MHOUI-Miner: An Exact Method for Mining Maximal High-Occupancy Utility Itemsets

Quan Van
Independent Researcher
vanhaminhquan2406@gmail.com

Abstract

Frequent itemset mining discovers itemsets that occur frequently, high-occupancy itemset mining discovers itemsets that cover a large fraction of their supporting transactions, and high-utility itemset mining discovers itemsets with high economic or semantic value. Existing approaches usually optimize these objectives separately, or combine utility with utility-occupancy, where occupancy is defined as the ratio between itemset utility and transaction utility. This paper proposes a new itemset pattern type called the *Maximal High-Occupancy Utility Itemset* (MHOUI). In contrast to utility-occupancy mining, MHOUI uses classical transaction occupancy, defined as $|X|/|T|$, and combines it with total utility $u(X)$ as an independent constraint. An itemset is first considered a High-Occupancy Utility Itemset (HOUI) if it satisfies minimum support, minimum average classical occupancy, and minimum total utility. A weak MHOUI is a HOUI that is not Pareto-dominated by any strict HOUI superset with respect to average occupancy and total utility. A strong MHOUI further removes an itemset if a strict HOUI superset has equal or higher occupancy and equal or higher utility. We formalize the MHOUI problem, prove key theoretical properties, derive safe support, utility, and occupancy upper bounds, and propose MHOUI-Miner, an exact hybrid vertical algorithm using bitsets and Bitset-Occupancy-Utility lists. The proposed framework provides a compact, interpretable, and utility-aware representation of dense transaction patterns.

Keywords: High-utility itemset mining; high-occupancy itemset mining; maximal pattern mining; utility mining; Pareto dominance; concise pattern representation; vertical bitset mining.

1 Introduction

Frequent itemset mining is a foundational task in data mining. Given a transaction database, its goal is to discover all itemsets whose support is no less than a user-defined minimum support threshold. The Apriori algorithm introduced the classical support anti-monotonicity principle for association rule mining [1]. FP-Growth improved efficiency by using FP-trees to avoid candidate generation [2]. Eclat popularized the vertical transaction-id representation, where support can be computed by intersecting transaction-id sets [3].

However, frequency alone is often insufficient. A frequent pattern may occur in many transactions but represent only a small portion of each transaction. To reduce redundancy, maximal frequent itemset mining reports only frequent itemsets with no frequent supersets. Representative algorithms include MAFIA, GenMax, and FPmax* [4, 5, 6]. Although maximality compresses frequent itemset outputs, support-based maximality does not consider transaction coverage or utility.

Occupancy-based pattern mining addresses the coverage limitation by measuring how much of a supporting transaction is occupied by an itemset. Classical occupancy is usually defined as the ratio between itemset length and transaction length, namely $|X|/|T|$. Occupancy-based frequent pattern mining and high-occupancy itemset mining have shown that occupancy can improve pattern representativeness [7, 8, 9]. However, high-occupancy itemset mining does not directly consider item utilities such as profit, quantity, risk, or importance.

High-utility itemset mining (HUIM) addresses the utility limitation by discovering itemsets whose total utility is no less than a user-defined minimum utility threshold. Two-Phase introduced transaction-weighted utilization as an anti-monotone upper bound [12]. HUI-Miner introduced utility lists for one-phase high-utility itemset mining [13]. FHM and EFIM further improved pruning and scalability [14, 15]. Recent work has also studied concise representations such as closed and maximal high-utility itemsets [18].

A related but different direction is high utility-occupancy pattern mining. HUOPM defines utility occupancy as $u(X,T)/TU(T)$ and mines patterns satisfying frequency, utility, and utility-occupancy constraints [16]. HUOPM+ and later variants improve pruning and representation quality [17]. These methods are important relatives, but their occupancy concept is based on utility ratio rather than classical itemset coverage $|X|/|T|$.

This paper proposes a new itemset pattern family called *Maximal High-Occupancy Utility Itemsets* (MHOUIs). The proposed pattern type combines three independent requirements:

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

(2) **Classical occupancy**: the itemset must occupy a large fraction of its supporting transactions.

(3) **Total utility**: the itemset must have high total utility in the database.

After satisfying these constraints, a pattern is retained only if it is not dominated by a strict high-occupancy utility superset. The maximality used in MHOUI is therefore not ordinary support-maximality. It is a superset-based Pareto maximality criterion over classical occupancy and total utility.

The main contributions of this paper are:

- We introduce the High-Occupancy Utility Itemset (HOUI) and Maximal High-Occupancy Utility Itemset (MHOUI) definitions.
- We define weak and strong MHOUI variants using strict-superset Pareto dominance.
- We prove key theoretical properties, including support anti-monotonicity, TWU anti-monotonicity, utility non-monotonicity, and occupancy non-monotonicity.
- We derive safe utility and occupancy upper bounds for exact pruning.
- We propose MHOUI-Miner, a hybrid vertical algorithm using bitsets and Bitset-Occupancy-Utility lists.
- We provide a Q1-style experimental protocol for comparing MHOUI-Miner against frequent, maximal, occupancy-based, high-utility, and utility-occupancy baselines.

2 Related Work

2.1 Frequent and Maximal Frequent Itemset Mining

Apriori introduced the support-based framework for mining frequent itemsets and association rules [1]. Its main pruning principle is that if an itemset is infrequent, all of its supersets are also infrequent. FP-Growth avoids candidate generation using FP-trees [2], while Eclat mines frequent itemsets using vertical transaction-id sets [3].

Maximal frequent itemset mining compresses the output by reporting only frequent itemsets without frequent supersets. MAFIA uses bitmap-based depth-first search and maximality pruning [4]. GenMax improves maximality checking through progressive focusing [5]. FPmax* mines maximal frequent itemsets using FP-tree-based structures [6]. These methods motivate the use of maximality for output compression, but they do not consider occupancy or utility.

2.2 High-Occupancy Itemset Mining

Occupancy-based pattern mining evaluates how representative an itemset is within its supporting transactions. Tang et al. introduced occupancy into pattern mining for high-quality pattern recommendation [7]. Zhang et al. developed occupancy-based frequent pattern mining [8]. Deng proposed high-occupancy itemset mining to discover itemsets whose average occupancy is above a threshold [9]. Later work studied transaction occupancy and early pruning strategies [10, 11].

MHOUI differs from high-occupancy itemset mining because it also requires high total utility and applies superset Pareto maximality.

2.3 High-Utility Itemset Mining

High-utility itemset mining discovers itemsets with high total utility. The Two-Phase algorithm introduced transaction-weighted utilization (TWU), a safe anti-monotone upper bound for utility pruning [12]. HUI-Miner introduced utility lists to mine high-utility itemsets without candidate generation [13]. FHM improves pruning using estimated utility co-occurrence [14]. EFIM uses database projection, transaction merging, and tighter utility bounds for efficient mining [15]. Bit-based high-utility itemset mining has also been studied to improve efficiency on dense databases [20].

MHOUI builds on HUIM ideas but adds classical occupancy and dominance-based maximality.

2.4 High Utility-Occupancy Pattern Mining

HUOPM mines high utility-occupancy patterns, where utility occupancy is defined as the fraction of transaction utility covered by the itemset, i.e.,

$$uo(X,T) = \frac{u(X,T)}{TU(T)}.$$

Gan et al. proposed HUOPM using utility-occupancy lists and a frequency-utility tree [16]. HUOPM+ introduced length-aware pruning for flexible high utility-occupancy pattern mining [17]. Recent work has also studied concise representations of frequent high-utility occupancy patterns [19].

MHOUI is different from HUOPM-style mining because it treats classical occupancy $|X|/|T|$ and total utility $u(X)$ as separate objectives. Thus, a MHOUI pattern is both transaction-covering and utility-significant.

2.5 Positioning of MHOUI

Table 1 summarizes the relationship between MHOUI and related pattern families.

Table 1: Positioning of MHOUI against related itemset mining families.

