

Exact Top-K Order-Preserving Pattern Mining under Exponential Forgetting

Khac-Chien Nguyen

nkchienster@gmail.com

People's Police University

Nhut-Minh Bui

People's Police University

Cong-Phat Tran

People's Police University

Research Article

Keywords: order-preserving pattern mining, time-series mining, exponential forgetting, top-K pattern mining, branch-and-bound, pattern fusion

Posted Date: October 22nd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11042151/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Exact Top-K Order-Preserving Pattern Mining under Exponential Forgetting

Khac-Chien Nguyen^{1*}, Nhut-Minh Bui¹ and Cong-Phat Tran¹

^{1*}Faculty of Information Technology, People’s Police University, Ho Chi Minh City, Vietnam.

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

Abstract

Order-preserving pattern mining represents recurring time-series trends through relative ranks. Existing forgetting-aware methods require a user-chosen minimum-support threshold, whereas a native global top- K query is harder: after right extension, surviving occurrences move toward more recent endpoints and receive larger exponential weights, so ordinary support anti-monotonicity no longer justifies pruning. We formulate exact top- K forgetting-aware mining for strict OP windows and introduce SP-TK-HJ, a single-shot compatibility-fusion search. The method bounds every admissible descendant depth by evaluating the future positions to which a parent’s occurrences can shift, and it combines the profiles of two compatible parents to reject unpromising fused lineages before occurrence-list alignment. We prove descendant-bound safety, pair-bound safety, complete child realization, preservation of parents required by a future top- K pattern, and global top- K completeness. The Java implementation passes 20,000 ordered differential comparisons against an independently coded raw-window oracle. On six public series with 2,514–2,049,280 observations, the proposed pruning reduces endpoint-alignment comparisons by 60.5–70.9% and yields 5.74–19.48 $\times$ median speedups over the same fusion engine with pruning disabled. A paired ablation shows that the pair-level bound removes a further 7.9–13.0% of alignment work beyond subtree pruning, although its runtime benefit is workload-dependent. Sensitivity experiments vary forgetting strength, K , and maximum pattern length, and controlled scaling reaches $n = 50,000$. The study establishes an exact and reproducible strict-window, single-shot solution; extending the semantics to ties and repeated indexed queries remains future work.

Keywords: order-preserving pattern mining, time-series mining, exponential forgetting, top- K pattern mining, branch-and-bound, pattern fusion

1 Introduction

Order-preserving representations describe a window through the relative order of its observations rather than their absolute amplitudes. This representation is invariant under strictly increasing transformations and has motivated order-preserving matching, indexing, pattern mining, rule mining, approximation, co-occurrence analysis, outlier discovery, and stability-aware mining [1–9]. In applications where recent behavior

should matter more than old behavior, OPF-Miner introduced an exponential forgetting mechanism for OP patterns [10]. Its mining objective, however, remains thresholded: users must provide a minimum forgetting support.

Choosing a useful support threshold is difficult when the score distribution is unknown. OPF-Miner itself identifies parameter-free/top- K forgetting-aware mining as a natural future direction [10]. This makes the problem well motivated,

but not conceptually unforeseen. The unresolved issue is algorithmic: a native global top- K procedure cannot safely inherit ordinary anti-monotone support pruning. With endpoint-based forgetting, a surviving occurrence becomes *more recent* after a right extension and therefore receives a larger weight. The occurrence set can shrink while the total forgetting support increases.

The difficulty is visible in the original running series used throughout this paper,

$$T = (8, 3, 6, 1, 7, 4, 9, 2, 5, 10), \quad n = 10. \quad (1)$$

For $\lambda = 0.15$, pattern $(1, 3, 2)$ occurs at endpoint 6 and has forgetting support 0.548811636094. Its right extension $(1, 3, 2, 4)$ occurs at endpoint 7 and has support 0.637628151622. Thus a descendant is more highly scored than its parent even though it has no additional occurrence. Classical “support cannot increase” pruning is therefore invalid.

This paper addresses a deliberately narrow but useful setting: a *single-shot* query over one raw univariate series, strict (tie-free) OP windows, a requested length interval, and no prebuilt suffix index. We call a query single-shot when mining starts from the raw series and does not assume a previously materialized OP index whose construction cost can be amortized across later queries. This setting matters for ad-hoc or interactive analysis of a newly received series, where only one or a few mining requests may be issued. It complements the OP suffix-tree line, which invests in an index for scalable repeated mining and comparison [11, 12]. We focus on strict windows because the standard OP representation is a permutation only when values are pairwise distinct; weak-order semantics require a different canonical representation, compatibility rule, and completeness proof. The objective also differs from contrast top- K OP mining [13] and top- K pattern mining in other data models [14–16].

Our contributions are as follows.

1. **Problem formulation.** We formalize global top- K strict OP mining under endpoint-based exponential forgetting, with no user-specified minimum support and with a deterministic total order for score ties.
2. **Shift-safe theory.** We derive a shift-evaluated upper bound SEUB_d for every remaining extension depth and its subtree envelope SE-DUB .

The formulation reads precomputed future endpoint weights directly instead of multiplying an old partial sum by $e^{\lambda d}$.

3. **Lineage-aware pair pruning.** For compatible prefix-suffix parents, we derive P-PDUB, which combines both shift profiles and safely rejects a complete fused lineage before occurrence alignment.
4. **Exact single-shot search under an explicit contract.** We integrate the bounds with a compatibility hash join, immutable two-pointer occurrence alignment, child deduplication, and a capacity- K heap. New live-parent and child-realization lemmas close the completeness proof for both prefix and suffix parents.
5. **Reproducible evaluation.** We validate the implementation against an independently coded raw-window oracle and evaluate correctness, pruning, runtime, parameter sensitivity, memory, and scaling on public and controlled synthetic series.

