupled upper bound

Quan Van^{1*}

^{1*}Independent Researcher, Ho Chi Minh City, Vietnam.

Corresponding author(s). E-mail(s): vanhaminhquan2406@gmail.com;

Abstract

Occupancy, as defined by Deng, measures how much of its supporting transactions an itemset covers: it is the sum, over these transactions, of the fraction of each transaction that the itemset contains. An itemset is a high occupancy itemset (HOI) if its occupancy reaches a threshold ξ . Since the HOIs can be numerous, we study the maximal HOIs (MHOIs), the HOIs with no HOI as a proper superset. We prove that every HOI has support at least $\lceil \xi \rceil$ and every MHOI is closed; that a maximal frequent itemset that is an HOI is an MHOI, whereas an MHOI need not be one; and we observe that, when all transactions have the same length, the upper bound used in existing HOI mining equals the support. We propose an upper bound on the occupancy of the descendants of a search node that fixes the number of added items across the supporting transactions, and prove that it never exceeds the existing bound. The algorithm MHOI-Miner searches the set-enumeration tree of the frequent itemsets and prunes with this bound, head-union-tail look-ahead and subsumption. Its output agrees with an exact oracle on 8,100 randomized cases. On five of the six datasets used by Deng, with his thresholds, it returns the same MHOIs as the two published HOI miners followed by maximality filtering, evaluates a median of 60.6 and 125 times fewer itemsets, is faster in most settings and completes all of them; on one sparse dataset it is slower.

Keywords: Pattern mining, Occupancy, High occupancy itemsets, Maximal itemsets, Closed itemsets, Upper bounds

1 Introduction

A frequent itemset is a set of items that appears together in at least a given number of transactions [1]. Support does not say how much of a transaction an itemset accounts for: a pair of products that is frequent because it appears in many large orders describes a small part of each of them. Tang et al. introduced occupancy to capture this aspect: the occupancy of an itemset in a transaction is the fraction of the transaction that the itemset covers [2, 3]. They combined a support threshold with a threshold on the average occupancy. Deng observed that an appropriate pair of thresholds is hard to choose, and defined the occupancy of an itemset as the *sum* of these fractions over its supporting transactions, which grows both with support and with the fraction covered and is used with a single threshold [4]. An itemset whose occupancy reaches the threshold is a high occupancy itemset (HOI). Deng proposed the level-wise algorithm HEP to mine all HOIs, and Nguyen et al. the depth-first algorithm DFHOI [5]; later work studied variants of the problem [6–11].

The number of HOIs can be large. Deng reports, for example, 331,704 HOIs on the `mushroom` dataset at a threshold of 4% of the number of transactions [4, Table 3]. As for frequent itemsets [12], the maximal elements give a smaller answer: a maximal HOI (MHOI) is an HOI with no HOI as a proper superset, and every HOI is a subset of an MHOI. Unlike maximal frequent itemsets, MHOIs do not determine all HOIs, because occupancy is not anti-monotone: a subset of an MHOI need not be an HOI. MHOIs report the largest itemsets that reach the threshold; they are not a lossless representation of the HOIs.

This paper is about computing exactly the set of MHOIs. Two approaches follow directly from existing algorithms. The first mines all HOIs with an existing algorithm [4, 5] and keeps the maximal ones; its cost grows with the number of itemsets that these algorithms enumerate. The second mines the maximal frequent itemsets and keeps those that are HOIs; we show that it is incorrect. The contributions are as follows.

1. We relate MHOIs to frequent, closed and maximal frequent itemsets (Section 4). Every HOI is frequent at the support threshold $\lceil \xi \rceil$, where ξ is the occupancy threshold (this follows from a result of Deng), and every MHOI is closed. Every maximal frequent itemset that is an HOI is an MHOI, but an MHOI need not be a maximal frequent itemset, also when all transactions have the same length; we give examples. When all transactions have the same length L , an itemset X is an HOI if and only if its support is at least $\xi L / |X|$, a length-decreasing support constraint in the sense of Seno and Karypis [13].
2. We propose a bound on the occupancy of the descendants of a search node that fixes the number k of added items and sums only over the transactions that contain at least k items of the tail (Theorem 6). We prove that it never exceeds the bound UBO of Deng [4], which is used by the existing HOI miners (Theorem 7), nor a per-transaction bound of the kind used in high-utility mining (Theorem 8). We also note that, when all transactions have the same length, UBO equals the support and prunes nothing beyond it (Proposition 5).

3. We present MHOI-Miner, a depth-first search of the set-enumeration tree of the itemsets with support at least $\lceil \xi \rceil$ that prunes with the bound, with head-union-tail look-ahead and with subsumption by recorded HOIs, and we prove that it returns exactly the MHOIs (Theorem 12).
4. We compare MHOI-Miner with the two published HOI miners [4, 5], each followed by maximality filtering, on the datasets and threshold ranges used by Deng [4], after verifying all implementations against an exact oracle (Section 7). A case study on retail data compares MHOIs with maximal frequent itemsets (Section 8).

Section 2 reviews related work and Section 3 defines the problem; a small example database is used throughout to illustrate the definitions, the bounds and a complete run of the algorithm. Section 4 proves the structural properties, Section 5 develops the bound and Section 6 the algorithm. Section 7 reports the experiments, Section 8 the case study, Section 9 the limitations, and Section 10 concludes.

2 Related work

Frequent, closed and maximal itemsets.

Apriori [14], FP-growth [15] and Eclat [16] enumerate frequent itemsets using the anti-monotonicity of support. Closed itemsets, which have no proper superset with the same support, form a lossless condensed representation [17]; CHARM [18] mines them with tidset intersections and LCM [19, 20] with prefix-preserving closure extension, which visits every frequent closed itemset exactly once without storing the itemsets found so far. Maximal frequent itemsets (MFIs) have no frequent proper superset; Max-Miner [12], MAFIA [21], GenMax [22] and FPMMax* [23] mine them with head-union-tail look-ahead and superset checking. Seno and Karypis studied frequent itemsets under a support threshold that decreases with the length of the itemset, and pruned the search with the smallest length that an extension must reach to satisfy the constraint [13]; on databases whose transactions have equal length, HOIs are itemsets of this kind (Theorem 4(c)), and the bound of Section 5 expresses a related length argument through the capacities of the transactions. Constraints that are neither anti-monotone nor monotone have been pushed into frequent itemset mining when they are convertible [24], and combined with closed itemset mining [25]; that framework assumes constraints defined by values attached to items, whereas the occupancy of an itemset depends on its supporting transactions. Surveys are available in [26, 27].

Occupancy.

Tang et al. [2] and Zhang et al. [3] defined the average occupancy of an itemset and mined the itemsets that satisfy both a support threshold and an average-occupancy threshold. Deng [4] defined occupancy as the sum of the per-transaction fractions, introduced HOI mining with a single threshold, proved that the occupancy of an itemset is at most its support, and proposed HEP. HEP stores, for each itemset, the list of its supporting transactions with their lengths, generates candidates level by level, and prunes with an upper bound UBO on the occupancy of all supersets of an itemset (Definition 3). DFHOI [5] uses the same bound in a depth-first search over equivalence classes. Later work considered transaction occupancy [6], incremental

databases [7], an adaptive threshold [8], top- k mining [9, 10] and data streams [11]. These algorithms report all HOIs; we know of no algorithm that mines maximal HOIs.

Utility occupancy.

Utility occupancy replaces the number of items by their utility [28, 29]. Like the occupancy of Tang et al., it is averaged over the supporting transactions and used together with a support threshold. Closed and maximal frequent high-utility occupancy itemsets were studied by Duong et al. [30] and Hoang et al. [31]. Their problem has two thresholds and an averaged measure; it does not contain the problem of this paper, whose measure is a sum used with one threshold.

Bounds evaluated before projection.

Upper bounds on what can still be added in a depth-first enumeration are standard in high-utility itemset mining [32–34]; EFIM [33] computes bounds for the children of a node before projecting their databases. Corollary 9 applies the same idea to the bound of Theorem 6.

3 Problem definition

Let I be a finite set of items and $D = \langle T_1, \dots, T_n \rangle$ a database of non-empty transactions $T_q \subseteq I$; $|T_q|$ denotes the number of items of T_q . For an itemset $X \subseteq I$, $\Gamma(X) = \{q : X \subseteq T_q\}$ is its cover and $\text{supp}(X) = |\Gamma(X)|$ its support.

Definition 1 (Occupancy [4]) The occupancy of a non-empty itemset X is

$$\text{occ}(X) = \sum_{q \in \Gamma(X)} \frac{|X|}{|T_q|} = |X| \sum_{q \in \Gamma(X)} \frac{1}{|T_q|}. \quad (1)$$

Definition 2 (HOI and MHOI) Let $\xi > 0$ be an occupancy threshold. A non-empty itemset X is a high occupancy itemset (HOI) if $\text{occ}(X) \geq \xi$; $\mathcal{H}$ denotes the set of HOIs. An HOI is maximal (an MHOI) if no proper superset of it is an HOI; $\mathcal{M} = \text{Max}(\mathcal{H})$ denotes the set of MHOIs, where $\text{Max}(\mathcal{S})$ is the set of elements of $\mathcal{S}$ with no proper superset in $\mathcal{S}$. The problem is to compute $\mathcal{M}$.

As in [4], a threshold can be given relative to the number of transactions: a relative threshold r corresponds to $\xi = rn$. Since $\mathcal{H}$ is finite, every HOI is a subset of an MHOI. For a support threshold σ , $\mathcal{F}_\sigma$ denotes the set of non-empty itemsets with $\text{supp}(X) \geq \sigma$ and $\text{MFI}_\sigma = \text{Max}(\mathcal{F}_\sigma)$.

Throughout the paper we illustrate the definitions, the bounds and the algorithm on the database of Table 1, with $\xi = 2$.

Example 1 In Table 1, $\Gamma(\{a, f\}) = \{3, 4, 5, 6\}$, so

$$\text{occ}(\{a, f\}) = \frac{2}{4} + \frac{2}{2} + \frac{2}{5} + \frac{2}{6} = \frac{67}{30} \approx 2.233,$$

Table 1 Example database D .

TID	Length	Items
T_1	3	b, d, e
T_2	5	b, c, d, e, f
T_3	4	a, c, d, f
T_4	2	a, f
T_5	5	a, c, d, e, f
T_6	6	a, b, c, d, e, f
T_7	5	a, b, c, d, e
T_8	3	c, e, f

Table 2 The HOIs of the database of Table 1 with $\xi = 2$, and its maximal frequent itemsets at $\sigma = \lceil \xi \rceil = 2$.