Pattern Family	Support	Classical Occupancy	Total Utility	Utility Occupancy	Maximality
FI	✓	–	–	–	–
MFI	✓	–	–	–	Support-maximal
HOI	✓	✓	–	–	–
HUI	Optional	–	✓	–	–
MaxHUI	Optional	–	✓	–	Utility-based concise form
HUOP / FHUOI	✓	–	Indirect	✓	Optional concise form
HOUI	✓	✓	✓	–	–
Weak MHOUI	✓	✓	✓	–	Superset Pareto dominance
Strong MHOUI	✓	✓	✓	–	Equality-inclusive superset dominance

To the best of our survey, no prior work mines exactly the support-constrained, classical-occupancy-plus-total-utility, superset-maximal Pareto target formalized in this paper. This claim should be interpreted carefully: MHOUI is close to HOI, HUIM, HUOPM, FHUOI, and MaxHUI, but differs in its simultaneous use of classical occupancy, total utility, support, and strict-superset dominance.

3 Preliminaries

Let

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

be a finite set of items. A quantitative transaction database is denoted by

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

Each transaction T_q contains a set of items. For each item $i \in T_q$, let $q(i, T_q)$ denote its internal utility, such as quantity. Let $p(i)$ denote the external utility of item i, such as unit profit.

Definition 1 (Item Utility)**.** *The utility of item i in transaction T_q is*

$$u(i, T_q) = p(i) \cdot q(i, T_q).$$

Definition 2 (Itemset Utility in a Transaction)**.** *For an itemset $X \subseteq T_q$, the utility of X in transaction T_q is*

$$u(X, T_q) = \sum_{i \in X} u(i, T_q).$$

Definition 3 (Transaction Utility)**.** *The transaction utility of T_q is*

$$TU(T_q) = \sum_{i \in T_q} u(i, T_q).$$

Definition 4 (Supporting Transaction Set)**.** *The supporting transaction set of itemset X is*

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

Definition 5 (Support)**.** *The support count of X is*

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

Definition 6 (Database Utility)**.** *The total utility of itemset X in the database is*

$$u(X) = \sum_{T_q \in \Gamma(X)} u(X, T_q).$$

Definition 7 (Classical Transaction Occupancy)**.** *For $X \subseteq T_q$, the classical occupancy of X in T_q is*

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

Definition 8 (Average Classical Occupancy)**.** *The average classical occupancy of X is*

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

Definition 9 (Transaction-Weighted Utilization)**.** *The transaction-weighted utilization of itemset X is*

$$TWU(X) = \sum_{T_q \in \Gamma(X)} TU(T_q).$$

4 Maximal High-Occupancy Utility Itemsets

4.1 High-Occupancy Utility Itemsets

Definition 10 (High-Occupancy Utility Itemset)**.** *Given a quantitative transaction database $\mathcal{D}$, a minimum support threshold σ, a minimum occupancy threshold β, and a minimum utility threshold μ, an itemset X is a High-Occupancy Utility Itemset (HOUI) if*

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

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

and

$$u(X) \geq \mu.$$

The complete set of HOUIs is

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

4.2 Superset Pareto Dominance

Definition 11 (Weak Superset Dominance)**.** *For two itemsets X and Y, Y weakly dominates X, denoted by $X \prec_w Y$, if*

$$X \subset Y,$$

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

$$u(Y) \geq u(X),$$

and at least one of the following is true:

$$\mathrm{aocc}(Y) > \mathrm{aocc}(X) \quad \textit{or} \quad u(Y) > u(X).$$

Definition 12 (Strong Superset Dominance)**.** *For two itemsets X and Y, Y strongly dominates X, denoted by $X \prec_s Y$, if*

$$X \subset Y,$$

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

and

$$u(Y) \geq u(X).$$

In strong dominance, equality in both objective values still counts as domination because Y is a strict superset of X.

4.3 Weak and Strong MHOUI

Definition 13 (Weak MHOUI)**.** *An itemset X is a weak Maximal High-Occupancy Utility Itemset if*

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

and there does not exist $Y \in \mathcal{HOUI}(\mathcal{D}, \sigma, \beta, \mu)$ such that

$$X \prec_w Y.$$

The weak MHOUI set is

$$\mathcal{MHOUI}^w(\mathcal{D}, \sigma, \beta, \mu) = \{X \in \mathcal{HOUI} \mid \nexists Y \in \mathcal{HOUI} : X \prec_w Y\}.$$

Definition 14 (Strong MHOUI)**.** *An itemset X is a strong Maximal High-Occupancy Utility Itemset if*

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

and there does not exist $Y \in \mathcal{HOUI}(\mathcal{D}, \sigma, \beta, \mu)$ such that

$$X \prec_s Y.$$

The strong MHOUI set is

$$\mathcal{MHOUI}^s(\mathcal{D}, \sigma, \beta, \mu) = \{X \in \mathcal{HOUI} \mid \nexists Y \in \mathcal{HOUI} : X \prec_s Y\}.$$

Theorem 1 (Inclusion Chain)**.** *For any database $\mathcal{D}$, support threshold σ, occupancy threshold β, and utility threshold μ,*

$$\mathcal{MHOUI}^s(\mathcal{D}, \sigma, \beta, \mu) \subseteq \mathcal{MHOUI}^w(\mathcal{D}, \sigma, \beta, \mu) \subseteq \mathcal{HOUI}(\mathcal{D}, \sigma, \beta, \mu).$$

Proof. Every weak or strong MHOUI must first be a HOUI by definition. Strong dominance is more permissive than weak dominance because it also treats the equal-occupancy and equal-utility case as domination when the dominator is a strict superset. Thus any itemset removed by weak dominance is also removed by strong dominance. Therefore,

$$\mathcal{MHOUI}^s \subseteq \mathcal{MHOUI}^w \subseteq \mathcal{HOUI}.$$

□

5 Theoretical Properties

5.1 Support Anti-Monotonicity

Theorem 2 (Support Anti-Monotonicity)**.** *If $X \subseteq Y$, then*

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

Proof. Every transaction containing Y also contains X. Hence,

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

Taking cardinalities gives

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

Therefore,

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

□

Property 1 (Safe Support Pruning)**.** *If $\sup(X) < \sigma$, then no superset of X can be a HOUI. Thus the branch rooted at X can be safely pruned.*

5.2 TWU Anti-Monotonicity

Theorem 3 (TWU Anti-Monotonicity)**.** *If $X \subseteq Y$, then*

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

Proof. Since $X \subseteq Y$,

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

Because transaction utilities are nonnegative,

$$TWU(Y) = \sum_{T_q \in \Gamma(Y)} TU(T_q) \leq \sum_{T_q \in \Gamma(X)} TU(T_q) = TWU(X).$$

□

Corollary 1 (Safe TWU Pruning)**.** *If $TWU(X) < \mu$, then no superset of X can satisfy the minimum utility threshold μ. Therefore, the branch rooted at X can be safely pruned.*

5.3 Utility Non-Monotonicity

Theorem 4 (Utility Non-Monotonicity)**.** *Exact utility $u(X)$ is neither monotone nor anti-monotone on the itemset lattice.*

Proof. Anti-monotonicity fails because adding an item may increase total utility. Let

$$T_1 = \{a, b\}, \quad T_2 = \{a\}.$$

Suppose

$$u(a, T_1) = 1, \quad u(b, T_1) = 100, \quad u(a, T_2) = 1.$$

Then

$$u(\{a\}) = 2,$$

but

$$u(\{a, b\}) = 101.$$

Thus, a superset can have larger utility.

Monotonicity also fails because extending an itemset may reduce its support. Let

$$T_1 = \{a, b\}, \quad T_2 = \{a\},$$

with

$$u(a, T_1) = 100, \quad u(b, T_1) = 1, \quad u(a, T_2) = 100.$$

Then

$$u(\{a\}) = 200,$$

but

$$u(\{a, b\}) = 101.$$

Thus, a superset can have smaller utility. □

5.4 Occupancy Non-Monotonicity

Theorem 5 (Average Occupancy Non-Monotonicity)**.** *Average classical occupancy $\mathrm{aocc}(X)$ is neither monotone nor anti-monotone on the itemset lattice.*

Proof. Consider the transaction 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 $Y = \{a, b\}$, $X \subset Y$, and Y occurs only in T_1. Therefore,

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

Thus a superset can have larger average occupancy, so anti-monotonicity fails.