We do *not* claim that top- K is new in OP mining, nor that hash fusion or a top- K heap is itself novel. The solution-level novelty is the forgetting-specific shift bound, its pair-lineage composition, and the proof that these bounds preserve an exact single-shot global top- K search. The rest of the paper is organized as follows. Section 2 positions the work against OP, forgetting-aware, top- K , time-fading stream, and indexed mining. Section 3 formalizes the problem and numerical contract. Sections 4–6 develop the shift-safe bounds and algorithm; Section 7 proves correctness and analyzes complexity. Section 8 reports the experiments, Section 9 discusses limitations and scope, and Section 10 concludes.

2 Related work and positioning

2.1 Order-preserving mining

Order-preserving matching formalized comparison by relative order [1]; later algorithms improved order-isomorphism testing and matching efficiency [17], and OP indexes support repeated matching workloads [2]. OPP-Miner established a fusion-based framework for frequent OP patterns [3]. OPR-Miner, AOP-Miner, co-occurrence OP mining, OUTO-Miner, and SOPP-Miner extend the

family toward rules, approximation, aligned co-occurrence, outlying occurrences, and stable distributions [4–8]. OP ranks should also be distinguished from symbolic time-series representations such as SAX [18]: SAX discretizes normalized amplitudes into symbols, whereas strict OP patterns retain only within-window relative order. These works motivate the inherited notions of rank pattern, occurrence, prefix/suffix compatibility, and pattern fusion, but they do not solve the present top- K forgetting objective.

2.2 Forgetting-aware OP mining

OPF-Miner is the closest semantic predecessor [10]. It weights occurrences exponentially according to recency and uses threshold-oriented strategies, including pattern fusion and pruning, to enumerate patterns meeting a user-provided minimum forgetting support. In a native top- K query, however, the effective threshold is the evolving K -th score, and a pruning rule must remain safe for *all* descendant depths as this threshold rises. More importantly, endpoint shift means that a surviving occurrence changes weight after extension. A thresholded algorithm cannot be converted into an exact global top- K miner merely by replacing a user threshold with the current heap threshold unless its bounds are proved safe under this shift.

Forgetting factors also appear in incremental sequential-pattern mining, for example OIP-Miner [19]; that line addresses one-off occurrences in evolving sequence databases rather than strict order patterns in a single raw time series. It is therefore methodologically relevant to recency weighting but is not a direct baseline for the objective defined here.

Time-decay models are also well established in streaming frequent-item and pattern mining. For example, time-fading stream methods progressively downweight older observations rather than using OP endpoint semantics [20, 21]. These studies reinforce the value of recency-aware mining, but their objects are items/itemsets in streams and their bounds do not face the OP-specific endpoint-shift amplification created by right extension.

2.3 Upper-bound pruning beyond OP mining

A useful methodological comparison comes from high-utility sequential pattern mining (HUSPM), where raw utility is also non-anti-monotone. CRoM bounds the utility remaining after a matched prefix so that candidates can be pruned before generation [22], while HUSP-SP derives the tighter reduced sequence utility (TRSU) bound from a projected representation [23]. Candidate-list maintenance is another strategy for reducing HUSPM search [24]. These methods and the present work share a branch-and-bound philosophy, but the quantities being bounded are different: CRoM/TRSU bound remaining utility in a sequence database, whereas our bounds must compensate for the *increase* in recency weight caused by shifting an OP occurrence’s endpoint under right extension. Consequently, neither HUSPM bound is directly valid for the forgetting score defined here; the connection is conceptual rather than a transferable pruning formula.

For experimental context we therefore distinguish three baselines. BF-Oracle exhaustively enumerates raw windows and is used only for correctness on small inputs. HJ-TopK uses the same compatibility hash fusion as SP-TK-HJ but disables the proposed bounds, isolating their effect. CAL-THRESH-HJ is an *oracle-calibrated* semantic baseline: it receives the exact K -th support in advance and enumerates patterns at or above that threshold. It is intentionally favorable to the threshold route and is not claimed to reproduce the published OPF-Miner implementation.

2.4 Native top- K and indexed OP methods

COPP-Miner uses a native top- K objective for class contrast, showing that top- K ranking itself is not new in OP mining [13]. Top- K threshold raising also has a long history in itemset and sequential/co-occurrence pattern mining [14–16]. Their scores and upper bounds differ from forgetting support and do not address endpoint-shift amplification.

A separate line constructs scalable OP suffix structures for frequent, maximal, closed, and comparison queries [11, 12]. Such indexes are compelling for repeated queries or very large archives.

SP-TK-HJ instead targets the one-query/raw-series case and includes no index-construction phase. Any direct runtime comparison between these settings must charge index construction for a single-shot query or explicitly analyze amortized reuse.

In summary, prior studies separately cover thresholded forgetting-aware OPP, native top- K contrast OPP, time-fading stream statistics, and indexed frequent/maximal/closed OP mining. To the best of our literature search, none combines exact global top- K ranking under endpoint-based forgetting with a depth-aware descendant bound that follows endpoint shift and a compatible-pair lineage bound that can prune before occurrence alignment in the raw-series single-shot setting. This is the specific gap addressed below.

