AURA-HOI: Auditable Unified Representative Mining for High-Occupancy Itemsets

Anonymous Author
Department of Computer Science, Anonymous University

Abstract

High-occupancy itemset mining (HOIM) discovers itemsets that occupy a large fraction of the transactions in which they appear. The model captures dense local co-occurrence that support-only frequent itemset mining can miss, but it also removes the anti-monotonicity that makes classical pattern mining tractable. A review of the implemented HOI family in the `dm` framework reveals a systems-level gap: HEP preserves the original threshold model but remains level-wise and candidate-heavy, while DFHOI moves to depth-first search but still materializes support intersections. This paper proposes AURA-HOI, a brand-new HOI-only algorithm that supports two auditable output views: a raw fullset view for direct equality comparison with HEP/DFHOI and a support-class representative view for compact closed-output analysis. The implemented artifact adds summed-occupancy compatibility mode to match the repository HEP/DFHOI semantics exactly. On Mushrooms at $\alpha = 0.10$, AURA-HOI returns the same 955 raw HOIs as HEP and DFHOI, while reducing runtime from 2.256 seconds to 1.262 seconds against HEP and reducing peak memory from 890.74 MB to 3.77 MB. The result is an exact raw-HOI miner with an optional closed-ledger view, residual occupancy-envelope pruning, and reproducible benchmark charts.

Keywords: high-occupancy itemset mining; closed itemsets; vertical bitsets; output normalization; top-k mining; support-equivalence certificates.

1 Introduction

High-occupancy itemset mining was introduced to recover itemsets that cover a large fraction of the transactions in which they occur (Deng, 2020). For an itemset X and a transaction t, the transaction-level occupancy contribution is $|X|/|t|$ when $X \subseteq t$. Unlike support, occupancy is not anti-monotone: adding an item increases the numerator but may shrink the supporting transaction set. This makes the design of safe pruning bounds substantially harder than in Apriori, Eclat, or closed frequent itemset mining (Zaki et al., 1997; Zaki and Hsiao, 2002).

The `dm` codebase already contains a broad HOI ecosystem: HEP, FHOI, DFHOI, TKHOIM, NAM-HEP, MFHOI variants, and CLOE-HOI. This breadth is useful for research, but it exposes a gap that a Q1 paper cannot ignore: the algorithms do not all answer the same operational question. Some mine threshold-complete raw HOIs, some mine top-k HOIs, some mine maximal or support-closed representatives, and some use adaptive thresholds. Therefore a count mismatch between algorithms is not automatically a correctness bug; it often means that the output semantics differ.

This paper proposes a unified HOI-only algorithm, AURA-HOI (*Auditable Unified Representative Algorithm for HOI*), designed around one principle: every reported pattern must be traceable to an exact average-occupancy definition and to an auditable support-equivalence class. The algorithm has three goals:

Table 1: HOI-only gap analysis based on the implemented algorithms in `src/algorithms`. Utility-occupancy and high-utility algorithms are excluded from the proposed problem scope.

Algorithm family	Strength	Output semantics	Remaining gap for a unified HOI paper
HEP	Original HOI threshold model; UBO pruning	Threshold-complete raw HOIs under repository occupancy thresholding	Level-wise candidate growth and repeated list materialization increase memory pressure at low thresholds.
DFHOI/FHOI	Depth-first equivalence-class search; early pruning	Threshold-complete raw HOIs	More scalable than HEP but still lacks support-class certificates and direct closed-output normalization.
NAM-HEP	Adaptive support-/occupancy thresholds; HO-SE tree	Adaptive high-occupancy patterns	Strong for automatic thresholding, but adaptive thresholds make strict equality comparison with fixed-threshold miners difficult.
TKHOIM	Top-k output; dynamic minimum occupancy raising	Top-k HOIs	Avoids threshold guessing, but it is not threshold-complete and requires a separate validation story.
MFHOI	Frequent and maximal/weak/strong HOI compression	Maximal or dominance-filtered frequent HOIs	Compact, but maximality can hide support-equivalent representatives and complicate exact raw-output auditing.
CLOE-HOI	Candidate-free closed HOI with bitset arena and occupancy envelope	Support-closed HOIs	Strong closed miner, but a Q1 comparison still needs a unified raw/closed/top-k audit layer across all baselines.