Itemset	Cover	Support	occ	MHOI
<i>HOIs</i>				
$\{a, f\}$	3, 4, 5, 6	4	$67/30 \approx 2.233$	
$\{c, d\}$	2, 3, 5, 6, 7	5	$61/30 \approx 2.033$	
$\{c, e\}$	2, 5, 6, 7, 8	5	$11/5 = 2.2$	
$\{c, f\}$	2, 3, 5, 6, 8	5	$23/10 = 2.3$	
$\{d, e\}$	1, 2, 5, 6, 7	5	$11/5 = 2.2$	
$\{a, c, d\}$	3, 5, 6, 7	4	$49/20 = 2.45$	
$\{b, d, e\}$	1, 2, 6, 7	4	$27/10 = 2.7$	
$\{c, d, e\}$	2, 5, 6, 7	4	$23/10 = 2.3$	
$\{c, d, f\}$	2, 3, 5, 6	4	$49/20 = 2.45$	
$\{c, e, f\}$	2, 5, 6, 8	4	$27/10 = 2.7$	
$\{a, c, d, e\}$	5, 6, 7	3	$34/15 \approx 2.267$	yes
$\{a, c, d, f\}$	3, 5, 6	3	$37/15 \approx 2.467$	yes
$\{b, c, d, e\}$	2, 6, 7	3	$34/15 \approx 2.267$	yes
$\{c, d, e, f\}$	2, 5, 6	3	$34/15 \approx 2.267$	yes
<i>Maximal frequent itemsets ($\sigma = 2$)</i>				
$\{a, b, c, d, e\}$	6, 7	2	$11/6 \approx 1.833$	
$\{a, c, d, e, f\}$	5, 6	2	$11/6 \approx 1.833$	
$\{b, c, d, e, f\}$	2, 6	2	$11/6 \approx 1.833$	

and $\text{occ}(\{a\}) = \frac{1}{4} + \frac{1}{2} + \frac{1}{5} + \frac{1}{6} + \frac{1}{5} = \frac{79}{60} \approx 1.317$. With $\xi = 2$, no single item is an HOI, and there are 14 HOIs (Table 2). Four of them have no HOI as a proper superset, so $\mathcal{M} = \{\{a, c, d, e\}, \{a, c, d, f\}, \{b, c, d, e\}, \{c, d, e, f\}\}$. Occupancy is not anti-monotone: $\{c, d, f\}$ is an HOI, with occupancy $49/20 = 2.45$, whereas its subset $\{d, f\}$, with $\Gamma(\{d, f\}) = \{2, 3, 5, 6\}$, has occupancy $2(\frac{1}{5} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6}) = \frac{49}{30} \approx 1.633$ and is not.

A straightforward way to compute $\mathcal{M}$ is to mine all HOIs with an existing algorithm [4, 5] and to discard every HOI that is a subset of another one. In Example 1 this computes the 14 HOIs and discards 10 of them. Its cost is that of enumerating all HOIs, together with all itemsets that the HOI miner examines to find them, and of comparing the HOIs with each other, although most of them are discarded. Section 7.4 shows that on the benchmark datasets the HOIs outnumber the MHOIs by a median factor of 10.2 and by up to 3,134, and that the HOI miners exceed the time or memory limit at thresholds where the MHOIs are few. This motivates a search that is directed at the maximal HOIs and prunes the itemsets that cannot lead to one.

4 Structural properties

Lemma 1 ([4, Theorem 1]) $\text{occ}(X) \leq \text{supp}(X)$ for every non-empty itemset X . Consequently $\mathcal{H} \subseteq \mathcal{F}_\sigma$ with $\sigma = \lceil \xi \rceil$.

Proof Each term $|X|/|T_q|$ of (1) is at most 1 because $X \subseteq T_q$. If $\text{occ}(X) \geq \xi$, then $\text{supp}(X) \geq \xi$, and $\text{supp}(X) \geq \lceil \xi \rceil$ because support is an integer. $\square$

Throughout, $\sigma = \lceil \xi \rceil$. The closure of an itemset X with $\text{supp}(X) > 0$ is $\text{cl}(X) = \bigcap_{q \in \Gamma(X)} T_q$, and X is closed if $\text{cl}(X) = X$; $X \subseteq \text{cl}(X)$ and $\Gamma(\text{cl}(X)) = \Gamma(X)$ [17]. Let $\mathcal{C}_\sigma$ be the set of closed itemsets in $\mathcal{F}_\sigma$.

Lemma 2 If $\text{supp}(X) > 0$, then $\text{occ}(\text{cl}(X)) = \frac{|\text{cl}(X)|}{|X|} \text{occ}(X) \geq \text{occ}(X)$. In particular, if $X \in \mathcal{H}$ then $\text{cl}(X) \in \mathcal{H}$.

Proof X and $\text{cl}(X)$ have the same cover, so the sum in (1) is the same for both; only the factor $|X|$ changes, and $|\text{cl}(X)| \geq |X|$. $\square$

Example 2 In Table 1, $\text{cl}(\{d, f\}) = T_2 \cap T_3 \cap T_5 \cap T_6 = \{c, d, f\}$, and $\text{occ}(\{c, d, f\}) = \frac{3}{2} \text{occ}(\{d, f\}) = \frac{3}{2} \cdot \frac{49}{30} = \frac{49}{20}$. All 14 HOIs of Table 2 are closed.

Theorem 3 Every MHOI is closed, and $\mathcal{M} = \text{Max}(\mathcal{H} \cap \mathcal{C}_\sigma)$.

Proof Let $X \in \mathcal{M}$. If X were not closed, $\text{cl}(X) \supsetneq X$ would be an HOI by Lemma 2, contradicting maximality. By Lemma 1, $X \in \mathcal{F}_\sigma$, so $X \in \mathcal{H} \cap \mathcal{C}_\sigma$; since X has no proper superset in $\mathcal{H}$, it has none in $\mathcal{H} \cap \mathcal{C}_\sigma$. Conversely, let $X \in \text{Max}(\mathcal{H} \cap \mathcal{C}_\sigma)$ and suppose that some $Y \in \mathcal{H}$ satisfies $Y \supsetneq X$. By Lemmas 2 and 1, $\text{cl}(Y) \in \mathcal{H} \cap \mathcal{C}_\sigma$ and $\text{cl}(Y) \supsetneq Y \supsetneq X$, a contradiction. Hence $X \in \mathcal{M}$. $\square$

Theorem 3 restricts the search to frequent closed itemsets. The next result relates MHOIs to maximal frequent itemsets.

Theorem 4(a) $\text{MFI}_\sigma \cap \mathcal{H} \subseteq \mathcal{M}$.

- (b) The inclusion $\mathcal{M} \subseteq \text{MFI}_\sigma$ does not hold in general, even when all transactions have the same length.
- (c) If all transactions have length L , then $X \in \mathcal{H}$ if and only if $\text{supp}(X) \geq \xi L / |X|$.

Proof (a) Let $M \in \text{MFI}_\sigma \cap \mathcal{H}$. A proper superset Y of M that is an HOI would satisfy $\text{supp}(Y) \geq \sigma$ by Lemma 1, contradicting $M \in \text{MFI}_\sigma$. (b) In Example 1, $\sigma = 2$ and $\text{MFI}_2 = \{\{a, b, c, d, e\}, \{a, c, d, e, f\}, \{b, c, d, e, f\}\}$, each of occupancy $11/6 < 2$ (Table 2); none of them is an HOI, and none of the four MHOIs is an MFI, so keeping the MFIs that are HOIs returns the empty set. For equal lengths, let $L = 3$, $\xi = 2$ and $D =$

$\langle \{a, b, x_1\}, \{a, b, x_2\}, \{a, c, x_3\}, \{a, c, x_4\}, \{a, d, x_5\}, \{a, d, x_6\} \rangle$. Then $\text{occ}(\{a\}) = 6/3 = 2$; each of $\{a, b\}$, $\{a, c\}$, $\{a, d\}$ has support 2 and occupancy $4/3$; every other itemset with two or more items has support at most 1 and occupancy at most 1; and every item other than a has occupancy at most $2/3$. Hence $\mathcal{M} = \{\{a\}\}$, while $\text{MFI}_2 = \{\{a, b\}, \{a, c\}, \{a, d\}\}$. (c) With equal lengths, $\text{occ}(X) = |X| \text{supp}(X)/L$. $\square$

By (c), on databases with equal transaction lengths the HOIs are the itemsets that satisfy the length-decreasing support constraint $f(\ell) = \xi L/\ell$ of [13], and the MHOIs are the maximal such itemsets. A superset of an HOI is longer, so it needs less support, but it can have much less support; this is why (b) holds also for equal lengths.

5 Upper bounds

5.1 An existing upper bound

Definition 3 (UBO [4, Definition 7]) Let X be an itemset with $\text{supp}(X) > 0$, and let $l_1 < \dots < l_u$ be the distinct lengths of the transactions in $\Gamma(X)$. For $1 \leq x \leq u$, let

$$\text{UBO}_x(X) = l_x \sum_{q \in \Gamma(X), |T_q| \geq l_x} \frac{1}{|T_q|},$$

and $\text{UBO}(X) = \max_{1 \leq x \leq u} \text{UBO}_x(X)$.

Deng proved that $\text{occ}(Y) \leq \text{UBO}(X)$ for every superset Y of X [4, Theorem 4]; the HOI miners of [4, 5] do not extend X when $\text{UBO}(X) < \xi$.

Example 3 The cover of $\{a, c, d, e\}$ in Table 1 consists of T_5 , T_6 and T_7 , of lengths 5, 6 and 5. Hence $\text{UBO}_1 = 5 \left(\frac{1}{5} + \frac{1}{6} + \frac{1}{5} \right) = \frac{17}{6}$ and $\text{UBO}_2 = 6 \cdot \frac{1}{6} = 1$, so $\text{UBO}(\{a, c, d, e\}) = \frac{17}{6} \approx 2.833 \geq 2$: with $\xi = 2$, UBO does not prune the supersets of $\{a, c, d, e\}$.

Proposition 5 *If all transactions of $\Gamma(X)$ have the same length, then $\text{UBO}(X) = \text{supp}(X)$. In particular, if all transactions of D have the same length, UBO prunes no itemset X with $\text{supp}(X) \geq \xi$.*

Proof With a single length l_1 , $\text{UBO}(X) = \text{UBO}_1(X) = l_1 \cdot \text{supp}(X)/l_1$. $\square$

On such databases, a search that prunes with UBO enumerates all itemsets with support at least σ , as the baseline algorithm of [4] does.