3 Preliminaries and problem statement

3.1 Strict OP patterns and occurrences

Let $T = (t_1, \dots, t_n) \in \mathbb{R}^n$. A length- m window $x = (x_1, \dots, x_m)$ is *strict* if its values are pairwise distinct. Following standard OP semantics [1, 3, 10], its normalized pattern is the permutation

$$\text{norm}(x)_i = 1 + |\{j \in \{1, \dots, m\} : x_j < x_i\}|. \quad (2)$$

For strict x , $\text{norm}(x)$ belongs to S_m , the set of permutations of $\{1, \dots, m\}$. For an observed strict pattern p , define its 1-based endpoint occurrence set

$$\text{Occ}(p) = \{e \in \{|p|, |p| + 1, \dots, n\} : \text{norm}(t_{e-|p|+1}, \dots, t_e) = p\}. \quad (3)$$

Tied windows are outside the search space studied here. Section 9 quantifies how this restriction affects each public dataset.

3.2 Forgetting support

For a finite decay rate $\lambda \geq 0$, the endpoint weight inherited from forgetting-aware OP mining [10] is

$$w_e(\lambda) = \exp[-\lambda(n - e)], \quad 1 \leq e \leq n. \quad (4)$$

The forgetting support of p is

$$F_\lambda(p) = \sum_{e \in \text{Occ}(p)} w_e(\lambda). \quad (5)$$

For finite $\lambda \geq 0$, $0 < w_e(\lambda) \leq 1$ over the reals and $w_n = 1$. We extend the positive-decay domain commonly used by the predecessor to include $\lambda = 0$ as a base case for validation; all bounds below hold for every finite $\lambda \geq 0$. When $\lambda = 0$, every weight is one and F_λ reduces to ordinary occurrence count.

Definition 1 (TK-OPF-S) Given a finite series T , a finite $\lambda \geq 0$, $K \geq 1$, and $2 \leq \ell_{\min} \leq \ell_{\max} \leq n$, let $\mathcal{P}$ contain every observed strict OP pattern whose length lies in $[\ell_{\min}, \ell_{\max}]$. For equal-length integer rank vectors, $p <_{\text{lex}} q$ if at the first position r at which they differ, $p_r < q_r$. Define $p \succ q$ iff (i) $F_\lambda(p) > F_\lambda(q)$; or (ii) the scores are equal and $|p| < |q|$; or (iii) score and length are equal and $p <_{\text{lex}} q$. Return the first $\min(K, |\mathcal{P}|)$ patterns under $\succ$.

Proposition 1 (Well-defined output order) *The relation $\succ$ is a strict total order on the finite candidate set $\mathcal{P}$.*

Proof Patterns are first ordered by a scalar score in descending order. Equal-score patterns are then ordered by integer length, and equal-score equal-length patterns are ordered lexicographically by their distinct rank vectors. Each component order is strict and transitive, and every pair of distinct candidates differs in at least one component. Their lexicographic composition is therefore irreflexive, transitive, and total on distinct candidates. $\square$

The capacity- K heap threshold θ is zero until the heap is full; thereafter it is the primary score of the current worst heap element. Because secondary tie-breaks are part of the output order, a branch with upper bound exactly θ cannot be discarded solely on its primary score.

3.3 Declared numerical contract

The mathematical inequalities below are first stated over nonnegative real weights. The reference implementation then instantiates them under a reproducible Java-21 binary64 contract:

1. all observations and λ must be finite and $\lambda \geq 0$;

Table 1 Positioning relative to representative prior work. “Threshold” denotes a user-specified support-type threshold; query mode distinguishes raw-series single-shot mining from index-amortized repeated querying.

Method line	Object / score	Forgetting	Native top- K	Tie model	Generation	Main pruning / bound	Index	Query mode
OPP-Miner [3]	frequent OP	no	no	strict	fusion	support-based	no	single-shot
OPP-Miner [10]	forgetting support	yes	no	strict	fusion	threshold-oriented OPF pruning	no	single-shot
COPP-Miner [13]	class contrast	no	yes	strict	fusion	contrast-specific	no	single-shot
OUTO-Miner [7]	outlying OP occurrences	no	no	strict	fusion	maximal-pattern screening	no	single-shot
SOPP-Miner [8]	stable-distribution OP	no	no	strict	fusion	stability/frequency pruning	no	single-shot
Time-fading streams [20, 21]	decayed items/itemssets	yes	varies	n/a	stream summary	decay/sketch bounds	stream state	continuous
HUSPM bounds [22, 23]	sequence utility	no	no	n/a	sequence extension	CRoM/TRSU utility bounds	no	database query
OPST line [11, 12]	frequent/maximal/closed OP	no	no	strict	suffix structure	index traversal	yes	repeated
SP-TK-HJ	global forgetting rank	yes	yes	strict	compatibility fusion	shift-depth + pair-lineage	no	single-shot

2. -0.0 and $+0.0$ observations are canonicalized to $+0.0$ before ranking;
3. endpoint weights are computed by **StrictMath.exp**;
4. support and bound sums are left folds over increasing endpoint order;
5. no epsilon is used; a pruning test is strictly $UB < \theta$.

Binary64 is used deliberately because the contribution under study is search-space pruning rather than arbitrary-precision transcendental arithmetic, and every evaluated algorithm uses the same operational score. **StrictMath.exp** plus a fixed endpoint-sum order makes the Java contract reproducible. **BigDecimal** alone would not specify an exact exponential function, while compensated summation would define a different operational score rather than certify the real-valued ordering. Certified interval or multiprecision fallback for adversarial near-ties is therefore treated as a separate extension. Underflow of an extremely old weight to $+0.0$ is part of this binary64 contract. We do not claim correctly rounded arbitrary-precision ranking for adversarial near-ties.