Conversely, extending an itemset can remove short supporting transactions and preserve longer ones, which may reduce the average occupancy. Therefore, monotonicity also fails. □

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

6 Safe Upper Bounds

Let a search node be denoted as

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

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

$$Y = P \cup Z,$$

where $Z \subseteq E$.

For each transaction $T \in \Gamma(P)$, define the remaining extension count

$$rc_N(T) = |E \cap T|,$$

and remaining extension utility

$$ru_N(T) = \sum_{i \in E \cap T} u(i, T).$$

6.1 Remaining-Utility Upper Bound

Definition 15 (Remaining-Utility Upper Bound)**.** *For a node* $N = (P, E)$*, define*

$$UUB(N) = \sum_{T \in \Gamma(P)} \left(u(P, T) + ru_N(T)\right).$$

Theorem 6 (Remaining-Utility Upper Bound)**.** *For every descendant* $Y = P \cup Z$ *of node* N*,*

$$u(Y) \leq UUB(N).$$

Proof. For every transaction $T \in \Gamma(Y)$, we have $Z \subseteq E \cap T$. Therefore,

$$u(Y, T) = u(P, T) + \sum_{i \in Z} u(i, T) \leq u(P, T) + ru_N(T).$$

Since $\Gamma(Y) \subseteq \Gamma(P)$,

$$u(Y) = \sum_{T \in \Gamma(Y)} u(Y, T) \leq \sum_{T \in \Gamma(P)} \left(u(P, T) + ru_N(T)\right) = UUB(N).$$

□

Property 3 (Safe Utility Pruning)**.** *If*

$$UUB(N) < \mu,$$

then no descendant of N *can be a HOUI.*

6.2 Occupancy Size Bound

Let

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

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

Theorem 7 (Support-Constrained Size Bound)**.** *For any descendant $Y = P \cup Z$ of N such that $\sup(Y) \geq \sigma$,*

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

Proof. Let $d = |Z|$. If Y has support at least σ, then at least σ transactions in $\Gamma(P)$ contain all items in Z. Hence $d \leq rc_N(T)$ for at least σ supporting transactions. Thus,

$$d \leq c_{(\sigma)}.$$

Therefore,

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

□

6.3 Occupancy Upper Bound 1

Let

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

Let

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

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

Definition 16 (Occupancy Upper Bound 1)**.**

$$OUB_1(N) = \big(|P| + c_{(\sigma)}\big) \cdot \frac{1}{\sigma} \sum_{j=1}^{\sigma} r_{(j)}.$$

Theorem 8. *For every descendant Y of N with $\sup(Y) \geq \sigma$,*

$$\mathrm{aocc}(Y) \leq OUB_1(N).$$

Proof. By the support-constrained size bound,

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

Also, $\Gamma(Y) \subseteq \Gamma(P)$ and $|\Gamma(Y)| \geq \sigma$. The maximum possible average of $1/|T|$ over any support set of size at least σ is bounded by the average of the top σ reciprocal transaction lengths in $\Gamma(P)$. Therefore,

$$\mathrm{aocc}(Y) \leq \big(|P| + c_{(\sigma)}\big) \cdot \frac{1}{\sigma} \sum_{j=1}^{\sigma} r_{(j)} = OUB_1(N).$$

□

6.4 Occupancy Upper Bound 2

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

$$v_N(T) = \frac{|P| + rc_N(T)}{|T|}.$$

Let

$$v_{(1)} \geq v_{(2)} \geq \cdots$$

be the sorted values of $v_N(T)$.

Definition 17 (Occupancy Upper Bound 2)**.**

$$OUB_2(N) = \frac{1}{\sigma} \sum_{j=1}^{\sigma} v_{(j)}.$$

Theorem 9. *For every descendant Y of N with $\sup(Y) \geq \sigma$,*

$$\mathrm{aocc}(Y) \leq OUB_2(N).$$

Proof. For every transaction $T \in \Gamma(Y)$, the extension part of Y is contained in $E \cap T$. Hence,

$$|Y| \leq |P| + rc_N(T).$$

Thus,

$$\mathrm{occ}(Y, T) = \frac{|Y|}{|T|} \leq \frac{|P| + rc_N(T)}{|T|} = v_N(T).$$

Since $\Gamma(Y) \subseteq \Gamma(P)$ and $|\Gamma(Y)| \geq \sigma$, the average occupancy of Y is bounded by the average of the top σ values of $v_N(T)$:

$$\mathrm{aocc}(Y) \leq OUB_2(N).$$

□

Property 4 (Safe Occupancy Pruning)**.** *If*

$$OUB_1(N) < \beta \quad \textit{or} \quad OUB_2(N) < \beta,$$

then no descendant of N can be a HOUI.

6.5 Branch Dominance Pruning

Theorem 10 (Strong Branch Dominance Pruning)**.** *Let M be a discovered HOUI. For a node $N = (P, E)$, if*

$$P \cup E \subset M,$$
$$\mathrm{aocc}(M) \geq OUB_2(N),$$

and

$$u(M) \geq UUB(N),$$

then every descendant of N is strongly dominated by M.

Proof. Every descendant Y of N satisfies

$$Y \subseteq P \cup E \subset M.$$

Thus M is a strict superset of Y. Moreover,

$$\mathrm{aocc}(Y) \leq OUB_2(N) \leq \mathrm{aocc}(M),$$

and

$$u(Y) \leq UUB(N) \leq u(M).$$

Therefore M strongly dominates every descendant Y of N. □

7 MHOUI-Miner Algorithm

7.1 Hybrid Vertical Representation

MHOUI-Miner uses a hybrid vertical representation. The first component is a packed bitset for each itemset:

$$B_X = \{tid(T) \mid X \subseteq T\}.$$

This bitset enables fast support computation using bitwise intersection and population count instructions.

The second component is a Bitset-Occupancy-Utility list, abbreviated as BOU-list.

Definition 18 (BOU-list Entry)**.** *For an itemset X, a BOU-list entry for transaction T is*

$$e = \langle tid, iu, ru, invlen, remcnt \rangle,$$

where:

- tid *is the transaction identifier.*
- $iu = u(X, T)$ *is the utility of X in T.*
- ru *is the remaining utility of extension items in T.*
- $invlen = 1/|T|$.
- $remcnt$ *is the number of remaining extension items in T.*

For an itemset X, the BOU-list L_X allows one-pass computation of:

$$\sup(X) = |L_X|,$$

$$u(X) = \sum_{e \in L_X} e.iu,$$

$$UUB(X) = \sum_{e \in L_X} (e.iu + e.ru),$$

and

$$\mathrm{aocc}(X) = |X| \cdot \frac{\sum_{e \in L_X} e.invlen}{|L_X|}.$$

7.2 Preprocessing

The preprocessing phase consists of:

(1) Read the quantitative transaction database.

(2) Compute transaction utilities $TU(T)$ and transaction lengths $|T|$.

(3) Compute singleton supports and singleton TWU values.

(4) Remove items with support below σ or TWU below μ.

(5) Sort remaining items by ascending TWU, ascending support, and item identifier.

(6) Reorder each transaction according to the global item order.

(7) Build singleton bitsets and singleton BOU-lists.

7.3 Two-Phase Exact Mining

The publication-safe version of MHOUI-Miner uses two phases:

(1) Mine all HOUIs exactly.

(2) Apply exact weak or strong dominance filtering.

The two-phase version is used as the correctness oracle for optimized direct mining.