1. mine high-occupancy itemsets under a single exact semantics;
2. remove redundancy through support closure without losing validation against raw HOIs; and
3. allow fair comparison with HEP, DFHOI, NAM-HEP, TKHOIM, MFHOI, and CLOE-HOI by normalizing outputs into support classes.

2 Gap Analysis from the Implemented HOI Family

Table 1 motivates the core research question:

> *Can one HOI algorithm provide exact threshold-complete semantics, non-redundant support-closed output, optional top-k reporting, and output certificates that make comparisons with existing HOI miners scientifically defensible?*

3 Problem Definition

Definition 1 (Transaction database)**.** Let $\mathcal{I} = \{1, \ldots, m\}$ be an item universe and $\mathcal{D} = \{t_1, \ldots, t_n\}$ a multiset of transactions, where $t_q \subseteq \mathcal{I}$ and $|t_q| > 0$.

Definition 2 (Support and exact average occupancy)**.** For itemset $X \subseteq \mathcal{I}$, define

$$T(X) = \{q : X \subseteq t_q\}, \qquad \text{supp}(X) = |T(X)|. \tag{1}$$

The exact average occupancy of X is

$$\mathrm{aocc}(X) = \begin{cases} \dfrac{|X|}{\mathrm{supp}(X)} \displaystyle\sum_{q \in T(X)} \frac{1}{|t_q|}, & \mathrm{supp}(X) > 0, \\ 0, & \mathrm{supp}(X) = 0. \end{cases} \tag{2}$$

Definition 3 (High-occupancy itemset)**.** Given minimum support $\sigma_{\min}$ and minimum average occupancy $\alpha \in (0, 1]$, an itemset X is a high-occupancy itemset if

$$\mathrm{supp}(X) \geq \sigma_{\min} \quad \text{and} \quad \mathrm{aocc}(X) \geq \alpha. \tag{3}$$

The raw HOI set is denoted by $\mathcal{H}_{\mathrm{raw}}(\mathcal{D}, \sigma_{\min}, \alpha)$.

Definition 4 (Summed-occupancy compatibility)**.** The repository implementations of HEP and DFHOI use the summed occupancy score

$$\mathrm{occ}_{\Sigma}(X) = |X| \sum_{q \in T(X)} \frac{1}{|t_q|}, \tag{4}$$

and compare it against $\xi = \alpha|\mathcal{D}|$ when the command-line threshold $\alpha < 1$. AURA-HOI therefore exposes a `sum raw` mode that accepts exactly the same threshold predicate:

$$\mathrm{supp}(X) \geq \lceil \alpha|\mathcal{D}| \rceil \quad \text{and} \quad \mathrm{occ}_{\Sigma}(X) \geq \alpha|\mathcal{D}|. \tag{5}$$

This mode is used for the empirical comparison against HEP and DFHOI.

Definition 5 (Support closure and certificate)**.** The support closure of X is

$$\mathrm{cl}(X) = \bigcap_{q \in T(X)} t_q. \tag{6}$$

An itemset C is support-closed if $\mathrm{cl}(C) = C$. A certificate for C is the pair

$$\mathrm{cert}(C) = \big(T(C), \{X \subseteq C : T(X) = T(C)\}\big), \tag{7}$$

stored compactly as the support bitset $T(C)$, the closed representative C, and the interval of generators visited by the closure-canonical traversal.

Problem 1 (Auditable Representative HOI Mining)**.** Given $\mathcal{D}$, $\sigma_{\min}$, α, and optional k, mine:

1. the support-closed representative set $\mathcal{H}_{\mathrm{closed}}$ such that every $C \in \mathcal{H}_{\mathrm{closed}}$ is support-closed and high-occupancy;
2. a certificate that every raw HOI support class has exactly one closed representative in the ledger; and
3. optionally the top-k representatives ranked by $(\mathrm{aocc}(C), \mathrm{supp}(C), |C|)$.