3.4 Unified running example

For Eq. (1), set $\lambda = 0.15$, $K = 3$, $\ell_{\min} = 2$, and $\ell_{\max} = 5$. The endpoint weights are

$$\begin{aligned}
 (w_1, \dots, w_5) &= (0.259240, 0.301194, \\
 &\quad 0.349938, 0.406570, 0.472367), \\
 (w_6, \dots, w_{10}) &= (0.548812, 0.637628, \\
 &\quad 0.740818, 0.860708, 1.000000).
 \end{aligned}$$

Exhaustive enumeration yields Table 2. The same series will be used in every subsequent example of shifting, profiles, pair pruning, and fusion.

Table 2 Top-3 for the unified running example.

Rank	Pattern	Endpoints	F_λ
1	(1, 2)	{3, 5, 7, 9, 10}	3.320640430
2	(2, 1)	{2, 4, 6, 8}	1.997393728
3	(3, 1, 2)	{3, 9}	1.210645726

4 Why ordinary support pruning fails

Lemma 1 (Endpoint shift) *For every endpoint e and integer $d \geq 0$ with $e + d \leq n$,*

$$w_{e+d} = \exp(\lambda d) w_e. \quad (6)$$

Proof By Eq. (4),

$$\begin{aligned}
 w_{e+d} &= \exp[-\lambda(n - (e + d))] \\
 &= \exp[-\lambda(n - e) + \lambda d] \\
 &= \exp(\lambda d) w_e.
 \end{aligned}$$

□

A right extension removes some occurrences but shifts every surviving endpoint d positions to the right. Hence occurrence-count anti-monotonicity does not imply forgetting-support anti-monotonicity. In the running example, $p = (1, 3, 2)$ has $\text{Occ}(p) = \{6\}$ and $F_\lambda(p) = w_6 = 0.548811636094$, whereas its depth-1 right extension $x = (1, 3, 2, 4)$ has $\text{Occ}(x) = \{7\}$ and $F_\lambda(x) = w_7 = 0.637628151622 > F_\lambda(p)$. A safe top- K bound must therefore account for every possible endpoint shift.

5 Shift-safe descendant profiles

A pattern x is a depth- d right descendant of p iff $|x| = |p| + d$ and $\text{norm}(x_1, \dots, x_{|p|}) = p$. This formalizes “order-isomorphic prefix” independently of any particular fusion path.

Definition 2 (Shift-evaluated depth bound) For observed parent p and feasible $d \geq 0$, define

$$\text{SEUB}_d(p) = \sum_{\substack{o \in \text{Occ}(p) \\ o+d \leq n}} w_{o+d}. \quad (7)$$

Unlike multiplying a partial parent sum by $\exp(\lambda d)$, Eq. (7) reads only precomputed weights in $[0, 1]$ and avoids a potentially huge multiplier times a tiny partial sum.

Theorem 1 (Depth safety over the reals) *For every depth- d right descendant x of p ,*

$$F_\lambda(x) \leq \text{SEUB}_d(p). \quad (8)$$

Proof Every occurrence of x ending at j contains the unique length- $|p|$ prefix window ending at $o = j - d$. By the definition of a depth- d right descendant, that prefix normalizes to p . Hence $j \mapsto o = j - d$ is injective and

$$\text{Occ}(x) \subseteq \{o + d : o \in \text{Occ}(p), o + d \leq n\}.$$

Because all endpoint weights are nonnegative, the sum over this subset cannot exceed the sum over the shifted parent endpoint set in Eq. (7). $\square$

Corollary 1 (Safety under the Java binary64 contract) *The inequality in Theorem 1 is preserved by the reference implementation’s ordered binary64 accumulation.*

Proof Each stored endpoint weight returned by `StrictMath.exp` is finite and lies in $[0, 1]$; underflow to $+0.0$ is permitted. The implementation is array-backed, so $n \leq \text{Integer.MAX_VALUE}$ and every support/bound sum is at most n , far below binary64 overflow. Both child and bound traverse endpoints in increasing order. For finite nonnegative binary64 values, round-to-nearest addition is monotone in each operand: if $a \leq b$ and $c \geq 0$, then $\text{fl}(a+c) \leq \text{fl}(b+c)$. Scanning the union of shifted endpoints, the bound either adds the same term as the child or an additional nonnegative term. Induction over this common endpoint order preserves the child-sum $\leq$ bound-sum relation. $\square$

Proposition 2 (Exact slack and tightness of the depth bound) *Let*

$$S_d(p) = \{o + d : o \in \text{Occ}(p), o + d \leq n\}.$$

For any depth- d descendant x of p ,

$$\begin{aligned} & \text{SEUB}_d(p) - F_\lambda(x) \\ &= \sum_{j \in S_d(p) \setminus \text{Occ}(x)} w_j. \end{aligned} \quad (9)$$

Over the real-valued model with finite λ , the bound is tight, i.e., $\text{SEUB}_d(p) = F_\lambda(x)$, if and only if $\text{Occ}(x) = S_d(p)$.

Proof Theorem 1 gives $\text{Occ}(x) \subseteq S_d(p)$. Subtracting the child subset sum from the bound’s superset sum leaves exactly the weights associated with shifted parent occurrences that do not survive in x . For finite λ every real-valued endpoint weight is strictly positive, so this slack is zero exactly when no shifted endpoint is lost. $\square$

Proposition 2 characterizes when the depth bound is informative. It is tight when a descendant preserves every feasible shifted occurrence of its parent and becomes progressively looser as more of those occurrences disappear. The subtree envelope additionally maximizes over depths, so it can be loose when its maximizing depth differs from the lineage depth that eventually realizes a descendant.

Remark 1 (Large depths) If no $o \in \text{Occ}(p)$ satisfies $o + d \leq n$, Eq. (7) is the empty sum and equals zero. Structurally, no observed depth- d right descendant can then inherit an occurrence from p . Thus deep profiles naturally collapse as shifted parent endpoints move beyond the series boundary; no special heuristic is required.