Algorithm 1 MHOUI-Miner

Require: Quantitative database $\mathcal{D}$, minimum support σ, minimum occupancy β, minimum utility μ, variant $v \in \{weak, strong\}$
Ensure: Weak or strong MHOUI set
Build singleton bitsets and BOU-lists
Apply support and TWU pruning to remove unpromising 1-items
Sort remaining items by ascending TWU and support
$\mathcal{C}_H \leftarrow \emptyset$
DFS-HOUI$(\emptyset, B_\emptyset, L_\emptyset, E, \mathcal{C}_H)$
if $v = weak$ **then**

    return FILTER-WEAK$(\mathcal{C}_H)$
else

    return FILTER-STRONG$(\mathcal{C}_H)$
end if

Algorithm 2 DFS-HOUI

Require: Prefix X, bitset B_X, BOU-list L_X, suffix items E, output list $\mathcal{C}_H$

for each item $i \in E$ **do**
  $Y \leftarrow X \cup \{i\}$
  $B_Y \leftarrow B_X \cap B_i$
  $s \leftarrow popcount(B_Y)$
  if $s < \sigma$ **then**
    continue
  end if
  $L_Y \leftarrow$ JOIN-BOU(L_X, L_i)
  if $TWU(Y) < \mu$ **then**
    continue
  end if
  if $UUB(Y) < \mu$ **then**
    continue
  end if
  if $OUB_1(Y) < \beta$ or $OUB_2(Y) < \beta$ **then**
    continue
  end if
  $util_Y \leftarrow \sum_{e \in L_Y} e.iu$
  $aocc_Y \leftarrow |Y| \cdot \sum_{e \in L_Y} e.invlen / |L_Y|$
  if $util_Y \geq \mu$ and $aocc_Y \geq \beta$ **then**
    Add $(Y, s, util_Y, aocc_Y)$ to $\mathcal{C}_H$
  end if
  $E' \leftarrow$ items after i in E
  DFS-HOUI$(Y, B_Y, L_Y, E', \mathcal{C}_H)$
end for

Algorithm 3 FILTER-STRONG

Require: HOUI list $\mathcal{C}_H$

Ensure: Strong MHOUI set

Sort $\mathcal{C}_H$ by decreasing itemset length
$Processed \leftarrow \emptyset$
$Output \leftarrow \emptyset$
for each pattern $X \in \mathcal{C}_H$ **do**
  $dominated \leftarrow false$
  for each pattern $Y \in Processed$ **do**
    if $X \subset Y$ and $\text{aocc}(Y) \geq \text{aocc}(X)$ and $u(Y) \geq u(X)$ **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 $Output$

Algorithm 4 FILTER-WEAK

Require: HOUI list $\mathcal{C}_H$
Ensure: Weak MHOUI set

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

Remark 1. *The $Processed$ list must contain all previously processed HOUIs, including those that are themselves dominated. A dominated HOUI may still dominate a smaller HOUI.*

8 Correctness Analysis

Theorem 11 (Soundness of HOUI Enumeration)**.** *Every itemset inserted into $\mathcal{C}_H$ by Algorithm 2 is a HOUI.*

Proof. An itemset Y is inserted into $\mathcal{C}_H$ only if:

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

$$u(Y) \geq \mu,$$

and

$$\text{aocc}(Y) \geq \beta.$$

These are exactly the three conditions of a HOUI. □

Theorem 12 (Completeness of HOUI Enumeration)**.** *Algorithm 2 does not prune any itemset that can be a HOUI.*

Proof. The DFS procedure enumerates the itemset lattice according to a fixed item order. Each itemset has exactly one ordered construction path. Support pruning is safe by support anti-monotonicity. TWU pruning is safe by TWU anti-monotonicity. UUB, OUB_1, and OUB_2 are proven upper bounds for utility and occupancy of descendants. Therefore, no HOUI is incorrectly pruned. □

Theorem 13 (Correctness of Strong Filtering)**.** *Algorithm 3 returns exactly $\mathcal{MHOUI}^s$.*

Proof. The input list contains all HOUIs. Sorting by decreasing length ensures that any strict superset of an itemset X is processed before X. For each X, the algorithm checks whether there exists a previously processed HOUI Y such that:

$$X \subset Y,$$

$$\text{aocc}(Y) \geq \text{aocc}(X),$$

and

$$u(Y) \geq u(X).$$

This is exactly the strong dominance condition. If such Y exists, X is not a strong MHOUI. Otherwise, X is retained. Hence the output is exactly $\mathcal{MHOUI}^s$. □

Theorem 14 (Correctness of Weak Filtering)**.** *Algorithm 4 returns exactly* $\mathcal{MHOUI}^w$.

Proof. The proof is analogous to the strong case, except that domination requires at least one strict improvement:

$$\text{aocc}(Y) > \text{aocc}(X) \quad \text{or} \quad u(Y) > u(X).$$

This matches the weak MHOUI definition. □

9 Complexity Analysis

Let $n = |\mathcal{D}|$ be the number of transactions, $m = |\mathcal{I}|$ be the number of items, and m' be the number of items remaining after singleton support and TWU pruning. Let

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

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

In the worst case, exact itemset mining remains exponential because the number of possible itemsets is

$$2^{m'} - 1.$$

For each extension, bitset intersection costs

$$O(W),$$

and support counting costs

$$O(W)$$

using population count instructions.

If L_Y is built by scanning the set bits of B_Y, BOU-list construction costs approximately

$$O(\text{sup}(Y)).$$

Thus, the practical cost per node is approximately

$$O(W + \text{sup}(Y)).$$

Let $q = |\mathcal{HOUI}|$ be the number of HOUIs. The naive dominance filtering phase compares each HOUI with previously processed HOUIs. If itemsets are represented as item-bitsets, subset checking costs

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

Thus, the worst-case filtering cost is

$$O\left(q^2 \cdot \left\lceil \frac{m'}{64} \right\rceil\right).$$

The memory usage is dominated by vertical bitsets, BOU-lists, and the HOUI candidate list. The vertical bitsets require

$$O(m'W)$$

machine words. The HOUI candidate list requires

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

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

10 Experimental Evaluation

Since MHOUI is a newly defined pattern type, no prior baseline mines exactly the same output. The experimental goal is therefore twofold: (i) compare the proposed MHOUI family with algorithms from neighboring pattern-mining families, and (ii) visualize how the proposed weak, strong, and direct-strong variants behave under support, occupancy, and utility threshold changes.

All experiments were executed on the real datasets available in the project repository and summarized in `results/mhoui_compare_full/run_summary.csv`. The visualization pack was generated from the same summary file using `scripts/plot_mhoui_results.py`. In total, the benchmark contains 473 threshold configurations. Among them, 442 completed successfully and 31 reached the configured time limit. All HOUI, weak MHOUI, strong MHOUI, and direct strong MHOUI runs completed; timeouts occurred only for baseline algorithms.

10.1 Baseline Algorithms

Table 2 summarizes the algorithms used in the comparison. The baselines were selected from the implemented algorithms whose input assumptions match the available utility datasets.

Table 2: Algorithms used for evaluating MHOUI-Miner.

Family	Algorithms	Purpose
High-utility itemset mining	Two-Phase, HUI-Miner, FHM, EFIM, UP-Growth	Utility-oriented baselines
Concise utility mining	Closed-FHUIM, CHUI-Miner	Concise high-utility baselines
Ablation	HOUI, Weak MHOUI, Strong MHOUI	Effect of MHOUI dominance filtering
Proposed	Direct Strong MHOUI	Final exact direct strong variant

10.2 Datasets

The experiments use four real utility datasets: Foodmart, Liquor, Fruithut Utility, and Chainstore. Foodmart, Liquor, and Fruithut Utility were evaluated with 141 runs each. Chainstore was evaluated with 50 runs because the larger transaction volume makes the strongest baseline settings more expensive. The datasets cover small, medium, and large utility-mining workloads and therefore show both low-latency and high-memory regimes.

10.3 Threshold Settings

The minimum utility threshold is defined as:

$$\mu = \left\lceil minutil\_ratio \cdot \sum_{T \in \mathcal{D}} TU(T) \right\rceil,$$

where $minutil\_ratio$ is dataset-specific. Foodmart uses support ratios $\{0.005, 0.01, 0.02\}$, occupancy thresholds $\{0.10, 0.20, 0.40\}$, and utility ratios $\{0.0001, 0.0005, 0.001\}$. Liquor and Fruithut Utility use support ratios $\{0.01, 0.02, 0.05\}$, occupancy thresholds $\{0.20, 0.40, 0.60\}$, and utility ratios

$\{0.0005, 0.001, 0.005\}$. Chainstore uses support ratios $\{0.02, 0.05\}$, occupancy thresholds $\{0.20, 0.40\}$, and utility ratios $\{0.001, 0.005\}$.