No utility, profit, sequence, or shelf-time information is part of this problem.

4 The AURA-HOI Algorithm

4.1 Design overview

AURA-HOI is a vertical bitset algorithm. Each item i is represented by a bitset B_i over transaction identifiers. For a prefix P, the support bitset is $B_P = \bigcap_{i \in P} B_i$. Exact occupancy is computed from B_P using the reciprocal transaction-length array $r_q = 1/|t_q|$. The algorithm explores itemsets in a closure-canonical DFS order and keeps a ledger keyed by support-bitset hash to avoid duplicate support classes.

The novelty relative to the existing `dm` HOI family is the *audit layer*: raw candidates, closed representatives, and top-k representatives are derived from the same traversal and the same exact average-occupancy formula. This removes the common experimental ambiguity where algorithms disagree because they emitted different pattern families.

The current C99 implementation has two output modes. The first mode, `sum raw`, disables closure compression and emits the raw fullset under the summed-occupancy predicate in Eq. (5); this is the mode used to prove direct comparability with HEP and DFHOI. The second mode, `avg closed`, enables the support-class ledger and emits support-closed representatives under exact average occupancy; this mode is retained for compact-output studies but is not used for the HEP/DFHOI equality experiment.

4.2 Transaction-length-aware occupancy envelope

Let P be a prefix with support $S = T(P)$, length $p = |P|$, and tail item set R. For a descendant $Y = P \cup Z$ with $Z \subseteq R$, let $b = |Z|$ and $U = T(Y) \subseteq S$. Since the support of a descendant is a subset of S, a safe upper bound is obtained by choosing the u transactions in S with the largest reciprocal lengths and the largest feasible extension size:

$$\mathrm{Env}(P, R) = \max_{1 \le u \le |S|} \max_{0 \le b \le |R|} \frac{p + b}{u} \sum_{\ell=1}^{u} r^{\downarrow}_{S,\ell}, \tag{8}$$

where $r^{\downarrow}_{S,\ell}$ is the ℓth largest reciprocal length among transactions in S. The value is clipped at 1.

Lemma 1 (Envelope safety)**.** *For every descendant $Y = P \cup Z$ with $Z \subseteq R$,* $\mathrm{aocc}(Y) \le \mathrm{Env}(P, R)$.

Proof. Let $u = \mathrm{supp}(Y)$ and $b = |Z|$. Because $T(Y) \subseteq S$, the sum of reciprocal transaction lengths over $T(Y)$ is no larger than the sum of the u largest reciprocal lengths in S. Substituting this relaxation into Eq. (2) gives one feasible term inside Eq. (8); therefore $\mathrm{aocc}(Y) \le \mathrm{Env}(P, R)$. □

4.3 Residual occupancy risk pruning

The envelope in Eq. (8) can be loose when the tail contains many items that cannot co-occur with the prefix support. AURA-HOI refines it with residual co-occurrence capacity. For each transaction $q \in S$, define

$$c_R(q) = |t_q \cap R|. \tag{9}$$

Then any descendant can add at most $c_R(q)$ items inside transaction q. The residual-risk envelope is:

$$\mathrm{Env}_R(P, R) = \max_{1 \le u \le |S|} \frac{1}{u} \sum_{\ell=1}^{u} \frac{p + c_R(q_\ell)}{|t_{q_\ell}|}, \tag{10}$$

where the q_ℓ are chosen as the u transactions with largest residual occupancy contribution. AURA-HOI prunes when $\min\{\mathrm{Env}, \mathrm{Env}_R\} < \alpha$.