Definition 3 (Shift profile and subtree envelope) If a state p has remaining depth budget $D = \ell_{\max} - |p|$, cache

$$\begin{aligned} P_p[d] &= \text{SEUB}_d(p), \quad d = 0, \dots, D, \\ \text{SE-DUB}_D(p) &= \max_{0 \leq d \leq D} P_p[d]. \end{aligned} \quad (10)$$

Corollary 2 (Safe subtree pruning) *If the heap is full and $\text{SE-DUB}_D(p) < \theta$, no allowed right descendant of p can enter the final top- K set through p as a prefix lineage.*

The strict inequality is necessary: if the upper bound equals θ , an equal-score pattern may still win by length or lexicographic tie-break.

5.1 Compatible pairs and pair-lineage pruning

For a strict length- m pattern $p = (p_1, \dots, p_m)$, let

$$\begin{aligned} \text{pre}(p) &= \text{norm}(p_1, \dots, p_{m-1}), \\ \text{suf}(p) &= \text{norm}(p_2, \dots, p_m). \end{aligned} \quad (11)$$

Two length- m patterns p, q are compatible when $\text{suf}(p) = \text{pre}(q)$, as in fusion-based OP mining [3, 10].

Theorem 2 (Pair-lineage profile bound) *Let p, q be compatible length- m patterns. Let x be obtained from one of their length- $(m+1)$ fused children followed by $d \geq 0$ further right extensions. Then*

$$F_\lambda(x) \leq \min\{P_p[d+1], P_q[d]\}. \quad (12)$$

Consequently, for $D = \ell_{\max} - (m+1)$,

$$\begin{aligned} \text{P-PDUB}_D(p, q) \\ = \max_{0 \leq d \leq D} \min\{P_p[d+1], P_q[d]\} \end{aligned} \quad (13)$$

is a safe pre-alignment upper bound for the complete pair lineage.

Proof Consider an occurrence of x ending at j . Its length- m prefix parent p ends at $j - (d+1)$, while the overlapping length- m suffix parent q ends at $j - d$. Both mappings are injective. Hence the child endpoints are subsets of the corresponding shifted endpoint sets represented by $P_p[d+1]$ and $P_q[d]$. Nonnegative summation bounds the score by each profile value, and therefore by their minimum. Maximizing this minimum over all allowed d yields Eq. (13). $\square$

Proposition 3 (Relation to single-parent envelopes) *For a compatible pair and the same feasible depth range,*

$$\begin{aligned} \text{P-PDUB}_D(p, q) \\ \leq \min\left\{\max_{1 \leq r \leq D+1} P_p[r], \max_{0 \leq r \leq D} P_q[r]\right\}. \end{aligned}$$

The inequality need not be strict; therefore pair pruning is safe but not guaranteed to add pruning on every workload.

Proof For every d , $\min\{P_p[d+1], P_q[d]\}$ is no larger than either argument. Taking maxima on the left and the corresponding maxima on the right proves the result. $\square$

Proposition 4 (When pair pruning is strictly additional) *For a full heap threshold θ and pair depth budget D , define the promising-depth sets*

$$\begin{aligned} H_p(\theta) &= \{d \in \{0, \dots, D\} : P_p[d+1] \geq \theta\}, \\ H_q(\theta) &= \{d \in \{0, \dots, D\} : P_q[d] \geq \theta\}. \end{aligned}$$

Then $\text{P-PDUB}_D(p, q) < \theta$ if and only if $H_p(\theta) \cap H_q(\theta) = \emptyset$. In particular, P-PDUB performs strictly additional pruning beyond the two individual parent tests exactly when both $H_p(\theta)$ and $H_q(\theta)$ are nonempty but their intersection is empty.

Proof By Eq. (13), $\text{P-PDUB}_D(p, q) \geq \theta$ holds exactly when there exists a depth d for which both $P_p[d+1] \geq \theta$ and $P_q[d] \geq \theta$. This is equivalent to a nonempty intersection of the two promising-depth sets. If each set is nonempty, neither parent envelope alone is below θ ; an empty intersection therefore identifies precisely the cases eliminated only after the two profiles are aligned by depth. $\square$

This characterization explains why P-PDUB can be valuable even though Proposition 3 guarantees no universal runtime gain: its opportunity depends on a *depth mismatch* between the promising regions of the two parents. For fixed profiles and threshold, increasing D can only enlarge the maximization domain in Eq. (13), so the pair bound cannot become smaller merely because a larger $\ell_{\max}$ is allowed. Likewise, for a fixed candidate set the final K -th score is nonincreasing with K ; larger K therefore weakens score-based pruning in general. No monotonic pruning law is claimed for λ , because changing λ changes both every profile and the top- K threshold.

5.1.1 Running pair trace

For $p = (3, 1, 2)$ and $q = (2, 3, 1)$,

$$\begin{aligned} \text{Occ}(p) &= \{3, 9\}, \\ P_p &= [1.210646, 1.406570, 0.472367], \end{aligned}$$

$$\begin{aligned} \text{Occ}(q) &= \{4, 8\}, \\ P_q &= [1.147388, 1.333075, 1.548812]. \end{aligned}$$

They are compatible because $\text{suf}(p) = \text{pre}(q) = (2, 1)$. With $m = 3$ and $\ell_{\max} = 5$, $D = 1$, so

$$\begin{aligned} \text{P-PDUB}_1(p, q) &= \max\{\min(1.406570, 1.147388), \\ &\quad \min(0.472367, 1.333075)\} \\ &= 1.147387880. \end{aligned}$$

Once the heap reaches the final running-example threshold $\theta = 1.210645726$, this pair can be rejected before alignment. Without this rejection, aligning endpoint pairs (3, 4) and (9, ·) realizes only child (4, 2, 3, 1) at endpoint 4 with score 0.406569660.