10.4 Evaluation Metrics

The benchmark records the following statistics:

- Runtime in seconds.
- Peak RAM usage in MB.
- Output footprint in bytes.
- Number of visited nodes.
- Number of generated candidates.
- Number pruned by support.
- Number pruned by TWU.
- Number pruned by UUB.
- Number pruned by OUB_1 and OUB_2.
- Number pruned by dominance.
- Number of HOUIs.
- Number of weak MHOUIs.
- Number of strong MHOUIs.
- Average support.
- Average utility.
- Average occupancy.
- Average itemset length.

The main compression ratio is:

$$CR_{HOUI} = \frac{|\mathcal{HOUI}|}{|\mathcal{MHOUI}^s|}.$$

The reduction percentage is:

$$Reduction_{HOUI} = \left(1 - \frac{|\mathcal{MHOUI}^s|}{|\mathcal{HOUI}|}\right) \times 100\%.$$

10.5 Ablation Studies

The central ablation verifies the inclusion chain:

$$|\mathcal{HOUI}| \geq |\mathcal{MHOUI}^w| \geq |\mathcal{MHOUI}^s|.$$

This verifies that weak and strong dominance filtering never expands the output. The direct strong variant is evaluated separately to confirm that it preserves the strong MHOUI result while avoiding unnecessary materialization.

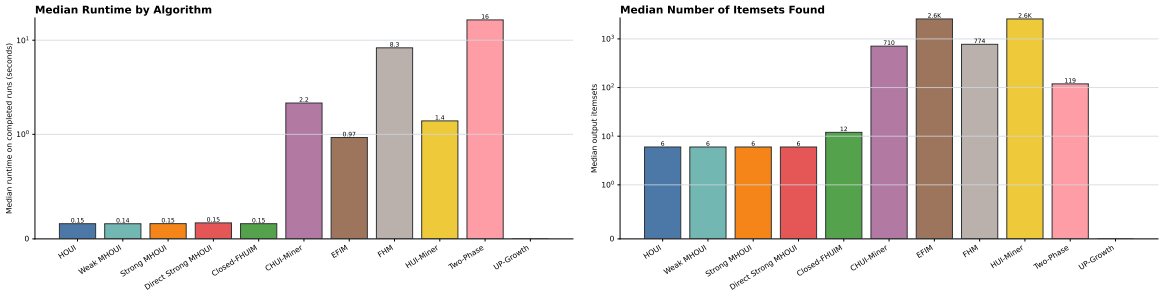
10.6 Benchmark Overview

Figure 1 summarizes the median behavior across completed runs. The MHOUI family is consistently positioned in the low-runtime region while producing compact outputs. Utility-only baselines can generate substantially larger itemset collections, which is expected because they do not enforce classical occupancy and MHOUI dominance.

Figure 2 separates the comparison by dataset. This view is useful because the datasets have different transaction counts, item distributions, and utility scales. Chainstore in particular stresses memory usage, while Foodmart exposes many zero-output threshold combinations.

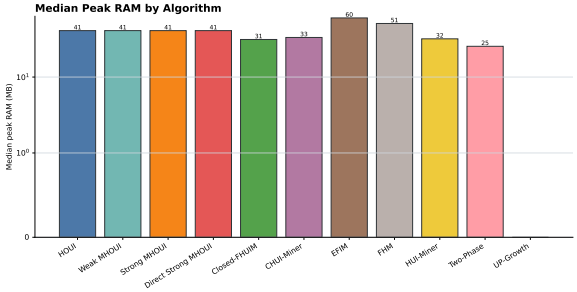
Figure 3 reports run completion behavior. The status chart shows that the MHOUI family completed all tested configurations, while timeouts are concentrated in the baseline algorithms.

Figure 4 shows the 90th-percentile behavior. These plots are important because median runtime can hide difficult threshold combinations. The MHOUI family remains stable in the high-percentile view.

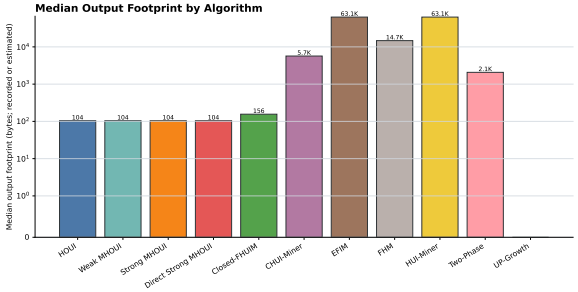


(a) Median runtime.

(b) Median output size.

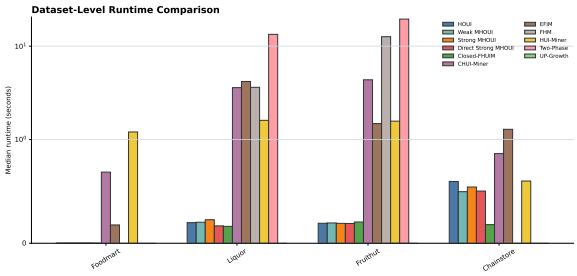


(c) Median peak RAM.

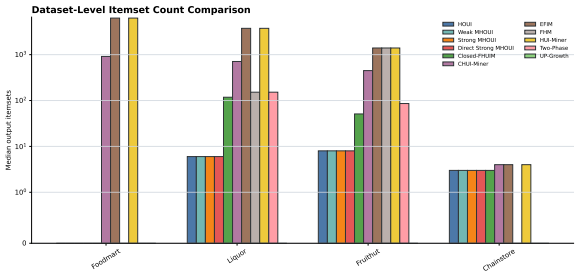


(d) Median output footprint.

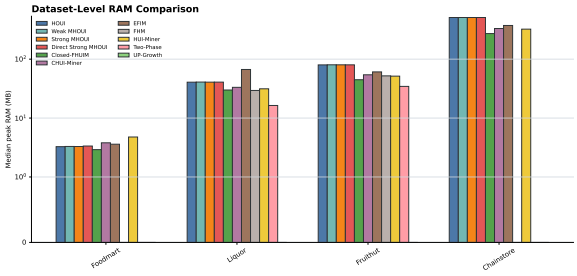
Figure 1: Global median comparison across completed benchmark runs.



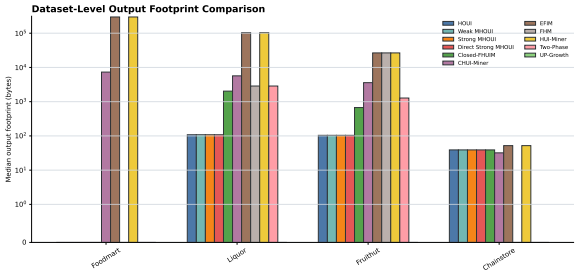
(a) Runtime by dataset.



(b) Output size by dataset.

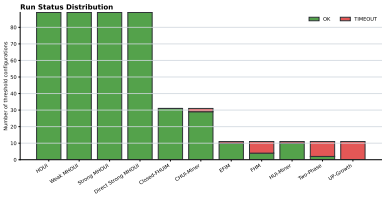


(c) Peak RAM by dataset.

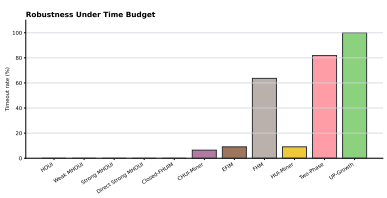


(d) Output footprint by dataset.

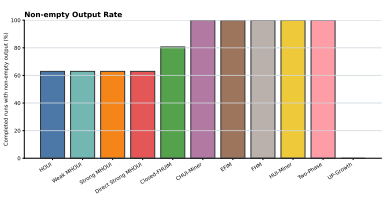
Figure 2: Dataset-level comparison of all algorithms.



(a) Run status.

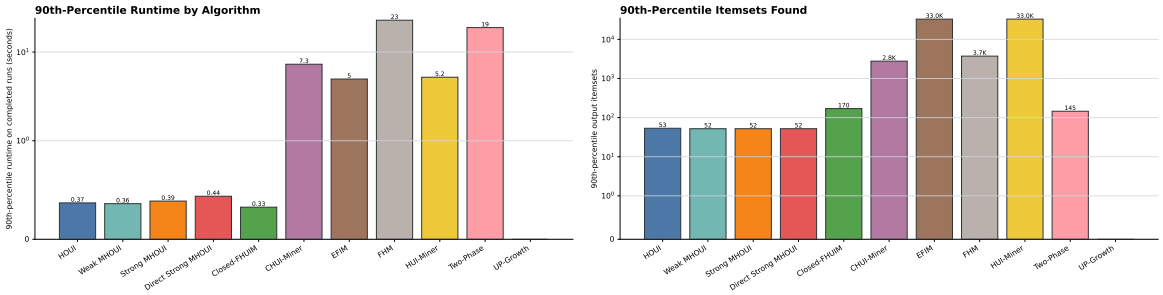


(b) Timeout rate.