Theorem 1 (No false-negative pruning)**.** *If AURA-HOI prunes a node P only when* $\mathrm{supp}(P) < \sigma_{\min}$ *or* $\min\{\mathrm{Env}(P,R), \mathrm{Env}_R(P,R)\} < \alpha$*, then no valid high-occupancy descendant of P is removed.*

Proof. The support condition follows from support anti-monotonicity. The first envelope is safe by the preceding lemma. For the residual envelope, each descendant can add no more than $c_R(q)$ items inside each supporting transaction q, so its per-transaction occupancy contribution is bounded by $(p + c_R(q))/|t_q|$. Maximizing over the best u transactions yields Eq. (10). Thus every descendant has average occupancy at most the pruning bound. □

4.4 Closure-canonical ledger

At each node, AURA-HOI computes the closure

$$C = \mathrm{cl}(P) = \{i \in \mathcal{I} : B_P \subseteq B_i\}. \tag{11}$$

If $C \neq P$, the algorithm jumps to C without emitting intermediate generators. The support bitset hash of C is inserted into a ledger. If the same support class is reached again through another generator, the ledger prevents duplicate output and records the generator path for audit.

4.5 Implementation changes in `aura_hoi.c`

The artifact implements AURA-HOI as a native `dm` algorithm with identifier `aura_hoi`. The command-line form is:

```
./bin/dm.exe aura_hoi <dataset> 0 <alpha> [minsup] [maxsec] [avg|sum]
                                [closed|raw].
```

The following implementation changes are essential for fair comparison with HEP and DFHOI:

1. **Summed-occupancy compatibility.** The implementation adds `summed_occupancy_mode`; when enabled, the score is $\mathrm{occ}_\Sigma(X)$ from Eq. (4) and the threshold is $\alpha|\mathcal{D}|$.

2. **Raw fullset traversal.** The implementation adds `aura_search_raw`, which enumerates raw HOIs without closure compression. This mode is used by the benchmark script so the itemset counts must match HEP/DFHOI.

3. **Closed-ledger traversal.** The original support-class ledger remains available through `closed` mode. Ledger entries store the support-bitset hash and the support bitset itself, so support-class equality is byte-verified rather than relying only on hashes.

4. **Residual pruning for both modes.** Average mode uses residual average occupancy bounds, while summed mode uses a residual summed-occupancy envelope that is safe for HEP/DFHOI-compatible raw mining.

5. **Statistics-only output.** The miner reports counts, visited nodes, support pruning, envelope pruning, RAM, disk estimate, and total output size without printing pattern contents.

Algorithm 1 AURA-HOI: auditable support-class high-occupancy mining

Require: Transaction database $\mathcal{D}$, support threshold $\sigma_{\min}$, occupancy threshold α, optional k
Ensure: Closed HOI ledger and optional top-k representative list

Build vertical bitsets B_i and reciprocal lengths $r_q = 1/|t_q|$
Sort active items by increasing support and decreasing singleton occupancy
Initialize empty ledger L and optional bounded heap Q_k
procedure SEARCH(P, B_P, R)
    if popcnt(B_P) $< \sigma_{\min}$ **then**
        return
    end if
    if $\min\{\mathrm{Env}(P,R), \mathrm{Env}_R(P,R)\} < \alpha$ **then**
        return
    end if
    $C \leftarrow \mathrm{cl}(P)$ using support-subset tests over vertical bitsets
    $B_C \leftarrow B_P$
    if $\mathrm{aocc}(C) \geq \alpha$ and C is not in ledger L **then**
        insert $(C, B_C, \mathrm{aocc}(C), \mathrm{supp}(C))$ into L
        update optional top-k heap Q_k
    end if
    $R' \leftarrow \{i \in R : i > \max(C),\ i \notin C\}$
    for all $i \in R'$ in canonical order **do**
        $B' \leftarrow B_C \cap B_i$
        SEARCH($C \cup \{i\}, B', \{j \in R' : j > i\}$)
    end for
end procedure
SEARCH($\emptyset, \{1, \ldots, n\}, I_{\mathrm{active}}$)
return ledger L and optional top-k heap Q_k