5.2 A size-coupled bound

Consider a node of a depth-first search that holds an itemset C with $\text{supp}(C) \geq \sigma$, and a set R of items not in C , its tail, such that every descendant of the node is an

itemset $C \cup Z$ with $\emptyset \neq Z \subseteq R$ (for the search of Section 6, this is Lemma 10). For $q \in \Gamma(C)$, the capacity of transaction q is $\text{cap}(q) = |T_q \cap R|$. For $k \geq 1$, let

$$Q_k = \{q \in \Gamma(C) : \text{cap}(q) \geq k\}, \quad S_k = \sum_{q \in Q_k} \frac{1}{|T_q|}.$$

Theorem 6 (Size-coupled bound) *Let*

$$B(C) = \max\{(|C| + k) S_k : k \geq 1, |Q_k| \geq \sigma\}, \quad (2)$$

with $B(C) = 0$ if no k satisfies $|Q_k| \geq \sigma$. Every itemset $Y = C \cup Z$ with $\emptyset \neq Z \subseteq R$ that is an HOI satisfies $\text{occ}(Y) \leq B(C)$. Hence, if $B(C) < \xi$, no descendant of the node is an HOI.

Proof Let $k = |Z| \geq 1$. Every $q \in \Gamma(Y)$ belongs to $\Gamma(C)$ and contains $Z \subseteq R$, so $\text{cap}(q) \geq k$ and $\Gamma(Y) \subseteq Q_k$. Since Y is an HOI, $|Q_k| \geq \text{supp}(Y) \geq \sigma$ by Lemma 1. As $|Y| = |C| + k$ and all terms are positive,

$$\text{occ}(Y) = (|C| + k) \sum_{q \in \Gamma(Y)} \frac{1}{|T_q|} \leq (|C| + k) S_k \leq B(C).$$

□

Theorem 7 $B(C) \leq \text{UBO}(C)$.

Proof If $B(C) = 0$ there is nothing to prove. Otherwise let $k \geq 1$ attain the maximum in (2), so $Q_k \neq \emptyset$, and let $l = \min\{|T_q| : q \in Q_k\}$, which is one of the lengths l_x of Definition 3. For $q \in Q_k$, C and $T_q \cap R$ are disjoint subsets of T_q , so $|T_q| \geq |C| + \text{cap}(q) \geq |C| + k$; hence $l \geq |C| + k$. Every $q \in Q_k$ satisfies $|T_q| \geq l$, so

$$B(C) = (|C| + k) S_k \leq l \sum_{q \in \Gamma(C), |T_q| \geq l} \frac{1}{|T_q|} = \text{UBO}_x(C) \leq \text{UBO}(C).$$

□

Bounds based on the items that remain to be added are standard in high-utility itemset mining [29, 32, 33]. Transferred to occupancy, such a bound lets every transaction assume the largest extension it can hold:

$$W(C) = \sum_{q \in \Gamma(C), \text{cap}(q) \geq 1} \frac{|C| + \text{cap}(q)}{|T_q|}.$$

Every HOI $Y = C \cup Z$ with $\emptyset \neq Z \subseteq R$ satisfies $\text{occ}(Y) \leq W(C)$, since each term of $\text{occ}(Y)$ is at most the corresponding term of $W(C)$.

Theorem 8 $B(C) \leq W(C)$.

Proof For $k \geq 1$ with $|Q_k| \geq \sigma$, every $q \in Q_k$ satisfies $\text{cap}(q) \geq k \geq 1$, so $(|C| + k)S_k = \sum_{q \in Q_k} \frac{|C|+k}{|T_q|} \leq \sum_{q \in Q_k} \frac{|C|+\text{cap}(q)}{|T_q|} \leq W(C)$. $\square$

W lets each transaction assume its own extension size; B uses one size k for all transactions that support the extension. UBO bounds the size of a superset by the length of the shortest transaction that may contain it. B bounds it by the number of tail items that the transactions actually contain, and uses the same size for all transactions that support the superset.

Example 4 In the search of Section 6 on Table 1 (Example 7), the node that holds $C = \{a, c, d, e\}$ has the tail $R = \{f\}$. Among the three transactions of $\Gamma(C)$, T_5 and T_6 contain f and T_7 does not, so $Q_1 = \{5, 6\}$, $|Q_1| = 2 \geq \sigma$, and

$$B(\{a, c, d, e\}) = (4 + 1) \left(\frac{1}{5} + \frac{1}{6} \right) = \frac{11}{6} \approx 1.833 < 2.$$

Rule P1 skips the subtree of the node, whereas $\text{UBO}(\{a, c, d, e\}) = 17/6 \geq 2$ (Example 3); the node itself, an HOI, is recorded. Its only descendant, $\{a, c, d, e, f\}$, has occupancy $11/6$. UBO allows a superset of $\{a, c, d, e\}$ to be supported by T_7 , which contains no item of the tail; B counts only the transactions that contain a tail item.

Example 5 Let all transactions have length $L = 10$, $|C| = 1$, $\text{supp}(C) = 10$, $\text{cap}(q) = 1$ for every $q \in \Gamma(C)$, and $\xi = 5$. By Proposition 5, $\text{UBO}(C) = 10 \geq \xi$, so UBO does not prune. $Q_1 = \Gamma(C)$ and $Q_k = \emptyset$ for $k \geq 2$, so $B(C) = 2 \cdot 10/10 = 2 < \xi$ and the node is pruned. Indeed, every descendant has two items and support at most 10, hence occupancy at most 2.

If all transactions have length L , then $S_k = |Q_k|/L$ and $B(C) = \max\{(|C| + k)|Q_k|/L : k \geq 1, |Q_k| \geq \sigma\}$: the bound is the largest value of $|Y| \cdot \text{supp}/L$ that the capacities allow, in agreement with Theorem 4(c).

A parent can evaluate the bound for a child before building the child's database. Let the tail R be ordered, let $e \in R$, and let the child node hold $C \cup \{e\}$ with tail $R_{>e} = \{x \in R : x \succ e\}$. For $q \in \Gamma(C \cup \{e\})$ let $\text{cap}_e(q) = |T_q \cap R_{>e}|$; it is obtained from the transactions of the parent by counting the tail items after e .

Corollary 9 (Child bound) *For $k \geq 0$ let $Q_k^e = \{q \in \Gamma(C \cup \{e\}) : \text{cap}_e(q) \geq k\}$ and $S_k^e = \sum_{q \in Q_k^e} 1/|T_q|$. Every itemset $Y = C \cup \{e\} \cup Z'$ with $Z' \subseteq R_{>e}$ (possibly empty) that is an HOI satisfies*

$$\text{occ}(Y) \leq B_e(C) := \max\{(|C| + 1 + k)S_k^e : k \geq 0, |Q_k^e| \geq \sigma\},$$

with $B_e(C) = 0$ if no k satisfies $|Q_k^e| \geq \sigma$.

Proof Apply the argument of Theorem 6 with $k = |Z'| \geq 0$: every $q \in \Gamma(Y)$ belongs to $\Gamma(C \cup \{e\})$ and contains $Z' \subseteq R_{>e}$. $\square$

Example 6 In the same search, the node that holds $C = \{b\}$, with $\Gamma(C) = \{1, 2, 6, 7\}$, has the tail $R = (a, c, d, e, f)$ in this order. For the child with a : $\Gamma(C \cup \{a\}) = \{6, 7\}$, $R_{>a} = \{c, d, e, f\}$,

$\text{cap}_a(6) = 4$ and $\text{cap}_a(7) = 3$. For $k = 0, \dots, 3$ both transactions qualify, and for $k = 4$ only T_6 does, which is fewer than $\sigma = 2$. Hence

$$B_a(C) = \max_{0 \leq k \leq 3} (2 + k) \left(\frac{1}{6} + \frac{1}{5} \right) = 5 \cdot \frac{11}{30} = \frac{11}{6} \approx 1.833 < 2,$$

and the child $\{a, b\}$ is skipped with its subtree, which contains 8 itemsets with support at least 2; the largest occupancy among them is that of $\{a, b, c, d, e\}$, $11/6$. For the child with f , $\Gamma(C \cup \{f\}) = \{2, 6\}$ and no tail item follows f , so $B_f(C) = 2 \left(\frac{1}{5} + \frac{1}{6} \right) = \frac{11}{15} < 2$.

Computation.

Given the pairs $(|T_q|, \text{cap}(q))$ of the node, accumulating $1/|T_q|$ and the number of transactions per capacity value and summing them from the largest capacity down to 1 gives all S_k and $|Q_k|$, and hence $B(C)$, in $O(r + c_{\max})$ time, where r is the number of rows of the node (transactions of the cover, merged as described in Section 6) and $c_{\max}$ the largest capacity.

6 The MHOI-Miner algorithm

6.1 Search space

Order the items i with $\text{supp}(\{i\}) \geq \sigma$ by increasing support, breaking ties by a fixed order on item identifiers. MHOI-Miner searches the set-enumeration tree of the itemsets with support at least σ [12], which contains all HOIs by Lemma 1. The root holds the empty itemset. A node holding C , whose last item in the order is $\text{last}(C)$ (none at the root), has a child $C \cup \{e\}$ for every item e of its tail

$$R(C) = \{e \succ \text{last}(C) : \text{supp}(C \cup \{e\}) \geq \sigma\},$$

with $\text{last}(C)$ taken to precede every item at the root. Since support is anti-monotone, every itemset with support at least σ is the itemset of exactly one node: its items, added in increasing order, form a path from the root.

Lemma 10 *Every descendant of the node holding C is an itemset $C \cup Z$ with $\emptyset \neq Z \subseteq R(C)$. Moreover, for $e \in R(C)$, the child $C \cup \{e\}$ and all its descendants are contained in $C \cup \{e\} \cup \{x \in R(C) : x \succ e\}$.*

Proof The tail of the child $C \cup \{e\}$ consists of items $x \succ e$ with $\text{supp}(C \cup \{e, x\}) \geq \sigma$; as $\Gamma(C \cup \{e, x\}) \subseteq \Gamma(C \cup \{x\})$, these items satisfy $\text{supp}(C \cup \{x\}) \geq \sigma$, so $R(C \cup \{e\}) \subseteq \{x \in R(C) : x \succ e\}$. Both claims follow by induction on the depth. $\square$

Lemma 10 is the setting of Section 5: a node C with a tail R whose descendants are of the form $C \cup Z$, $\emptyset \neq Z \subseteq R$. The search does not restrict itself to closed itemsets, although every MHOI is closed (Theorem 3): a version that enumerates only closed itemsets by prefix-preserving closure extension [19] visits fewer nodes, but computing closures at every node made it slower on all our datasets (Section 7.5). Closures are used only by rule P2 below.