(c) Non-empty output rate.

Figure 3: Completion and output availability across threshold configurations.



(a) 90th-percentile runtime.

(b) 90th-percentile output size.

Figure 4: Tail behavior across completed runs.

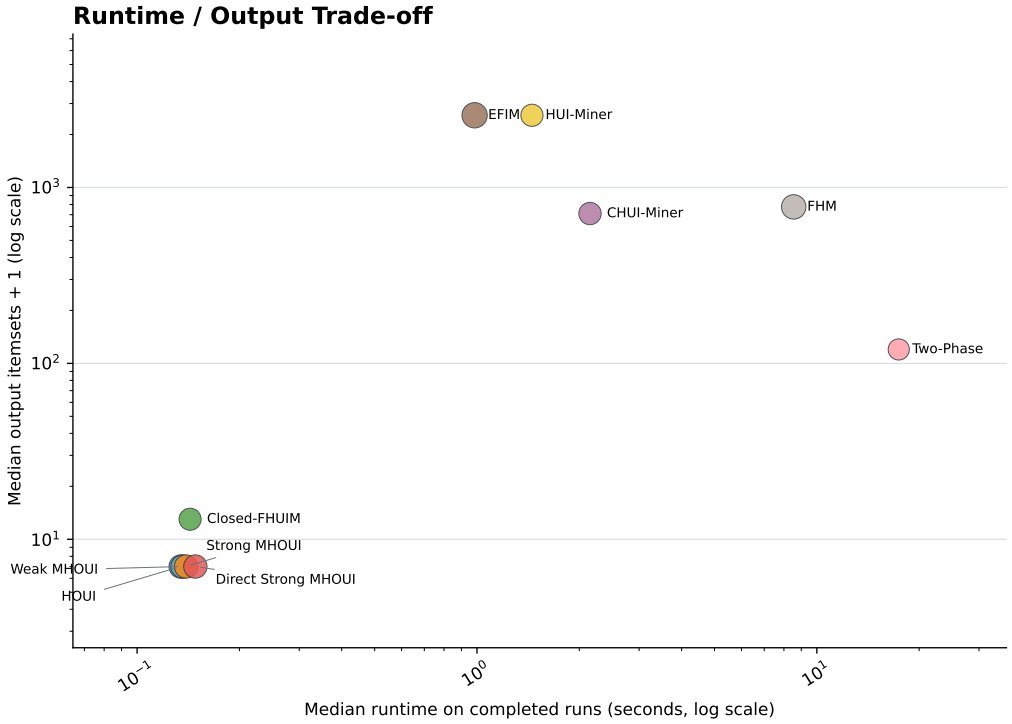


Figure 5: Runtime-output trade-off. Marker size reflects median peak RAM. This figure highlights that MHOUI variants occupy the compact, low-runtime region of the comparison.

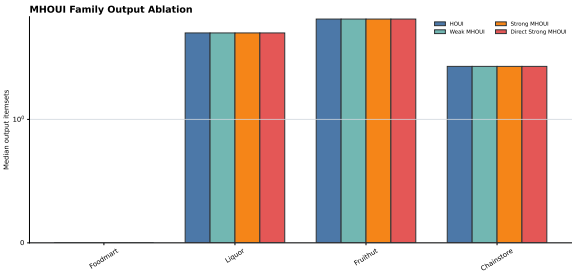
10.7 MHOUI-Specific Analysis

Figure 6 focuses on the MHOUI family. The output ablation directly visualizes the transition from HOUI to weak MHOUI to strong MHOUI. The runtime ablation compares the cost of the corresponding variants. The compression and pruning plots summarize the concise-representation effect and the contribution of safe pruning bounds.

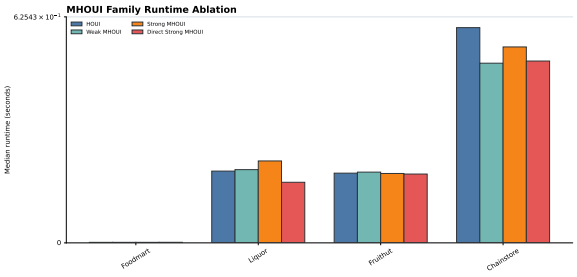
10.8 Threshold Sensitivity

Figures 7 and 8 show strong MHOUI sensitivity to support, occupancy, and utility thresholds. The output heatmaps emphasize how stricter occupancy and utility thresholds compact the result set, while the runtime heatmaps show the computational response to the same settings.

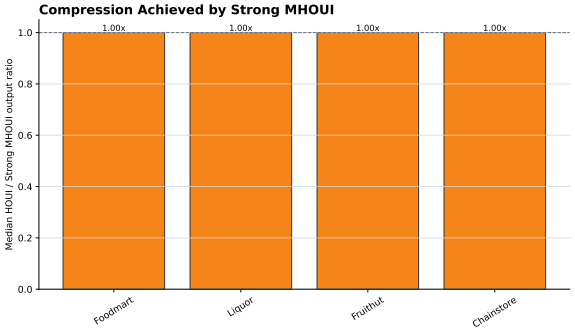
Figure 9 isolates the occupancy dimension. These profiles make the compactness effect easier to inspect because each curve tracks output count as the minimum occupancy threshold changes.



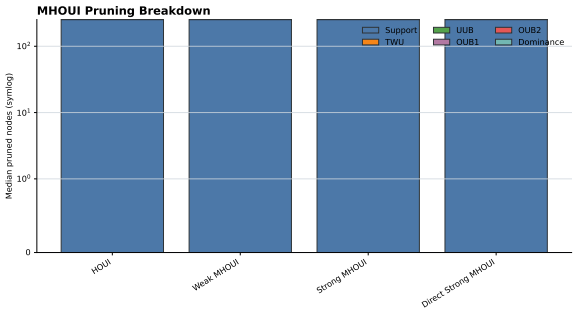
(a) Output ablation.



(b) Runtime ablation.

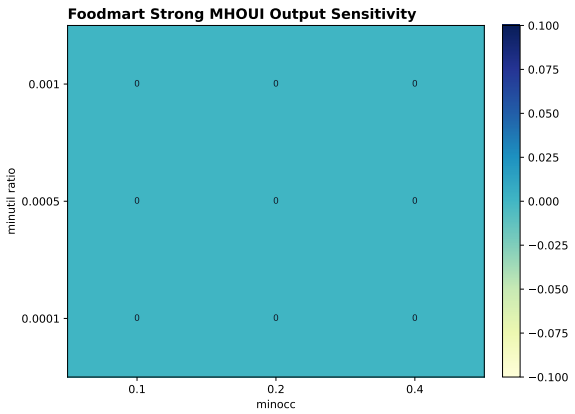


(c) HOUI-to-strong compression.

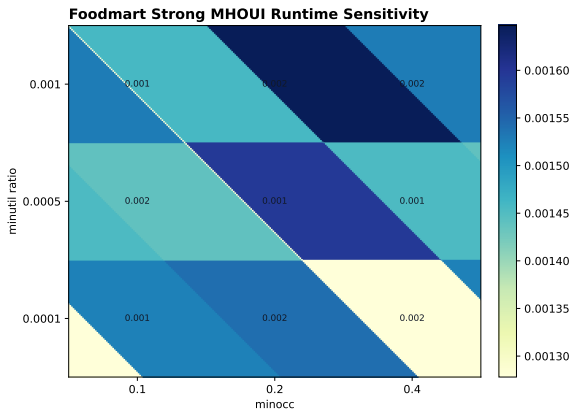


(d) Pruning breakdown.