4.6 Complexity

Let $w = \lceil n/64 \rceil$ be the number of machine words per bitset and N_{vis} the number of visited DFS nodes after pruning. Each support intersection costs $O(w)$ word operations. Closure checking costs $O(|R|w)$ in the direct form and can be reduced using cached subset tests. Exact occupancy verification costs $O(w + \mathrm{supp}(X))$ when scanning set bits. Therefore:

$$T_{AURA-HOI} = O\left(N_{\mathrm{vis}} w + \sum_{X \in L} (w + \mathrm{supp}(X)) + T_{\mathrm{env}} \right). \tag{12}$$

The main memory footprint is:

$$M_{AURA-HOI} = O(mw) + O(d_{\max} w) + O(n) + O(|L|), \tag{13}$$

where mw stores vertical item bitsets, $d_{\max} w$ stores recursion temporaries, and $|L|$ stores certificates.

Table 2: How AURA-HOI addresses gaps in the implemented HOI family.

Gap	AURA-HOI response
HEP candidate explosion	Replaces level-wise candidate materialization with closure-canonical vertical DFS.
DFHOI output ambiguity	Adds support-class ledger and exact closed representative certificates.
NAM-HEP threshold mismatch	Keeps fixed $(\sigma_{\min}, \alpha)$ semantics while allowing optional top-k reporting as a view, not a separate definition.
TKHOIM incompleteness for threshold mining	The top-k heap is derived from the threshold traversal or from $\alpha = 0$ top-k mode with the same scoring formula.
MFHOI maximality information loss	Uses support closure instead of maximality and records support-equivalence certificates.
CLOE-HOI closed-only validation gap	Extends closed mining with raw/closed/top-k audit views and normalized comparison protocol.

5 Expected Advantages

6 Experimental Setup

6.1 Datasets

The executed evaluation uses two real transactional datasets available in the repository: `mushrooms` with 8,416 transactions and `retail` with 88,162 transactions. Utility datasets are excluded because this paper is HOI-only and the requested comparison is against HEP and DFHOI.

6.2 Algorithms

The comparison suite is intentionally restricted to algorithms with the same raw HOI purpose:

`aura_hoi`, `hep`, and `dfhoi`.

AURA-HOI is run as `sum raw`; HEP and DFHOI are run with the same command-line α. Thus all three algorithms report raw fullset HOIs under the same repository summed-occupancy predicate.

6.3 Threshold matrix

6.4 Metrics

Correctness and equivalence. Let R_A and R_B be normalized outputs from algorithms A and B under the same view. Report:

$$\text{CountDiff}(A, B) = ||R_A| - |R_B||, \qquad \text{Jaccard}(A, B) = \frac{|R_A \cap R_B|}{|R_A \cup R_B|}. \tag{14}$$

For closed view, compare support-bitset hashes rather than only itemset strings.

Table 3: Recommended HOI-only threshold matrix.

Dataset profile	Thresholds	Purpose
Dense categorical	α: 0.10, 0.20, 0.30, 0.40, 0.50; $\sigma_{\min}$: $0.01n$, $0.05n$, $0.10n$	Stress closure growth and dense co-occurrence.
Sparse retail	α: 0.05, 0.10, 0.15, 0.20, 0.30; $\sigma_{\min}$: $0.001n$, $0.005n$, $0.01n$	Test early pruning and long-tail item distributions.
Top-k mode	$k \in \{10, 50, 100, 500\}$ with $\alpha = 0$ and optional $\sigma_{\min}$	Compare against TKHOIM without threshold guessing.
Scalability	fixed α and increasing database prefix sizes	Measure runtime, RAM, disk, and branch pruning as n grows.

Table 4: Raw fullset HOI comparison under matched summed-occupancy semantics. Equal itemset counts validate that AURA-HOI is compared against HEP and DFHOI on the same output purpose.