6.2 Conditional databases

A node keeps the transactions of its cover, projected on the items of its tail. Two projected transactions with the same items *and the same original length* are merged into one row with a weight equal to the number of merged transactions.

Lemma 11 *Merging rows with the same projected items and the same original length preserves, for every node and every candidate extension, the support, the sum $\sum 1/|T_q|$ over the cover, and the multiset of pairs $(|T_q|, \text{cap}(q))$.*

Proof Each of these quantities is a sum or a multiset over transactions in which a transaction enters only through its projected items and its original length, and the weight counts multiplicity. $\square$

Merging on items alone, as in frequent itemset mining [20], would change $\sum 1/|T_q|$ and therefore occupancy.

6.3 Pruning rules

At a node holding C with tail R , MHOI-Miner applies the following rules, in this order.

- P1** (subtree bound) If $B(C) < \xi$ (Theorem 6), no descendant is an HOI; the subtree is skipped.
- P2** (head-union-tail) Let $H = C \cup R$. If $\text{occ}(H) \geq \xi$, then $\text{cl}(H)$ is an HOI (Lemma 2) that contains every descendant; it is recorded unless it is a subset of a recorded itemset, and the subtree is skipped. $\text{cl}(H)$ is computed as the intersection of the transactions of $\Gamma(H)$.
- P3** (subsumption) If $C \cup R$ is a subset of a recorded itemset, no descendant is an MHOI that has not been recorded; the subtree is skipped.
- P4** (child bound) Before generating the child for $e \in R$, if $B_e(C) < \xi$ (Corollary 9), the child and its subtree are skipped without building the child's database.

K denotes the set of recorded itemsets. MINE returns true when it has found that some HOI contains C : C itself, an HOI recorded by P2 or found below a child, or a recorded HOI that contains $C \cup R$ (P3). A node whose itemset is an HOI is recorded when its tail is empty or P1 applies, and after its children have been explored if none of them returned true; it is not recorded when P2 or P3 applies, since an HOI that contains it is then known. After the search, the recorded itemsets that are subsets of other recorded itemsets are removed. Rule P2 is the head-union-tail look-ahead of maximal frequent itemset mining [12, 21] with the support condition replaced by the occupancy condition, and rule P3 is the corresponding superset check.

6.4 Example

Example 7 We run MHOI-Miner on Table 1 with $\xi = 2$, so $\sigma = 2$. The supports of a, b, c, d, e, f are 5, 4, 6, 6, 6, 6, so the items are ordered $b \prec a \prec c \prec d \prec e \prec f$ (ties broken alphabetically).

Algorithm 1 MHOI-Miner

Require: database D , occupancy threshold ξ

Ensure: the set $\mathcal{M}$ of MHOIs

```
1:  $\sigma \leftarrow \lceil \xi \rceil$ ;  $K \leftarrow \emptyset$  ▷ set of recorded HOIs
2: build the root database (items with support  $\geq \sigma$ , merged rows);  $\text{MINE}(\emptyset, \text{root rows, none})$ 
3: return the elements of  $K$  that are not subsets of other elements of  $K$ 

4: function  $\text{MINE}(C, \text{rows, last})$  ▷ returns true only if some HOI contains  $C$ 
5:   compute  $\text{occ}(C)$ , the tail  $R$  and the capacities from the rows
6:    $f \leftarrow [C \neq \emptyset \text{ and } \text{occ}(C) \geq \xi]$ 
7:   if  $R = \emptyset$  or  $B(C) < \xi$  then ▷ P1
8:     if  $f$  then add  $C$  to  $K$ ; return  $f$ 
9:   if  $\text{occ}(C \cup R) \geq \xi$  then ▷ P2
10:     add  $\text{cl}(C \cup R)$  to  $K$  unless it is a subset of an element of  $K$ ; return true
11:   if  $C \cup R \subseteq M$  for some  $M \in K$  then return true ▷ P3
12:    $g \leftarrow \text{false}$ 
13:   for  $e \in R$  in increasing order do
14:     if  $B_e(C) < \xi$  then continue ▷ P4
15:     build the rows of  $C \cup \{e\}$  (projection on the items after  $e$  with support  $\geq \sigma$ 
      in  $\Gamma(C \cup \{e\})$ , merging)
16:     if  $\text{MINE}(C \cup \{e\}, \text{its rows, } e)$  then  $g \leftarrow \text{true}$ 
17:   if  $f$  and not  $g$  then add  $C$  to  $K$ 
18:   return  $f$  or  $g$ 
```

The set-enumeration tree of the itemsets with support at least 2 has 55 nodes besides the root. Table 3 lists the 16 nodes that MHOI-Miner visits, in the order of the search, and Fig. 1 shows them together with the parts of the tree that are skipped.

Root. Every item is in the tail, so $\text{cap}(q) = |T_q|$ for every transaction and $B(\emptyset) = \max_k k \sum_{|T_q| \geq k} 1/|T_q| = 101/20 = 5.05$, attained at $k = 3$. The child bounds of e and f are $9/5$ and $33/20$, so rule P4 skips $\{e\}$ and $\{f\}$ and their subtrees; the children $\{b\}$, $\{a\}$, $\{c\}$ and $\{d\}$ are built.

Subtree of $\{b\}$. P4 skips the extensions of $\{b\}$ with a (Example 6), e and f . Along $\{b, c\}$ and $\{b, c, d\}$, which are not HOIs, P4 skips four more extensions, and the search reaches $\{b, c, d, e\}$, an HOI whose tail is $\{f\}$ and whose bound is $B = 11/6 < 2$: P1 records it. Similarly, $\{b, d, e\}$, reached through $\{b, d\}$, is an HOI with tail $\{f\}$ and $B = 22/15 < 2$, and P1 records it as well.

Subtree of $\{a\}$. Along $\{a, c\}$ the search reaches $\{a, c, d\}$, an HOI with two children: $\{a, c, d, e\}$, recorded by P1 (Example 4), and $\{a, c, d, f\}$, whose tail is empty and which is recorded. At $\{a, d\}$, P4 skips both extensions, so the node has no child. $\{a, e\}$ is skipped by P4, and $\{a, f\}$, an HOI with an empty tail, is recorded.

Node $\{c\}$. The tail is (d, e, f) , and the transactions that contain all three items are T_2 , T_5 and T_6 . Hence $\text{occ}(\{c, d, e, f\}) = 4(\frac{1}{5} + \frac{1}{5} + \frac{1}{6}) = \frac{34}{15} \geq 2$, and rule P2 records $\text{cl}(\{c, d, e, f\}) = T_2 \cap T_5 \cap T_6 = \{c, d, e, f\}$. The seven itemsets of the subtree of $\{c\}$ are not visited; all of them are HOIs and subsets of $\{c, d, e, f\}$.

Table 3 Nodes visited by MHOI-Miner on Table 1 with $\xi = 2$, in the order of the search (Example 7). Child bounds B_e of skipped extensions in parentheses; UBO is given where P1 applies with a non-empty tail.

Node C	$\Gamma(C)$	$\text{occ}(C)$	Tail R	$B(C)$	Outcome
1 $\emptyset$	1–8	–	b, a, c, d, e, f	5.05	children b, a, c, d ; P4: e (1.80), f (1.65)
2 $\{b\}$	1, 2, 6, 7	0.90	a, c, d, e, f	2.83	children c, d ; P4: a (1.83), e (1.80), f (0.73)
3 $\{b, c\}$	2, 6, 7	1.13	d, e, f	2.27	child d ; P4: e (1.70), f (1.10)
4 $\{b, c, d\}$	2, 6, 7	1.70	e, f	2.27	child e ; P4: f (1.47)
5 $\{b, c, d, e\}$	2, 6, 7	2.27	f	1.83	P1 (UBO = 2.83); recorded
6 $\{b, d\}$	1, 2, 6, 7	1.80	e, f	2.70	child e ; P4: f (1.10)
7 $\{b, d, e\}$	1, 2, 6, 7	2.70	f	1.47	P1 (UBO = 2.83); recorded
8 $\{a\}$	3–7	1.32	c, d, e, f	3.27	children c, d, f ; P4: e (1.13)
9 $\{a, c\}$	3, 5, 6, 7	1.63	d, e, f	3.27	child d ; P4: e (1.70), f (1.85)
10 $\{a, c, d\}$	3, 5, 6, 7	2.45	e, f	3.27	children e, f
11 $\{a, c, d, e\}$	5, 6, 7	2.27	f	1.83	P1 (UBO = 2.83); recorded
12 $\{a, c, d, f\}$	3, 5, 6	2.47	–	–	empty tail; recorded
13 $\{a, d\}$	3, 5, 6, 7	1.63	e, f	2.45	P4: e (1.70), f (1.85)
14 $\{a, f\}$	3, 4, 5, 6	2.23	–	–	empty tail; recorded
15 $\{c\}$	2, 3, 5–8	1.35	d, e, f	4.05	P2: records $\{c, d, e, f\}$
16 $\{d\}$	1–3, 5–7	1.35	e, f	2.70	P3

Node $\{d\}$. $C \cup R = \{d, e, f\} \subseteq \{c, d, e, f\}$, which has been recorded, so rule P3 skips the subtree, which contains the HOI $\{d, e\}$.

The search visits 16 of the 56 nodes. It records $\{b, c, d, e\}$, $\{b, d, e\}$, $\{a, c, d, e\}$, $\{a, c, d, f\}$, $\{a, f\}$ and $\{c, d, e, f\}$; the final filter removes $\{b, d, e\} \subset \{b, c, d, e\}$ and $\{a, f\} \subset \{a, c, d, f\}$ and returns the four MHOIs of Example 1. Of the 40 itemsets that are not visited, 8 are HOIs, all contained in $\{c, d, e, f\}$, which rule P2 has recorded. For comparison, the existing HOI miners [4, 5] compute the supporting transactions of 41 itemsets on this database and report the 14 HOIs, from which the maximal ones are then selected.