6 SP-TK-HJ algorithm

6.1 Compatibility hash join and child realization

A direct level-wise fusion would compare all pairs. SP-TK-HJ indexes each live pattern q by $\text{pre}(q)$ and probes the bucket keyed by $\text{suf}(p)$.

Lemma 2 (Compatibility-hash completeness) *A multimap keyed by $\text{pre}(q)$ and probed by $\text{suf}(p)$ emits all and only compatible parent pairs.*

Proof Canonical rank vectors are equal exactly when the shared $m - 1$ positions are order-isomorphic. Therefore compatibility is equivalent to hash-key equality. $\square$

For a compatible pair, the canonical fused templates can be constructed without inspecting occurrence lists. Algorithm 1 makes the standard OP fusion step explicit [3, 10]. The occurrence lists are then merged by two pointers. A prefix-parent occurrence ending at e and a suffix-parent occurrence ending at $e + 1$ define exactly one candidate length- $(m + 1)$ raw window. If two templates are possible, comparing the two non-overlapping raw observations (the first and last values of the new window) selects the strict child; equality rejects the window.

Lemma 3 (Child-realization completeness) *For a compatible pair p, q , ALIGNFUSE returns exactly the observed strict length- $(m + 1)$ children whose prefix pattern is p and suffix pattern is q , with complete sorted occurrence lists.*

Proof sketch Every valid child occurrence induces one adjacent prefix/suffix endpoint pair, and every such adjacent pair identifies one raw length- $(m + 1)$ window. Compatibility fixes all comparisons except the first-versus-last relation in the ambiguous case, which is

Algorithm 1 FUSETEMPLATES(p, q) for compatible length- m rank vectors

```

1:  $a \leftarrow p_1; b \leftarrow q_m$ 
2: if  $a < b$  then
3:    $c_1 \leftarrow a; c_{m+1} \leftarrow b + 1$ 
4:   for  $i = 2$  to  $m$  do
5:      $c_i \leftarrow p_i + \mathbf{1}[p_i > b]$ 
6:   end for
7:   return  $\{c\}$ 
8: else if  $a > b$  then
9:    $c_1 \leftarrow a + 1; c_{m+1} \leftarrow b$ 
10:  for  $i = 1$  to  $m - 1$  do
11:     $c_{i+1} \leftarrow q_i + \mathbf{1}[q_i > a]$ 
12:  end for
13:  return  $\{c\}$ 
14: else
15:   $r \leftarrow a; c_1^- \leftarrow r; c_{m+1}^- \leftarrow r + 1$ 
16:   $c_1^+ \leftarrow r + 1; c_{m+1}^+ \leftarrow r$ 
17:  for  $i = 2$  to  $m$  do
18:     $c_i^- \leftarrow p_i + \mathbf{1}[p_i > r]$ 
19:     $c_i^+ \leftarrow c_i^-$ 
20:  end for
21:  return  $\{c^-, c^+\}$ 
22: end if

```

resolved from the raw observations. Appendix A gives the complete necessity/sufficiency argument. $\square$

6.2 Top-K search

Algorithm 3 fixes the profile-lifecycle ambiguity that arises if a bound is used before being computed: every live state stores its profile, and a child profile is built exactly once when the unique child is first materialized. A child map deduplicates patterns generated from different parent pairs and asserts occurrence-list consistency.

The implementation also includes SP-DUB, which disables pair pruning, and SP-Gated, which evaluates P-PDUB only when its $O(D)$ profile scan is no larger than the simple proxy $|\text{Occ}(p)| + |\text{Occ}(q)|$ for upcoming two-list alignment. Skipping a safe bound cannot remove a candidate and is therefore correctness neutral.

7 Correctness and complexity

Lemma 4 (Heap-threshold monotonicity) *Once the capacity- K heap is full, its worst primary score θ never decreases.*

Algorithm 2 ALIGNFUSE(T, p, q)

Require: Compatible length- m states p, q **Require:** sorted endpoint lists

```
1:  $C \leftarrow \text{FUSETEMPLATES}(p, q)$ 
2: empty endpoint list for each template in  $C$ 
3:  $i \leftarrow 1, j \leftarrow 1$ 
4: while  $i \leq |\text{Occ}(p)|$  and  $j \leq |\text{Occ}(q)|$  do
5:    $e_p \leftarrow \text{Occ}(p)[i]; e_q \leftarrow \text{Occ}(q)[j]$ 
6:   if  $e_q = e_p + 1$  then
7:     if fusion has one template then
8:       append  $e_q$  to that child
9:     else
10:      compare  $t_{e_p-m+1}$  with  $t_{e_q}$ 
11:      append  $e_q$  to the matching strict child
12:      discard the window on equality
13:    end if
14:     $i \leftarrow i + 1; j \leftarrow j + 1$ 
15:  else if  $e_q < e_p + 1$  then
16:     $j \leftarrow j + 1$ 
17:  else
18:     $i \leftarrow i + 1$ 
19:  end if
20: end while
21: return nonempty realized children
```

Proof A replacement occurs only when the new pattern is better than the current worst element under the full total order. Its primary score is therefore greater than or equal to the removed score. If replacement is caused only by a secondary tie-break, the primary threshold remains equal. $\square$

Lemma 5 (Live-parent lineage safety) *Let p, q be compatible length- m parents and let x be any pattern produced from one of their fused children followed by $d \geq 0$ right extensions. Then*

$$\begin{aligned} F_\lambda(x) &\leq P_p[d+1] \leq \text{SE-DUB}_{D_p}(p), \\ F_\lambda(x) &\leq P_q[d] \leq \text{SE-DUB}_{D_q}(q). \end{aligned} \quad (14)$$

Thus a strict parent test $\text{SE-DUB} < \theta$ cannot remove either parent of a future fused lineage whose score is at least the current θ .