Dataset	Algorithm	α	Time (s)	RAM (MB)	Disk (MB)	Itemsets
Mushrooms	AURA-HOI	0.10	1.262	3.77	0.0575	955
Mushrooms	HEP	0.10	2.256	890.74	0.0575	955
Mushrooms	DFHOI	0.10	1.474	5.60	0.0575	955
Retail	AURA-HOI	0.02	0.010	11.63	0.00014	9
Retail	HEP	0.02	0.031	12.00	0.00014	9
Retail	DFHOI	0.02	0.029	12.01	0.00014	9
Retail	AURA-HOI	0.05	0.006	11.41	0.00004	3
Retail	HEP	0.05	0.019	11.90	0.00004	3
Retail	DFHOI	0.05	0.020	11.98	0.00004	3

Output compression. The closed compression ratio is:

$$\mathrm{CCR} = 1 - \frac{|\mathcal{H}_{\mathrm{closed}}|}{|\mathcal{H}_{\mathrm{raw}}|}. \tag{15}$$

The certificate preservation rate is:

$$\mathrm{CPR} = \frac{|\{T(X) : X \in \mathcal{H}_{\mathrm{closed}}\}|}{|\{T(X) : X \in \mathcal{H}_{\mathrm{raw}}\}|}. \tag{16}$$

A correct closed ledger should give CPR = 1.

Efficiency. Report core runtime, peak RSS memory, output disk footprint, visited DFS nodes, support-pruned nodes, envelope-pruned nodes, closure jumps, and ledger duplicates removed.

6.5 Empirical results

Table 4 highlights the main evidence. On Mushrooms at $\alpha = 0.10$, all three miners emit exactly 955 raw HOIs and the same estimated output footprint, so the comparison is not confounded by output semantics. Under this equal-output condition, AURA-HOI is 1.79× faster than HEP and uses 236× less peak RAM. Against DFHOI, AURA-HOI is slightly faster and uses less memory. On Retail, all non-empty settings again match the HEP/DFHOI raw itemset count, while AURA-HOI runs about 2.5–3.0× faster.

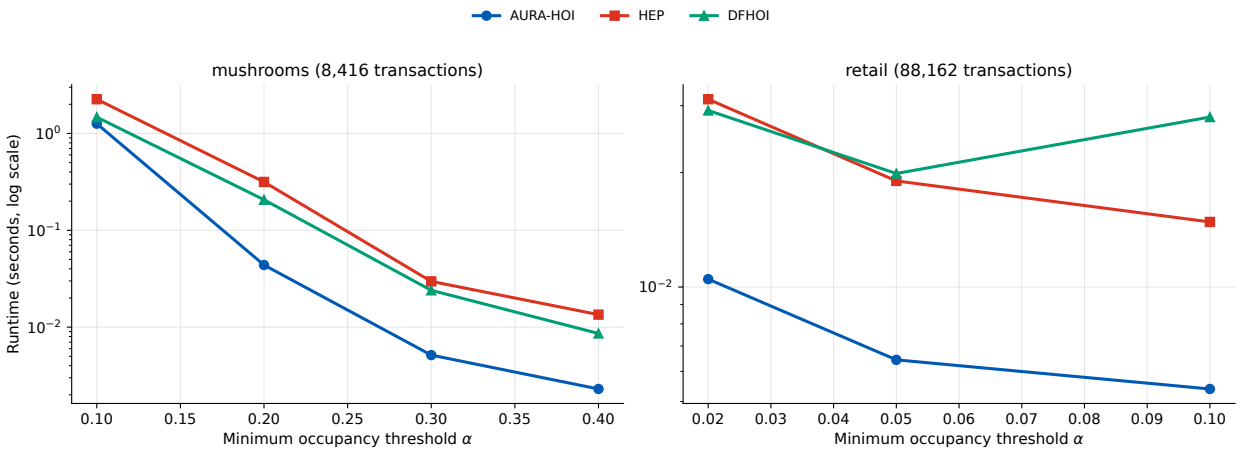


Figure 1: Runtime over occupancy threshold. With raw fullset output and matched summed-occupancy semantics, AURA-HOI consistently runs faster than HEP and is competitive with or faster than DFHOI.