6.5 Correctness

Theorem 12 *Algorithm 1 returns exactly $\mathcal{M}$.*

Proof Every element of K is an HOI: C is added only if $\text{occ}(C) \geq \xi$, and $\text{cl}(C \cup R)$ only if $\text{occ}(C \cup R) \geq \xi$, in which case its closure is an HOI by Lemma 2. Moreover, if $\text{MINE}(C)$ returns true, then some HOI contains C : this holds when C is an HOI, when P2 applies ($C \subseteq \text{cl}(C \cup R)$), when P3 applies ($C \subseteq C \cup R \subseteq M$), and, by induction on the height of the subtree, when a child $C \cup \{e\} \supsetneq C$ returns true. We show that every MHOI X is in K at the end of the search. The final filter then returns exactly $\mathcal{M}$: every element of K that is not an

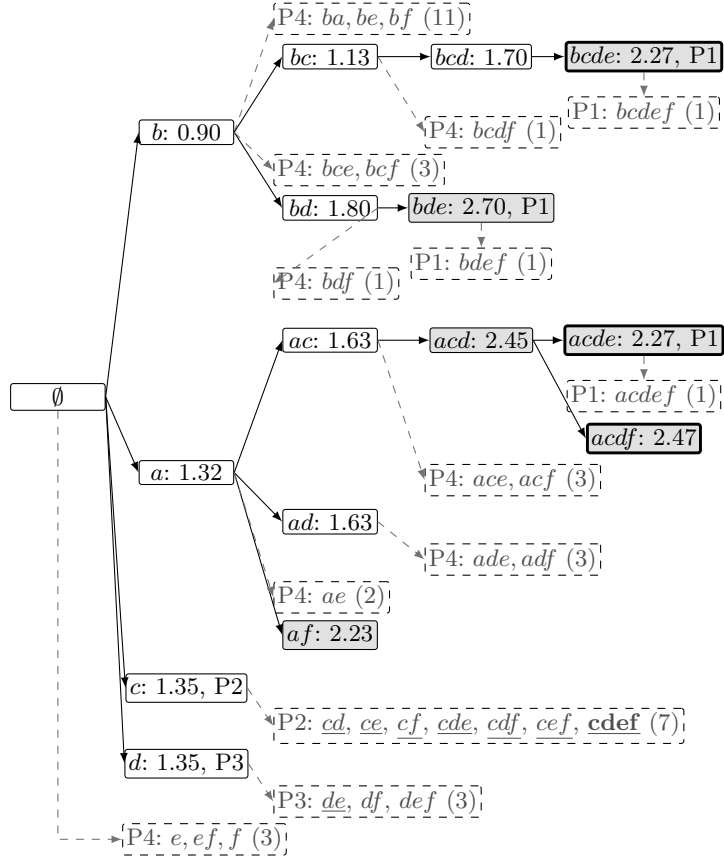


Fig. 1 Search of MHOI-Miner on Table 1 with $\xi = 2$ (Example 7, Table 3). Solid boxes: the 16 visited nodes, with the occupancy of their itemset and the rule (P1, P2, P3) that ends the search below them; items are listed in the search order. Shaded boxes: HOIs; thick boxes: MHOIs. Dashed boxes: the itemsets of the set-enumeration tree that are not visited, grouped by the rule that skips them, with their number; together with the visited nodes they are the 55 itemsets with support at least 2. In the dashed boxes, HOIs are underlined, and the MHOI $cdef$, recorded by rule P2, is in bold.

MHOI is a proper subset of an MHOI, which is in K , and an MHOI is not a proper subset of any HOI.

By Lemma 1, $\text{supp}(X) \geq \sigma$. Let $x_1 \prec \dots \prec x_t$ be the items of X and $C_i = \{x_1, \dots, x_i\}$; every C_i has support at least $\text{supp}(X) \geq \sigma$, so $C_0 = \emptyset, C_1, \dots, C_t = X$ is a path of the tree. Consider an ancestor C_i , $i < t$, with tail R . For $j > i$, $\text{supp}(C_i \cup \{x_j\}) \geq \text{supp}(X) \geq \sigma$, so $x_{i+1}, \dots, x_t \in R$, and $X = C_i \cup \{x_{i+1}\} \cup Z'$ with $Z' \subseteq \{x \in R : x \succ x_{i+1}\}$. Since X is an HOI, P1 does not apply at C_i , by Theorem 6, and P4 does not skip C_{i+1} , by Corollary 9. If P2 applies at C_i , then $X \subseteq C_i \cup R \subseteq \text{cl}(C_i \cup R)$, which is an HOI, so $X = \text{cl}(C_i \cup R)$ by maximality; it is added to K unless it is a subset of an element of K , which is an HOI and therefore equals X . If P3 applies at C_i , then $X \subseteq C_i \cup R \subseteq M$ for some $M \in K \subseteq \mathcal{H}$, so $X = M \in K$. Hence, unless X has already been placed in K , the node holding X is visited. There, every itemset of the subtree is a proper superset of X , so none is an HOI. If the tail is empty or P1 applies, X is added to K . With a non-empty tail R , P2 cannot apply, since $\text{cl}(X \cup R) \supsetneq X$ would be an HOI, and P3 cannot apply, since an element $M \supseteq X \cup R \supsetneq X$

of K would be an HOI. The children are then explored; a child returning true would give an HOI containing a proper superset of X , so none does, and X is added to K after the loop. $\square$

6.6 Complexity

A node costs one pass over its rows to count the items and to compute $\text{occ}(C)$ and the capacities, and $O(r + c_{\max})$ for P1, where r is the number of rows of the node. Rule P2 intersects the transactions that contain $C \cup R$, and rule P3 compares $C \cup R$ with the recorded itemsets in time linear in their total size. For each child, P4 needs one pass over the rows that contain e , and a generated child needs one more pass to build its rows and $O(r \log r)$ to merge them. The final filter compares pairs of recorded itemsets. The number of nodes depends on the data and the threshold; Section 7 measures it. The memory consists of the conditional databases on the current path of the search and the recorded itemsets.

7 Experimental evaluation

7.1 Setup

Methods.

No algorithm for MHOIs has been published. We compare MHOI-Miner with the two published HOI miners, each followed by a maximality filter:

- HEP+FILTER: HEP [4] computes all HOIs; the HOIs that are subsets of other HOIs are then removed.
- DFHOI+FILTER: DFHOI [5] computes all HOIs, followed by the same filter.

The filter sorts the itemsets by decreasing size and keeps an itemset if it is not a subset of an itemset kept before. To reduce the input of the filter, an HOI is not passed to it when an HOI was generated from it during the search (a join parent in HEP, an ancestor in DFHOI); such an HOI has an HOI superset and cannot be maximal. All three methods are therefore exact.

Implementations.

All methods are written in C. HEP and DFHOI follow the implementations of these algorithms in our pattern-mining code base: the order of the search, the UBO test of Definition 3, and the shortcut of DFHOI for databases in which all transactions have the same length L (it then computes $\text{occ}(X) = |X| \text{supp}(X)/L$ directly and uses $\text{supp}(X)$ as the bound) are unchanged; items are ordered by increasing support for HEP as in [4, Section 3.2.3]; the early termination of the UBO scan described there is not implemented, which affects the time of a UBO evaluation but not the pruning; and the itemsets are recorded for the filter. MHOI-Miner and the two baselines use the same input reader, the same comparison tolerance for thresholds and the same final filter. We compare the search effort by the number of *evaluated itemsets*. For HEP and DFHOI, it is the number of itemsets whose list of supporting transactions is computed. For MHOI-Miner, it is the number of visited nodes (calls of MINE in Algorithm 1) plus the number of extensions skipped by rule P4, since computing the

child bound B_e requires a pass over the cover of $C \cup \{e\}$; it exceeds the number of visited nodes by a factor between 2.2 and 33 in our experiments.

Platform.

All experiments ran on one workstation: an Intel Core i7-10700 processor (8 cores, 16 hardware threads, base frequency 2.9 GHz, maximum turbo frequency 4.8 GHz; 32 KiB L1 data cache and 256 KiB L2 cache per core, 16 MiB shared L3 cache) on an ASUS ROG STRIX B460-H GAMING motherboard, with 32 GiB of DDR4 memory and an NVMe solid-state drive, running Ubuntu 24.04.5 LTS with Linux kernel 6.8.0-146-generic. CPU frequency was managed by the `intel_pstate` driver with the `powersave` governor and turbo enabled, which are the default settings of the system. The programs were compiled with GCC 13.3.0 at optimization level `-O2` (glibc 2.39).

Protocol.

Every run used one thread and was pinned to one core with `taskset`. Runtime is the time spent in mining and in the final filter, measured with a monotonic clock; reading the input and building the initial database representation (the root database for MHOI-Miner, the lists of supporting transactions of single items for HEP and DFHOI) are excluded. Peak memory is the peak resident set size of the process (`VmHWM` in `/proc/self/status` [35]), which includes the loaded database; it is reported in MiB. Each run was limited to 600 s of wall-clock time and 20 GiB of address space. A configuration was repeated five times if its first run took less than 1 s, three times if it took less than 60 s, and once otherwise; we report the median runtime. Runs longer than 60 s are therefore single measurements. For each dataset the thresholds were run from the largest to the smallest, and a method that exceeded a limit was not run at smaller thresholds, since a smaller ξ can only enlarge its search. Visited nodes and outputs did not vary between repetitions.

7.2 Datasets and thresholds

We use the datasets and threshold ranges of Deng [4, Section 4] (Table 4): **mushroom**, derived from the UCI Mushroom data [36]; **retail**, market baskets of a Belgian supermarket [37]; **accidents**, traffic accident records [38]; **kosarak**, anonymized click-stream data of a Hungarian online news portal, provided by Ferenc Bodon to the FIMI repository [39]; and **T10I4D100K**, a synthetic dataset generated with the IBM generator [14]. Deng also used **T40I10D100K**; it is not part of the dataset collection we used, and we did not include it. The files were obtained from the SPMF library [40, 41]; its version of **mushroom** has 8,416 transactions, whereas the FIMI version used by Deng has 8,124, and its longest **kosarak** transaction has 2,497 distinct items, against 2,498 in [4, Table 2]. As in [4], ξ is given as a percentage of the number of transactions. For each dataset we use five thresholds spanning the range of Table 3 of [4]: 12% to 4% for **mushroom**, 0.05% to 0.01% for **retail**, 30% to 10% for **accidents**, 1% to 0.2% for **kosarak** and 0.1% to 0.02% for **T10I4D100K**. All transactions of **mushroom** have the same length, so Proposition 5 applies to it.

Table 4 Datasets. Length: minimum–maximum transaction length (a single value means that all transactions have the same length).