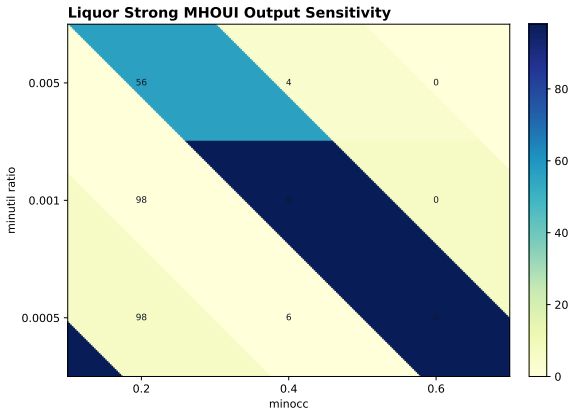
Figure 6: MHOUI-focused ablation, compression, and pruning analysis.



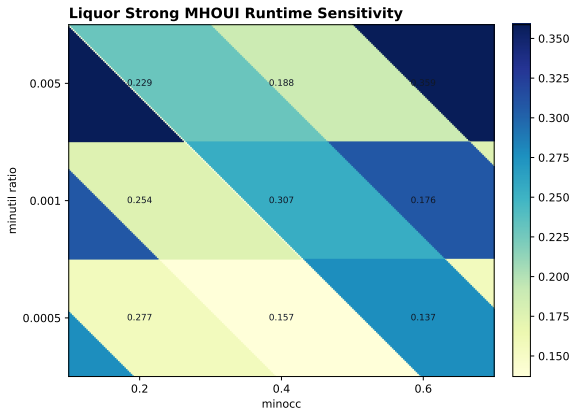
(a) Foodmart output.



(b) Foodmart runtime.

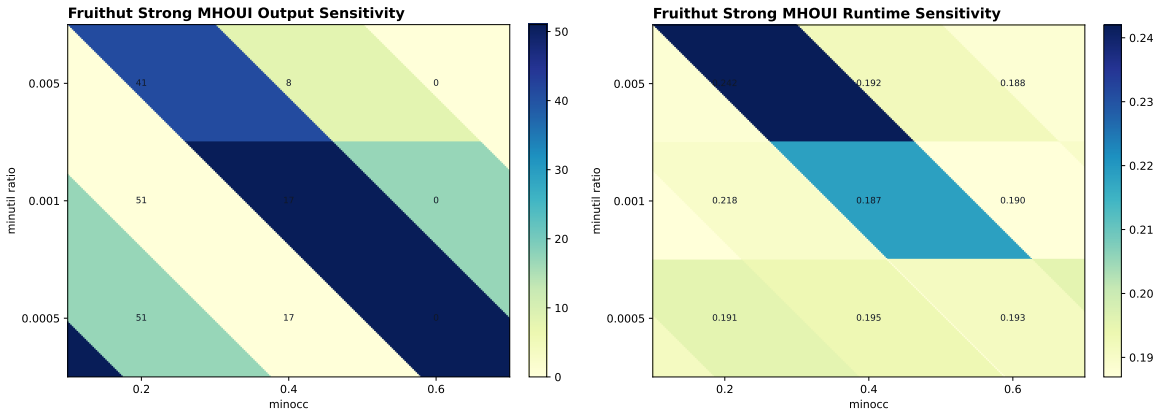


(c) Liquor output.



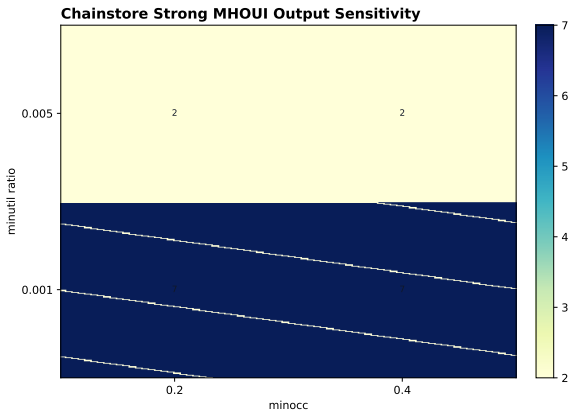
(d) Liquor runtime.

Figure 7: Strong MHOUI threshold heatmaps for Foodmart and Liquor.

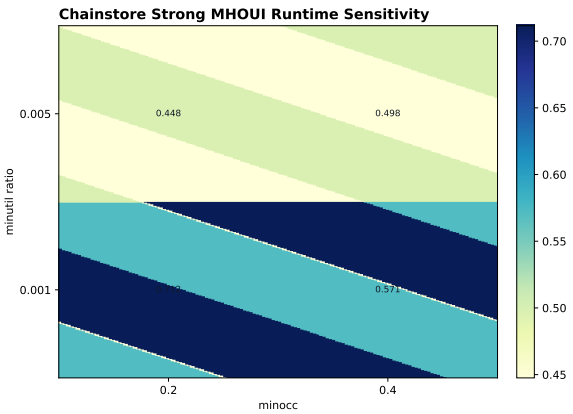


(a) Fruithut output.

(b) Fruithut runtime.

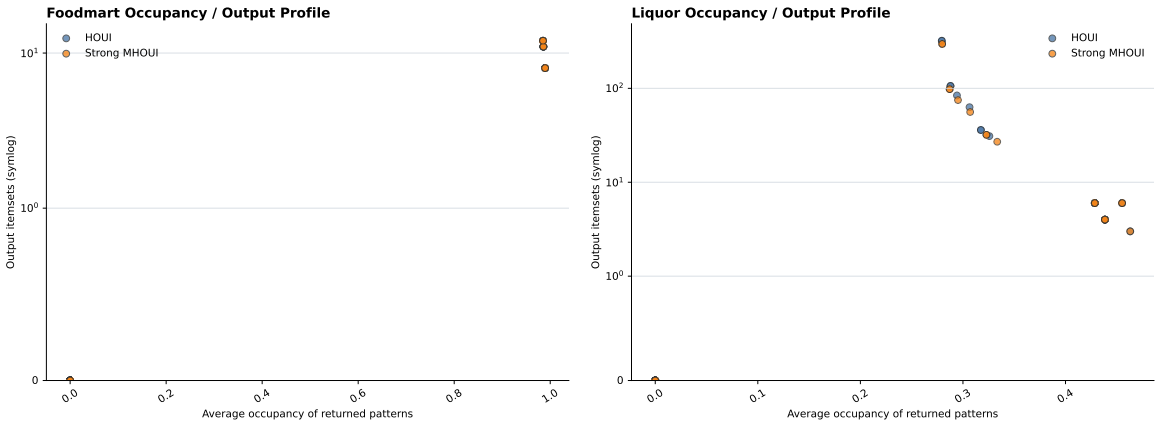


(c) Chainstore output.



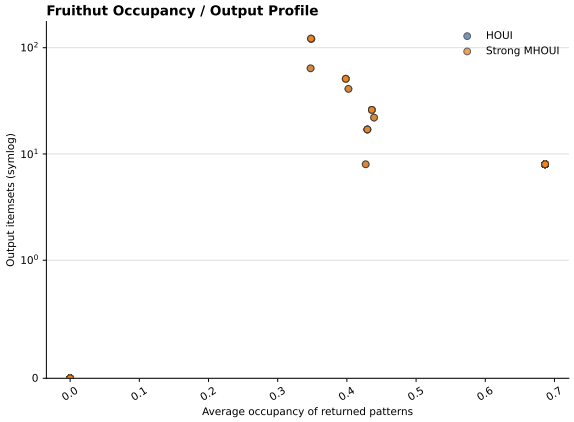
(d) Chainstore runtime.

Figure 8: Strong MHOUI threshold heatmaps for Fruithut and Chainstore.

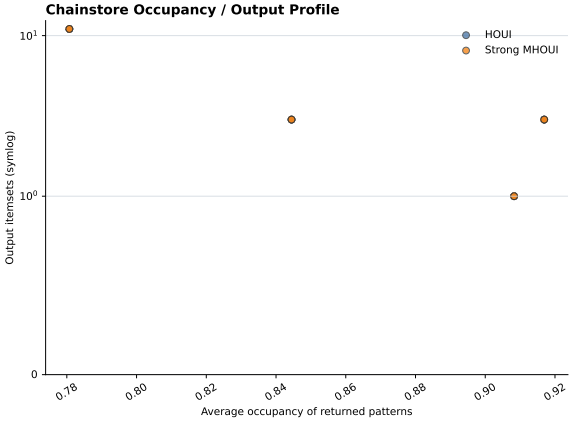


(a) Foodmart.