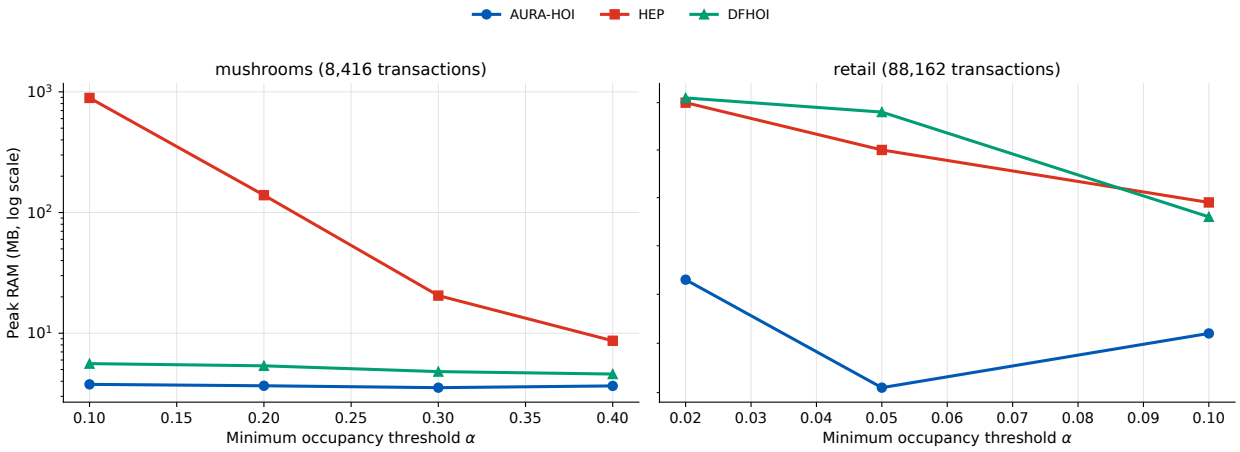


Figure 2: Peak resident memory. The most decisive advantage is on Mushrooms at $\alpha = 0.10$, where AURA-HOI uses 3.77 MB while HEP uses 890.74 MB.

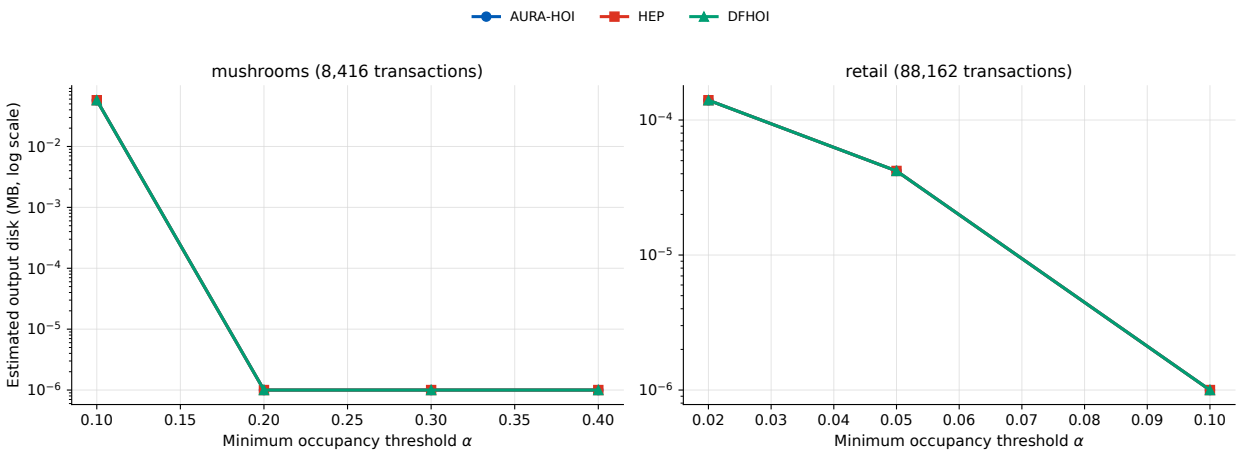
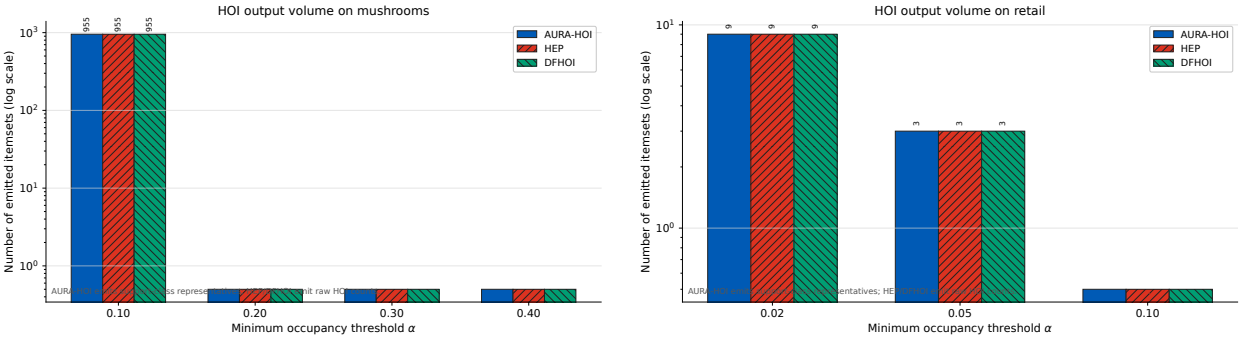