Dataset	Type	Transactions	Items	Avg. length	Length
mushroom	real, dense	8,416	119	23.0	23
retail	real, sparse	88,162	16,470	10.3	1–76
accidents	real, dense	340,183	468	33.8	18–51
kosarak	real, sparse	990,002	41,270	8.1	1–2497
T10I4D100K	synthetic	100,000	870	10.1	1–29

7.3 Correctness

We verified the implementations against an exact oracle. For 900 random databases with 4 to 10 frequent items and 4 to 24 transactions, of which one third have equal transaction lengths and one third contain additional items that occur once and only lengthen transactions, and for nine thresholds per database (ξ from 5% to 90% of the number of transactions, and $\xi = 1.5$), the oracle enumerates every itemset contained in some transaction, computes its occupancy in rational arithmetic and returns the maximal HOIs. In all 8,100 cases, MHOI-Miner, HEP+FILTER, DFHOI+FILTER and the six variants of Section 7.5 returned exactly the oracle’s family of itemsets. On the benchmark datasets, all methods that completed a configuration returned the same family of MHOIs. We compared the sorted families: the closed version of the algorithm (Section 7.5) returned the same family as HEP and DFHOI in all 23 configurations completed by at least two of these methods, and MHOI-Miner returned the same family as the closed version in all 25 configurations.

7.4 Results

Table 5 reports, for every dataset and threshold, the number of MHOIs, the number of HOIs (from HEP or DFHOI), and the evaluated itemsets, runtime and peak memory of the three methods; Fig. 2 and Fig. 3 plot these measurements.

Number of HOIs and MHOIs.

Fig. 2 compares the number of HOIs with the number of MHOIs. On **mushroom**, the HOIs outnumber the MHOIs by a factor between 15.5 and 3,134; at $\xi = 4\%$ there are 335,388 HOIs and 107 MHOIs. On **T10I4D100K** the factor is between 17.1 and 27.1, on **kosarak** between 3.94 and 4.88, and on **retail** between 3.74 and 3.93. The factor is largest on **mushroom**, the dense dataset in which all transactions have the same length, and smallest on the sparse datasets **retail** and **kosarak**. On **accidents** there are no HOIs at 30%, 25% and 20%; at 15% and 10%, where neither HOI miner completed, there are 1 and 212 MHOIs.

Agreement with Deng.

The numbers of HOIs computed by our implementations of HEP and DFHOI equal those of Table 3 of [4] at every threshold that appears in both studies on **retail** (1,170, 2,620 and 14,202 at 0.05%, 0.03% and 0.01%), **kosarak** (31, 71 and 265 at 1%, 0.6% and 0.2%) and **T10I4D100K** (4,363, 12,640 and 38,588 at 0.1%, 0.06% and

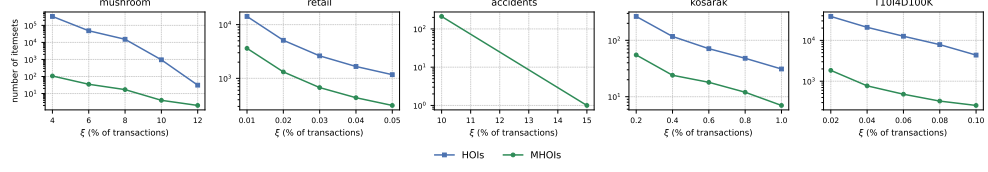


Fig. 2 Number of HOIs (reported by HEP or DFHOI) and of MHOIs as functions of ξ . On **accidents**, there are no HOIs at the thresholds completed by HEP or DFHOI, and only the MHOIs at 15% and 10% are shown.

0.02%). On **mushroom** they do not agree: we find 31, 955, 15,292 and 335,388 HOIs at 12%, 10%, 8% and 4%, whereas [4] reports 157, 2,723, 15,645 and 331,704. Our version of the dataset differs from Deng’s (8,416 against 8,124 transactions), so the counts are not expected to agree, but we have not verified that this difference explains them. On **accidents**, whose statistics agree with Table 2 of [4] (340,183 transactions, 468 items, average length 33.8, maximum length 51), HEP finds no HOI at 30%, and DFHOI finds none at 30%, 25% and 20%, whereas [4] reports 15 at 20%. The absence of HOIs at 20% can also be checked directly: at 15% the only MHOI is an itemset of 8 items with support 221,467 and occupancy 51,767.3, that is, 15.2% of the number of transactions; every HOI at 20% would be an HOI at 15% and hence a subset of this itemset, and evaluating its 255 non-empty subsets shows that none has a larger occupancy. We could not determine the cause of the difference.

Evaluated itemsets.

MHOI-Miner evaluates fewer itemsets than both baselines in every configuration. Over the configurations completed by both methods, the ratio of the itemsets evaluated by HEP+FILTER to those evaluated by MHOI-Miner has median 60.6 (range 1.92 to 2,081, 21 configurations), and the ratio for DFHOI+FILTER has median 125 (range 2.57 to 5,837, 23 configurations). The largest ratios occur on **accidents** at 30% and 25%, where MHOI-Miner evaluates 33 and 58 itemsets; the smallest occur on **kosarak**, where the baselines themselves evaluate few itemsets. On **mushroom**, whose transactions all have length 23, UBO equals the support (Proposition 5), so HEP and DFHOI prune only by support, whereas the bound of Theorem 6 still prunes.

Runtime.

MHOI-Miner is faster than HEP+FILTER in 16 of 21 configurations and faster than DFHOI+FILTER in 19 of 23; the median runtime ratios (baseline over MHOI-Miner) are 14.9 and 24.2. It is slower on **kosarak**: than HEP+FILTER at all five thresholds and than DFHOI+FILTER at the four largest ones, by a factor of up to 2.8 and 2.3, respectively. On this dataset the baselines evaluate at most a few thousand itemsets at the larger thresholds, and the cost of a node of MHOI-Miner, which computes the child bounds and builds a merged conditional database for each child, outweighs the reduction in the number of evaluated itemsets. The reduction in evaluated itemsets is not matched by a reduction in runtime of the same size on the other datasets either, for the same reason. If the time to read the input and build the initial representation

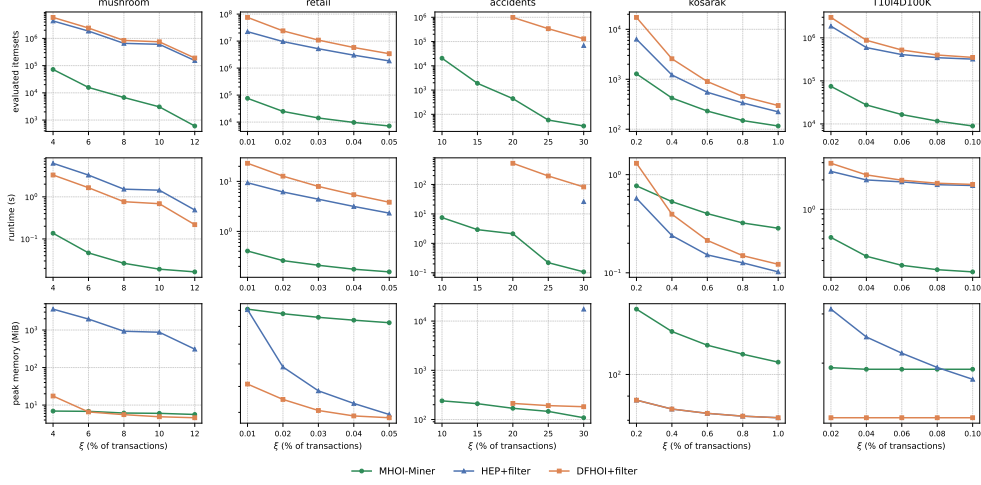


Fig. 3 Evaluated itemsets (top), runtime (middle) and peak memory (bottom) as functions of the occupancy threshold ξ . Missing points: the method exceeded a limit (Table 5).

is included, which is larger for MHOI-Miner because it projects and merges the transactions, the counts are unchanged (16 of 21 and 19 of 23), the median ratios are 10.9 and 17.9, and the slowdown on **kosarak** reaches 2.2 and 2.0.

Limits.

MHOI-Miner completed all 25 configurations; its longest run took 7.5 s (**accidents** at 10%). On **accidents**, HEP+FILTER used 17.1 GiB at 30% and exceeded the 20 GiB limit at 25%, because HEP keeps all candidates of a level with their lists of supporting transactions; DFHOI+FILTER took 525 s at 20% and exceeded the 600 s limit at 15%. At 10%, only MHOI-Miner completed; it returned 212 MHOIs.

Memory.

On **mushroom** and **accidents**, the peak memory of HEP+FILTER is much larger than that of the other methods. In most configurations, MHOI-Miner uses more memory than DFHOI+FILTER, up to 137 MiB against 88 MiB on **kosarak**: it stores the conditional databases of the current search path, whereas DFHOI stores lists of transaction identifiers. Over the configurations completed by both methods, the median ratio of the peak memory of DFHOI+FILTER to that of MHOI-Miner is 0.69.

7.5 Ablation

Table 6 reports, at the middle threshold of each dataset, the visited nodes and the runtime of MHOI-Miner and of six variants, measured with the protocol of Section 7.1. In the variants *no P4*, *W* and *UBO*, rule P4 is disabled and rule P1 uses the bound *B*, the per-transaction bound *W* of Theorem 8 and the bound *UBO* of Definition 3, respectively, so that these three variants differ only in the bound. In *no P2* and *no*

Table 5 Results. ξ in % of the number of transactions; HOIs: number of HOIs reported by HEP or DFHOI; M: MHOI-Miner, H: HEP+FILTER, D: DFHOI+FILTER. TL: time limit (600 s) exceeded; ML: memory limit (20 GiB) exceeded; -: not run because the method failed at a larger threshold.