Proof The two inequalities on the left are the two injective endpoint mappings used in Theorem 2, considered separately. The relevant depths lie inside each cached profile because $d \leq \ell_{\max} - (m+1)$, so $d+1 \leq \ell_{\max} - m$. Each profile value is bounded above by its envelope. The pruning conclusion follows immediately. $\square$

Theorem 3 (Top-K completeness) *Under the declared scoring/comparison contract, Algorithm 3 returns*

Algorithm 3 SP-TK-HJ($T, \lambda, K, \ell_{\min}, \ell_{\max}$)

Require: finite T ; finite $\lambda \geq 0$; $K \geq 1$; $2 \leq \ell_{\min} \leq \ell_{\max} \leq n$

```
1: validate input; precompute  $w_1, \dots, w_n$  with
   StrictMath.exp
2:  $F \leftarrow$  observed strict length-2 states
3:  $H \leftarrow$  capacity- $K$  heap by the output order
4: for all  $p \in F$  do
5:    $D_p \leftarrow \ell_{\max} - |p|$ ; build and cache  $P_p[0..D_p]$ 
6: end for
7: offer length-2 states to  $H$  if  $\ell_{\min} = 2$ 
8: discard  $p \in F$  only if  $H$  is full and  $\text{SE-DUB}_{D_p}(p) < \theta$ 
9: for  $m = 2$  to  $\ell_{\max} - 1$  while  $F \neq \emptyset$  do
10:  build  $I[\text{pre}(q)]$  for live  $q \in F$ ;  $N \leftarrow \emptyset$ 
11:  for all  $p \in F$  do
12:    for all  $q \in I[\text{suf}(p)]$  do
13:       $D_{pq} \leftarrow \ell_{\max} - (m+1)$ 
14:      if  $H$  full and  $\text{P-PDUB}_{D_{pq}}(p, q) < \theta$  then
15:        continue
16:      end if
17:      for all  $r \in \text{ALIGNFUSE}(T, p, q)$  do
18:        if  $|r| \geq \ell_{\min}$  then
19:          offer  $r$  to  $H$ ; update  $\theta$ 
20:        end if
21:        if  $|r| < \ell_{\max}$  then
22:           $D_r \leftarrow \ell_{\max} - |r|$ ; build  $P_r[0..D_r]$  once
23:          if  $H$  not full or  $\text{SE-DUB}_{D_r}(r) \geq \theta$  then
24:            merge unique  $r$  into  $N$ 
25:          end if
26:        end if
27:      end for
28:    end for
29:  end for
30:   $F \leftarrow \text{values}(N)$ 
31: end for
32: return  $H$  sorted by the output order
```

exactly the global top- K observed strict patterns in $[\ell_{\min}, \ell_{\max}]$.

Proof sketch Soundness follows from observed length-2 seeds and Lemma 3. Without pruning, induction on pattern length plus Lemmas 2–3 generates every observed strict pattern. With pruning, Lemma 4, Lemma 5, and Theorem 2 ensure that no lineage capable of reaching a final top- K score is removed before realization. The capacity- K heap then selects exactly the best generated patterns under the total order. The full induction and threshold argument are given in Appendix A. $\square$

7.1 Complexity

Let $\mathcal{F}_m$ be the live length- m states, $L_m = |\mathcal{F}_m|$, and

$$A_m = \sum_{p \in \mathcal{F}_m} |\text{Occ}(p)|, \quad D_m = \ell_{\max} - m.$$

Let $\mathcal{J}_m$ be the set of compatible pairs emitted by the hash join and $\mathcal{B}_m \subseteq \mathcal{J}_m$ the subset rejected by P-PDUB. Let $C_{\text{align}}(p, q) = O(|\text{Occ}(p)| + |\text{Occ}(q)|)$ denote the two-pointer alignment cost. Direct profile construction scans each occurrence for each

remaining depth, so its worst-case work is

$$C_{\text{profile}} = O\left(\sum_m D_m A_m\right). \quad (15)$$

Once profiles exist, all pair-bound scans cost

$$O\left(\sum_m |\mathcal{J}_m| D_m\right). \quad (16)$$

The non-pruned alignment work is

$$O\left(\sum_m \sum_{(p,q) \in \mathcal{J}_m \setminus \mathcal{B}_m} C_{\text{align}}(p, q)\right). \quad (17)$$

Hash-key construction contributes $O(\sum_m m L_m)$ if rank-vector keys are materialized, and S heap offers cost $O(S \log K)$. Building a hash table requires only $O(L_m)$ references at a level, but hashing does not improve the worst-case number of compatible pairs: if a key bucket is dense, $|\mathcal{J}_m|$ can be $\Theta(L_m^2)$. Thus HJ-TopK and SP-TK-HJ share the same worst-case search-space order. The gain comes from reducing live states through subtree pruning and avoiding the alignment term for pairs in $\mathcal{B}_m$; when $\mathcal{B}_m$ is large or occurrence lists are long, this removes substantial two-pointer work.