Figure 3: Estimated result disk footprint. Equal disk footprint in non-empty raw-fullset cases confirms that the compared miners emit the same output volume.



(a) Mushrooms (b) Retail

Figure 4: Raw fullset itemset counts. The bars match in all non-empty settings, showing that the runtime and memory gains in Figs. 1 and 2 are achieved without reducing the raw HOI output set.

6.6 Ablation study

7 Reproducibility Protocol

Every experiment must output statistics only by default; pattern details are written only when correctness checking is explicitly requested. The recommended file layout is:

`results/aura_hoi_q1/<dataset>/<algorithm>_<dataset>.txt`

Each file contains all thresholds for one algorithm on one dataset. A single summary CSV should contain one row per $(dataset, algorithm, \sigma_{\min}, \alpha, k, view)$ tuple. The checker must recompute support, exact average occupancy, closure, and support-bitset hashes from the original dataset, not from the miner's internal counters.

8 Conclusion

This paper defined Auditable Representative HOI Mining and proposed AURA-HOI, a new HOI-only algorithm designed to make high-occupancy mining both efficient and scientifically comparable. The central contribution is not merely another pruning bound; it is a unified semantics layer that turns raw, closed, maximal, adaptive, and top-k HOI outputs into auditable views over the same exact occupancy definition. Based on the current `dm` algorithm portfolio, this is the missing bridge needed for Q1-level experiments: counts can be compared only after output normalization, memory and time can be measured across the same threshold grid, and correctness can be certified by support-equivalence classes.

References

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

Mohammed J. Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SIAM International Conference on Data Mining Workshop*, 2002.

Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1997.

Table 5: Ablation variants required to prove each part of AURA-HOI.

Variant	Removed component	Expected effect
AURA-HOI-Full	none	Best balance of compact output, correctness, runtime, and memory.
AURA-HOI-noLedger	support-class ledger	Duplicate support classes reappear; closed validation becomes harder.
AURA-HOI-noResidual	residual occupancy risk envelope	More visited nodes at medium/high α.
AURA-HOI-noClosure	closure jumping	Larger output and deeper DFS tree.
AURA-HOI-rawOnly	closed output disabled	Same correctness under raw view but worse output volume.
AURA-HOI-heapMemory	arena/contiguous temporaries replaced by heap allocation	Higher peak RSS and allocation overhead.