ξ	MHOIs	HOIs	Evaluated itemsets			Runtime (s)			Peak memory (MiB)		
			M	H	D	M	H	D	M	H	D
mushroom											
12	2	31	605	153,592	190,283	0.017	0.49	0.22	5.6	309	4.6
10	4	955	3,061	606,105	750,526	0.019	1.44	0.69	6.0	873	4.9
8	17	15,292	6,748	658,007	842,181	0.026	1.52	0.77	6.1	927	5.6
6	35	48,316	15,790	1,857,389	2,408,580	0.047	3.28	1.65	6.8	1958	6.6
4	107	335,388	72,233	4,378,777	5,910,153	0.14	6.27	3.30	6.9	3599	17
retail											
0.05	313	1,170	7,173	1,845,307	3,415,601	0.16	2.32	3.83	20	12	12
0.04	436	1,662	9,715	3,009,850	5,794,555	0.18	3.15	5.41	21	13	12
0.03	673	2,620	14,191	5,215,055	10,891,104	0.21	4.40	7.92	21	14	12
0.02	1,318	5,108	24,933	9,623,356	23,973,690	0.26	6.14	12.7	22	16	13
0.01	3,617	14,202	75,884	22,351,676	75,462,045	0.41	9.39	23.0	22	22	14
accidents											
30	0	0	33	68,676	129,827	0.11	26.2	82.3	109	17473	182
25	0	0	58	ML	338,536	0.22	ML	194	146	ML	192
20	0	0	444	–	988,783	2.12	–	525	169	–	213
15	1	–	1,899	–	TL	2.92	–	TL	210	–	TL
10	212	–	20,497	–	–	7.49	–	–	240	–	–
kosarak											
1	7	31	116	223	298	0.28	0.10	0.12	106	81	81
0.8	12	48	149	336	453	0.32	0.13	0.15	110	82	82
0.6	18	71	231	549	900	0.40	0.15	0.21	115	83	83
0.4	24	117	422	1,219	2,573	0.53	0.24	0.40	123	84	84
0.2	55	265	1,284	6,337	17,318	0.77	0.57	1.31	137	88	88
T10I4D100K											
0.1	255	4,363	8,957	320,113	351,064	0.23	1.75	1.80	19	18	13
0.08	326	7,872	11,666	345,367	398,744	0.24	1.79	1.85	19	19	13
0.06	480	12,640	16,504	408,175	520,949	0.27	1.91	1.98	19	22	13
0.04	770	20,895	27,610	592,872	877,066	0.33	2.00	2.25	19	25	13
0.02	1,844	38,588	74,550	1,865,219	2,979,509	0.52	2.44	2.96	19	31	13

P3 one rule is disabled. *Closed* enumerates only closed itemsets, by prefix-preserving closure extension [19], with the same rules; it is the version of the algorithm that we first designed, motivated by Theorem 3.

- *Bounds*. Without *P4*, replacing *B* by *W* multiplies the number of visited nodes by a factor between 1.03–2.35, and replacing it by *UBO* by a factor between 1.26–1,088; the largest factors occur on **accidents**. With *UBO* the runtime grows by a factor of 10 on **mushroom** and 2.1 on **accidents**, and by at most 21% on the other datasets; with *W* it changes by at most 12%, the largest increase being on **accidents**.
- *Child bound*. Disabling *P4* multiplies the runtime by 1.29 to 3.02. The itemsets that *P4* rejects are the children at which *P1* would apply, so *P4* does not change the number of evaluated itemsets; it avoids building their conditional databases.

Table 6 Ablation at the middle threshold of each dataset (ξ in %): visited nodes (first line) and runtime (second line). no P4, W , UBO: rule P4 disabled and rule P1 with the bound B , W or UBO; no P2, no P3: rule disabled; closed: enumeration of closed itemsets only.

Dataset	ξ	MHOI-Miner	no P4	W	UBO	no P2	no P3	closed
mushroom	8	1,868 0.026 s	6,748 0.034 s	8,331 0.036 s	571,876 0.35 s	2,016 0.026 s	1,889 0.026 s	395 0.062 s
retail	0.03	4,206 0.20 s	14,191 0.25 s	16,409 0.27 s	29,147 0.31 s	4,798 0.20 s	4,206 0.20 s	4,199 0.22 s
accidents	20	23 2.22 s	444 6.71 s	1,045 7.53 s	482,973 14.0 s	23 2.17 s	23 2.18 s	23 2.61 s
kosarak	0.6	60 0.40 s	231 0.68 s	274 0.66 s	329 0.67 s	75 0.40 s	60 0.40 s	60 0.51 s
T10I4D100K	0.06	3,727 0.27 s	16,504 0.36 s	16,966 0.36 s	20,814 0.37 s	6,303 0.28 s	3,767 0.27 s	3,726 0.30 s

- *Look-ahead and subsumption.* Disabling P2 increases the number of nodes by up to a factor of 1.69 (T10I4D100K) and changes the runtime by at most 4%; disabling P3 changes the number of nodes by at most 1% and the runtime by at most 2%.
- *Closed itemsets.* The closed version visits between 0.21 and 1.00 times as many nodes as MHOI-Miner, fewest on **mushroom**, but takes between 1.11 and 2.40 times its runtime: computing closures and testing prefix preservation at every node costs more than it saves on these datasets.

8 Case study: retail baskets

To compare MHOIs with maximal frequent itemsets on real data, we use the Online Retail dataset [42, 43], which contains the transactions of a United Kingdom non-store online retailer of gifts between 1 December 2010 and 9 December 2011; many of its customers are wholesalers. We grouped the lines of each invoice into one transaction of distinct product codes, after removing cancelled invoices, lines with a non-positive quantity and non-product codes such as postage and manual adjustments; the script is part of the code. The resulting database has 20,538 transactions and 3,926 products; transactions have a median length of 15, a mean of 25.2 and a maximum of 1,108.

With $\xi = 0.2\%$ of the transactions ($\xi = 41.08$), there are 66 HOIs and 44 MHOIs; the support threshold of Lemma 1 is $\sigma = 42$, at which there are 888,052 MFIs. None of the MFIs is an HOI and none of the MHOIs is an MFI, so keeping the MFIs that are HOIs (Theorem 4) returns nothing here. The two families describe different itemsets. The median MFI has 5 products, support 43 and occupancy 0.6; it lies in large orders (median length 508 of its supporting invoices), where it covers a small fraction of each. The median MHOI has 1 product, support 916 and occupancy 48.0, and its supporting invoices have a median length of 34.

Of the MHOIs, 33 are single products and 11 have two or more (at most 4). Table 7 lists all MHOIs with two or more products and, for comparison, the single-product MHOIs with the highest occupancy and the MFIs with the highest support. That most MHOIs are single products is a property of occupancy as defined by Deng rather

Table 7 Online Retail, $\xi = 0.2\%$ of the transactions. Size: number of products; Support: number of invoices; Length: median number of distinct products of these invoices; occ: occupancy (Definition 1).

Products	Size	Support	Length	occ
<i>All MHOIs with two or more products</i>				
jumbo bag pink polkadot; jumbo bag red retrospot	2	825	39	61.7
alarm clock bakelike green; alarm clock bakelike red	2	640	35	55.2
red hanging heart t-light holder; white hanging heart t-light holder	2	489	40	51.3
jumbo bag red retrospot; jumbo bag strawberry	2	537	37	49.3
jumbo bag red retrospot; jumbo storage bag suki	2	724	45	47.8
jumbo bag baroque black white; jumbo bag red retrospot	2	585	46	46.0
green regency teacup and saucer; pink regency teacup and saucer;	4	301	32	45.6
regency cakestand 3 tier; roses regency teacup and saucer				
wooden frame antique white; wooden picture frame white finish	2	505	32	42.7
paper chain kit 50's christmas; paper chain kit vintage christmas	2	552	41	42.4
jumbo bag red retrospot; jumbo shopper vintage red paisley	2	680	44	41.5
jumbo bag red retrospot; jumbo bag vintage doily	2	577	34	41.1
<i>Single-product MHOIs with the highest occupancy</i>				
party bunting	1	1686	29	108.0
rabbit night light	1	1007	27	105.4
assorted colour bird ornament	1	1455	25	85.1
<i>MFIs with the highest support</i>				
set of salt and pepper toadstools	1	199	32	8.0
polyester filler pad 40x40cm	1	190	21	22.1
12 pencils tall tube posy	1	172	32	8.5

than of the algorithm: occupancy grows with support, so at this threshold the itemsets that reach it are mainly products that appear in many invoices, together with a few sets of products that are bought together often enough. All multi-product MHOIs in Table 7 consist of variants of one product line (jumbo bags, alarm clocks, hanging-heart T-light holders, Regency teacups with the matching cake stand, picture frames, paper chain kits).

9 Limitations

- MHOIs are not a lossless representation of HOIs: since occupancy is not anti-monotone, the HOIs cannot be recovered from the MHOIs without evaluating the occupancy of their subsets.
- The measure is the occupancy of Deng, which grows with support. In the case study most MHOIs are single products with high support. Users who want itemsets that cover a large fraction of each supporting transaction regardless of support may prefer the two-threshold formulation of Tang et al. [2], which this paper does not address.
- Enumerating only closed itemsets, which Theorem 3 suggests, did not pay off in our implementation (Section 7.5); a cheaper closure computation might change this. MHOI-Miner spends more time per node than HEP and DFHOI. It was slower on **kosarak**, where these algorithms enumerate few itemsets, and it used more memory than DFHOI on the sparse datasets.
- The comparison uses our C implementations of HEP and DFHOI, not the authors' code; their HOI counts agree with those published by Deng on three of the datasets. We used five of the six datasets of Deng, the SPMF version of **mushroom**, and one

machine. Runtimes depend on the implementations and the machine; visited nodes do not depend on the machine.

- Subsumption checks (rule P3 and the final filter) scan the recorded itemsets linearly. This was not a bottleneck in our experiments, where at most a few thousand itemsets were recorded, but it would be with many more MHOIs.

10 Conclusion

We studied the maximal high occupancy itemsets, for the occupancy measure of Deng. Every MHOI is a frequent closed itemset at the support threshold $\lceil \xi \rceil$; maximal frequent itemsets that are HOIs are MHOIs, but MHOIs need not be maximal frequent itemsets, also when all transactions have the same length, so they cannot be obtained by filtering maximal frequent itemsets. When all transactions have the same length, the bound UBO of Deng reduces to the support. We proposed a bound that fixes the size of a descendant across its supporting transactions, proved that it never exceeds UBO nor a per-transaction bound, and used it in MHOI-Miner, a depth-first search of the frequent itemsets with a child bound, head-union-tail look-ahead and subsumption, which we proved correct and verified against an exact oracle. On the datasets and thresholds of Deng, MHOI-Miner evaluated a median of 60.6 and 125 times fewer itemsets than HEP and DFHOI followed by maximality filtering, was faster in most configurations, was the only method that completed all of them, and was slower on *kosarak*. Reducing the cost per node, for example by avoiding the construction of conditional databases for children that are subsumed, and indexing the recorded itemsets for the subsumption checks, are directions for further work.

Acknowledgements. Not applicable.