(b) Liquor.



(c) Fruithut.

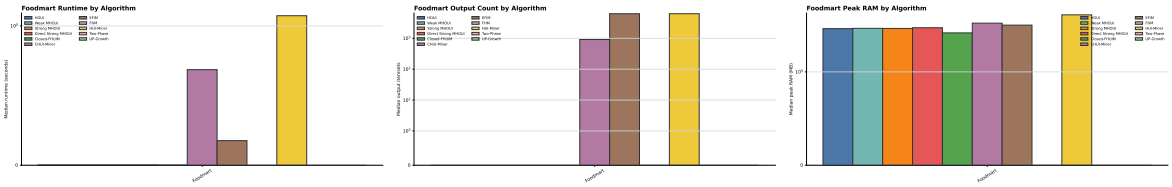


(d) Chainstore.

Figure 9: Occupancy-output sensitivity profiles for strong MHOUI.

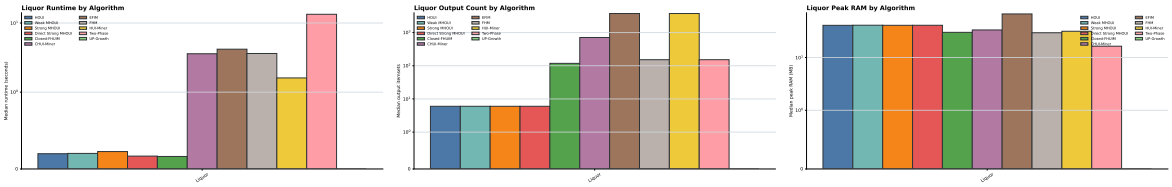
10.9 Per-Dataset Views

Figures 10, 11, 12, and 13 provide the full per-dataset comparison. Each dataset has a runtime chart, an output-size chart, and a RAM chart so that the reader can inspect algorithm behavior without averaging away dataset-specific effects.



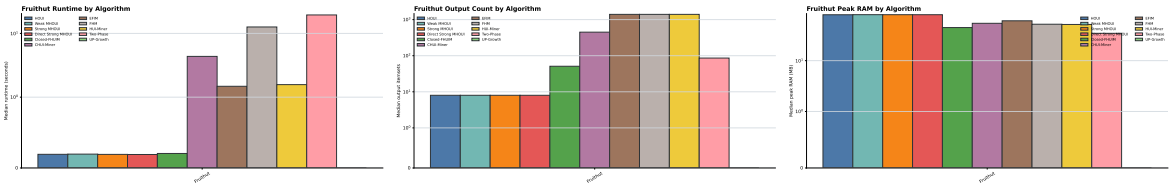
(a) Runtime. (b) Output size. (c) Peak RAM.

Figure 10: Foodmart per-algorithm comparison.



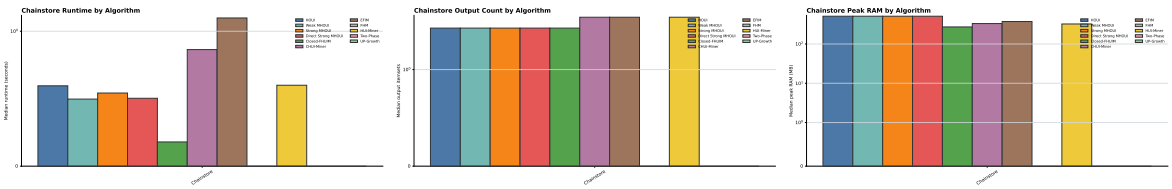
(a) Runtime. (b) Output size. (c) Peak RAM.

Figure 11: Liquor per-algorithm comparison.



(a) Runtime. (b) Output size. (c) Peak RAM.

Figure 12: Fruithut Utility per-algorithm comparison.



(a) Runtime. (b) Output size. (c) Peak RAM.

Figure 13: Chainstore per-algorithm comparison.

11 Discussion

MHOUI-Miner is designed as a concise representation method for utility-aware and occupancy-aware transaction patterns. Unlike frequent itemset mining, it does not return every frequent combination. Unlike maximal frequent itemset mining, its maximality is not based only on support. Unlike high-occupancy itemset mining, it requires high total utility. Unlike high-utility itemset mining, it also requires high classical transaction occupancy. Unlike HUOPM-style mining, it separates classical occupancy from total utility rather than combining them into a utility-occupancy ratio.

The proposed weak and strong MHOUI variants provide two levels of compactness. Weak MHOUI preserves itemsets when no strict superset improves at least one objective without worsening the other. Strong MHOUI further removes itemsets when a strict superset has equal or better occupancy and equal or better utility. Therefore, strong MHOUI is more compact.

The main algorithmic difficulty is that neither utility nor average classical occupancy is anti-monotone. Therefore, MHOUI-Miner relies on safe upper bounds rather than naive pruning. The hybrid bitset and BOU-list representation is designed to combine fast support computation with utility and occupancy bound evaluation.

12 Threats to Validity

The proposed MHOUI definition assumes positive utilities. If negative utilities are allowed, TWU and remaining-utility pruning must be modified. The base formulation also assumes static external utilities and deterministic transaction data. Dynamic profits, uncertain transactions, and streaming databases require additional theory.

Another threat concerns utility generation for binary datasets. If quantities and external utilities are synthetically generated, the generator, random seed, utility range, and distribution must be reported for reproducibility.

Finally, runtime comparisons must be interpreted carefully because MHOUI-Miner solves a different output task from FI, MFI, HOI, HUI, and HUOPM algorithms. Therefore, the primary evidence should focus on output compactness, pattern quality, and ablation correctness, not only runtime.

The output-footprint figures should also be interpreted as a storage proxy. The benchmark was configured to keep statistics rather than detailed pattern files, so footprint values are taken from recorded reports when available and estimated from output counts and average lengths otherwise.

13 Conclusion

This paper introduced Maximal High-Occupancy Utility Itemset mining and proposed MHOUI-Miner, an exact method for discovering weak and strong MHOUIs. The proposed definition combines support, classical occupancy, total utility, and strict-superset Pareto maximality. We formalized HOUI, weak MHOUI, and strong MHOUI, proved their theoretical properties, and derived safe support, utility, and occupancy pruning bounds.

MHOUI-Miner uses a hybrid vertical representation combining transaction-id bitsets with Bitset-Occupancy-Utility lists. The algorithm first mines all HOUIs exactly and then applies weak or strong dominance filtering. A direct optimized version can further use branch dominance pruning.

The proposed framework provides a compact, interpretable, and utility-aware representation of dense transaction patterns. Future work will focus on tighter upper bounds, faster dominance filtering, top-k MHOUI mining, incremental MHOUI mining, and extensions to negative utilities and uncertain transaction databases.

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] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] Ying Liu, Wei-keng Liao, and Alok Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 689–695, 2005.

[13] Mengchi Liu and Junfeng Qu. Mining high utility itemsets without candidate generation. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 55–64, 2012.

[14] Philippe Fournier-Viger, Cheng-Wei Wu, Souleymane Zida, and Vincent S. Tseng. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, pages 83–92, 2014.

[15] Souleymane Zida, Philippe Fournier-Viger, Jerry Chun-Wei Lin, Cheng-Wei Wu, and Vincent S. Tseng. EFIM: A fast and memory efficient algorithm for high-utility itemset mining. *Knowledge and Information Systems*, 51:595–625, 2017.

[16] 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.

[17] Chien-Ming Chen, Lili Chen, and Wensheng Gan. Flexible pattern discovery and analysis. *Information Sciences*, 2021.

[18] Hai Duong, Tin Truong, Bay Vo, and Philippe Fournier-Viger. Efficient algorithms for mining maximal high utility itemsets. *Knowledge-Based Systems*, 2022.

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

[20] Wensheng Gan, Jerry Chun-Wei Lin, Han-Chieh Chao, and Philip S. Yu. A survey of utility-oriented pattern mining. *IEEE Transactions on Knowledge and Data Engineering*, 2021.