Declarations

- **Funding:** Funding not applicable. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.
- **Competing interests:** The author declares no competing interests.
- **Ethics approval and consent to participate:** Not applicable.
- **Consent for publication:** Not applicable.
- **Data availability:** The benchmark datasets are publicly available from the SPMF library [40, 41]; *retail*, *accidents*, *kosarak* and *T10I4D100K* are also distributed by the FIMI repository [39, 44]. The Online Retail dataset is available from the UCI Machine Learning Repository [42] under the CC BY 4.0 licence.
- **Materials availability:** Not applicable.
- **Code availability:** The source code of all methods and the scripts that reproduce every experiment are available from the author on reasonable request.
- **Author contribution:** Q. Van conceived the method, proved the results, implemented the algorithms, performed the experiments and wrote the manuscript.

References

- [1] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216. ACM, New York, NY, USA (1993). <https://doi.org/10.1145/170035.170072>
- [2] Tang, L., Zhang, L., Luo, P., Wang, M.: Incorporating occupancy into frequent pattern mining for high quality pattern recommendation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM), pp. 75–84. ACM, New York, NY, USA (2012). <https://doi.org/10.1145/2396761.2396775>
- [3] Zhang, L., Luo, P., Tang, L., Chen, E., Liu, Q., Wang, M., Xiong, H.: Occupancy-based frequent pattern mining. *ACM Transactions on Knowledge Discovery from Data* **10**(2), 1–33 (2015) <https://doi.org/10.1145/2753765>
- [4] Deng, Z.-H.: Mining high occupancy itemsets. *Future Generation Computer Systems* **102**, 222–229 (2020) <https://doi.org/10.1016/j.future.2019.07.039>
- [5] Nguyen, L.T.T., Mai, T., Pham, G.-H., Yun, U., Vo, B.: An efficient method for mining high occupancy itemsets based on equivalence class and early pruning. *Knowledge-Based Systems* **267**, 110441 (2023) <https://doi.org/10.1016/j.knosys.2023.110441>
- [6] Datta, S., Mali, K., Ghosh, U.: High occupancy itemset mining with consideration of transaction occupancy. *Arabian Journal for Science and Engineering* **47**(2), 2061–2075 (2022) <https://doi.org/10.1007/s13369-021-06075-8>
- [7] Kim, H., Ryu, T., Lee, C., Kim, H., Truong, T., Fournier-Viger, P., Pedrycz, W., Yun, U.: Mining high occupancy patterns to analyze incremental data in intelligent systems. *ISA Transactions* **131**, 460–475 (2022) <https://doi.org/10.1016/j.isatra.2022.05.003>
- [8] Tran, T.-N., Truong Hoang, V., Truong, T.-C., Voznak, M.: A hierarchical set-enumeration tree enabling high occupancy item set mining and the use of an adaptive occupancy threshold. *Applied Intelligence* **55**(3), 205 (2025) <https://doi.org/10.1007/s10489-024-06166-7>
- [9] Yıldırım, İ.: Mining top-k high occupancy itemsets. *Black Sea Journal of Engineering and Science* **8**(6), 1723–1730 (2025) <https://doi.org/10.34248/bsengineering.1744061>
- [10] Ngo, T.-K., Nguyen, H.-S., Pedrycz, W., Vo, B.: Efficient algorithms for mining top-k high occupancy itemsets. *Information Sciences* **741**, 123226 (2026) <https://doi.org/10.1016/j.ins.2026.123226>

- [11] Cho, M., Kim, H., Kim, H., Ryu, T., Lee, C., Kim, H., Vo, B., Lin, J.C.-W., Yun, U.: Damped window based high occupancy pattern mining with one scanning of data streams. *Engineering Applications of Artificial Intelligence* **174**, 114511 (2026) <https://doi.org/10.1016/j.engappai.2026.114511>
- [12] Bayardo, R.J.: Efficiently mining long patterns from databases. *ACM SIGMOD Record* **27**(2), 85–93 (1998) <https://doi.org/10.1145/276305.276313>
- [13] Seno, M., Karypis, G.: Finding frequent patterns using length-decreasing support constraints. *Data Mining and Knowledge Discovery* **10**(3), 197–228 (2005) <https://doi.org/10.1007/s10618-005-0364-0>
- [14] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pp. 487–499 (1994). <https://www.vldb.org/conf/1994/P487.PDF>
- [15] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 1–12. ACM, New York, NY, USA (2000). <https://doi.org/10.1145/342009.335372>
- [16] Zaki, M.J.: Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering* **12**(3), 372–390 (2000) <https://doi.org/10.1109/69.846291>
- [17] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: *Database Theory – ICDT’99. Lecture Notes in Computer Science*, vol. 1540, pp. 398–416. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-49257-7_25
- [18] Zaki, M.J., Hsiao, C.-J.: CHARM: An efficient algorithm for closed itemset mining. In: *Proceedings of the 2002 SIAM International Conference on Data Mining*, pp. 457–473. SIAM, Philadelphia, PA, USA (2002). <https://doi.org/10.1137/1.9781611972726.27>
- [19] Uno, T., Kiyomi, M., Arimura, H.: LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI’04)*. *CEUR Workshop Proceedings*, vol. 126 (2004). <https://ceur-ws.org/Vol-126/uno.pdf>
- [20] Uno, T., Kiyomi, M., Arimura, H.: LCM ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In: *Proceedings of the 1st International Workshop on Open Source Data Mining (OSDM’05)*, pp. 77–86. ACM, New York, NY, USA (2005). <https://doi.org/10.1145/1133905.1133916>
- [21] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., Yiu, T.: MAFIA: A maximal frequent itemset algorithm. *IEEE Transactions on Knowledge and Data*

- Engineering **17**(11), 1490–1504 (2005) <https://doi.org/10.1109/TKDE.2005.183>
- [22] Gouda, K., Zaki, M.J.: GenMax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery* **11**(3), 223–242 (2005) <https://doi.org/10.1007/s10618-005-0002-x>
 - [23] Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-trees. *IEEE Transactions on Knowledge and Data Engineering* **17**(10), 1347–1362 (2005) <https://doi.org/10.1109/TKDE.2005.166>
 - [24] Pei, J., Han, J., Lakshmanan, L.V.S.: Pushing convertible constraints in frequent itemset mining. *Data Mining and Knowledge Discovery* **8**(3), 227–252 (2004) <https://doi.org/10.1023/B:DAMI.0000023674.74932.4c>
 - [25] Bonchi, F., Lucchese, C.: On closed constrained frequent pattern mining. In: *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 35–42. IEEE, Los Alamitos, CA, USA (2004). <https://doi.org/10.1109/ICDM.2004.10093>
 - [26] Aggarwal, C.C., Han, J. (eds.): *Frequent Pattern Mining*. Springer, Cham (2014). <https://doi.org/10.1007/978-3-319-07821-2>
 - [27] Luna, J.M., Fournier-Viger, P., Ventura, S.: Frequent itemset mining: A 25 years review. *WIREs Data Mining and Knowledge Discovery* **9**(6), 1329 (2019) <https://doi.org/10.1002/widm.1329>
 - [28] Shen, B., Wen, Z., Zhao, Y., Zhou, D., Zheng, W.: OCEAN: Fast discovery of high utility occupancy itemsets. In: *Advances in Knowledge Discovery and Data Mining (PAKDD 2016)*. *Lecture Notes in Computer Science*, vol. 9651, pp. 354–365. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31753-3_29
 - [29] Gan, W., Lin, J.C.-W., Fournier-Viger, P., Chao, H.-C., Yu, P.S.: HUOPM: High-utility occupancy pattern mining. *IEEE Transactions on Cybernetics* **50**(3), 1195–1208 (2020) <https://doi.org/10.1109/TCYB.2019.2896267>
 - [30] Duong, H., Pham, H., Truong, T., Fournier-Viger, P.: Efficient algorithms to mine concise representations of frequent high utility occupancy patterns. *Applied Intelligence* **54**(5), 4012–4042 (2024) <https://doi.org/10.1007/s10489-024-05296-2>
 - [31] Hoang, T., Huynh, L., Duong, H., Truong, T.: Efficient mining of concise patterns for frequent high-utility occupancy itemsets using extended pruning strategies. *Journal on Information Technologies & Communications* **2025**(2), 108–128 (2025) <https://doi.org/10.32913/mic-ict-research.v2025.n2.1360>
 - [32] Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 55–64. ACM, New York, NY, USA (2012).

<https://doi.org/10.1145/2396761.2396773>

- [33] Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S.: EFIM: A fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems* **51**(2), 595–625 (2017) <https://doi.org/10.1007/s10115-016-0986-0>
- [34] Gan, W., Lin, J.C.-W., Fournier-Viger, P., Chao, H.-C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering* **33**(4), 1306–1327 (2021) <https://doi.org/10.1109/TKDE.2019.2942594>
- [35] The Linux Kernel Documentation: The /proc Filesystem. <https://www.kernel.org/doc/html/latest/filesystems/proc.html>. Accessed 15 October 2026
- [36] Mushroom. UCI Machine Learning Repository (1981). <https://doi.org/10.24432/C5959T>
- [37] Brijs, T., Swinnen, G., Vanhoof, K., Wets, G.: Using association rules for product assortment decisions. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 254–260. ACM, New York, NY, USA (1999). <https://doi.org/10.1145/312129.312241>
- [38] Geurts, K., Wets, G., Brijs, T., Vanhoof, K.: Profiling of high-frequency accident locations by use of association rules. *Transportation Research Record* **1840**(1), 123–130 (2003) <https://doi.org/10.3141/1840-14>
- [39] Goethals, B., Zaki, M.J.: Frequent Itemset Mining Dataset Repository. <http://fimi.uantwerpen.be/>. Accessed 15 October 2026
- [40] Fournier-Viger, P.: SPMF: An Open-Source Data Mining Library – Datasets. <https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php>. Accessed 15 October 2026
- [41] Fournier-Viger, P., Lin, J.C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The SPMF open-source data mining library version 2. In: *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2016)*. Lecture Notes in Computer Science, vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
- [42] Chen, D.: Online Retail. UCI Machine Learning Repository (2015). <https://doi.org/10.24432/C5BW33>
- [43] Chen, D., Sain, S.L., Guo, K.: Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management* **19**(3), 197–208 (2012) <https://doi.org/10.1057/dbm.2012.17>

- [44] Goethals, B., Zaki, M.J.: Advances in frequent itemset mining implementations: report on FIMI'03. ACM SIGKDD Explorations Newsletter **6**(1), 109–117 (2004)
<https://doi.org/10.1145/1007730.1007744>

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [xicounts.pdf](#)
- [xiresults.pdf](#)
- [ablation.tex](#)
- [datasets.tex](#)
- [results.tex](#)