At peak level transition, current and next states coexist. Let L_{m+1} and A_{m+1} denote the number and total occurrences of the next-level states and $D_{m+1} = \ell_{\max} - (m+1)$. The implementation stores current and next occurrence lists, rank-vector keys, one cached profile per live state, $O(L_m)$ hash-bucket references, a next-level child map, and a K -element heap. A conservative bound is

$$\begin{aligned} O(&A_m + A_{m+1} \\ &+ L_m(m + D_m + 1) \\ &+ L_{m+1}(m + 1 + D_{m+1}) \\ &+ K \ell_{\max}) \end{aligned} \quad (18)$$

up to object/header overhead. This corrects a simpler bound that counted only current-level profiles. Process-level RSS and JVM-used-memory measurements are reported empirically in Section 8.

Proposition 5 (Coarse worst-case bound in n , K , and $\ell_{\max}$) *Let $L = \ell_{\max}$. Because every strict length- m window belongs to exactly one observed length- m pattern, $A_m \leq n - m + 1$ and $L_m \leq n - m + 1$. Under the level-wise implementation, a coarse worst-case bound is*

$$\begin{aligned} O(&nL^2 + n^2L^2 \\ &+ n^3L + nL \log K) \end{aligned} \quad (19)$$

for time and

$$O((n + K)L) \quad (20)$$

for resident algorithmic state, excluding JVM object/header overhead and the input array. The decay λ changes scores and practical prune rates but not this asymptotic worst case.

Proof The profile term is at most $O(nL^2)$ because $A_m = O(n)$ and the remaining-depth budgets sum to $O(L^2)$. At each level the compatibility join can emit $O(n^2)$ pairs; scanning their depth profiles contributes $O(n^2L^2)$ over all levels, and in the pessimistic case each surviving pair aligns two $O(n)$ occurrence lists, giving $O(n^3L)$. At most $O(nL)$ distinct observed states can be offered to the heap, adding $O(nL \log K)$. At a level transition the total occurrence membership at each of the two adjacent lengths is $O(n)$, while rank vectors and cached profiles use $O(nL)$ and the heap stores at most K patterns of length at most L . $\square$

The worst case is attained in the algorithmic sense when the heap is not yet selective, or when every live subtree and compatible pair has an upper bound at least θ . Then $\mathcal{B}_m$ is empty and no live state is removed by the proposed bounds, so SP-TK-HJ degenerates to the same observed-pattern enumeration and occurrence alignment performed by HJ-TopK, plus bound-evaluation overhead. Equations (15)–(18) remain the more informative data-sensitive characterization; Proposition 5 is deliberately coarse and is included to make the dependence on n , K , and L explicit.

8 Experimental evaluation

8.1 Research questions and artifact

We study:

- **RQ1: Correctness.** Do the proposed variants return the identical *ordered top- K* result as an independently coded raw-window oracle on small instances?

- **RQ2: Pruning effectiveness.** How much do shift profiles reduce fusion/alignment work relative to HJ-TopK?
- **RQ3: Pair-bound value.** Does P-PDUB add useful pruning, and what is its overhead?
- **RQ4: Runtime/scaling.** How do the methods scale with n and K under controlled multi-seed synthetic data?
- **RQ5: Scope effects.** How large is strict-window rejection on real datasets, and what memory footprint is observed?
- **RQ6: Parameter sensitivity.** Does the pruning advantage persist as the normalized decay λn , maximum pattern length, and requested K vary?
- **RQ7: Practical interpretation and guidance.** What do representative returned patterns look like, and how should λ , K , and $\ell_{\max}$ be chosen in a new analysis?

The core artifact has no third-party Java dependency and is compiled with `javac --release 21`. Endpoint weights are evaluated with Java `StrictMath`; no epsilon-based pruning is used. Public performance cells use 10 warmups and 20 measured repetitions, with randomized method order within each repetition. We report medians together with the first and third quartiles (Q_1 , Q_3); the machine-readable summaries also contain percentile-bootstrap 95% intervals. The six-dataset public campaign ran on a Microsoft Surface Laptop (7th Edition) with a 10-core ARM64 Snapdragon X X1P64100 processor (up to 3.40 GHz), 15.6 GiB RAM, Windows build 26200.9445, Oracle JDK 24.0.1, and `-Xms4g -Xmx4g`; the class files retain the Java 21 target. Two auxiliary Linux/KVM archives use OpenJDK 21.0.11, 5.8 GiB RAM, and a 4 GiB maximum heap, but they are not the same virtual CPU: the one-shot process-RSS campaign and the oracle-calibrated threshold cells ran on five Intel Xeon E5-2673 v4 vCPUs, whereas the revision synthetic-scaling and parameter-sensitivity campaigns ran on five AMD EPYC 9V74 vCPUs. Absolute timings are compared only within a campaign and machine. Scripts used to summarize tables and generate figures were prepared with assistance from large language models; all reported numbers were recomputed from archived CSVs and independently inspected by the authors.

8.2 Datasets and strict-window coverage

The public snapshots comprise FRED S&P 500 and NASDAQ Composite [25, 26], SILSO daily total sunspot number v2.0 [27], UCI Beijing PM2.5 [28], temperature from the Jena Climate archive recorded by the Max Planck Institute for Biogeochemistry [29, 30], and UCI Individual Household Electric Power Consumption [31]. We select the PM2.5, temperature (degrees Celsius), and global active power (kW) variables from the multivariate sources. Missing FRED markers, negative SILSO missing codes, missing PM2.5 values, and missing power values are removed. No interpolation, resampling, normalization, or tie-breaking jitter is applied. For each source, the public repository documentation records the identifier/URL, sampling description, missing-value handling, and licensing information; raw/processed SHA-256 checksums are retained for bundled snapshots [32]. Table 3 reports the resulting sample sizes and strict-window coverage. The strict-window fraction at length 12 ranges from 17.70% for household power to 99.28% for S&P 500. SILSO, Beijing PM2.5, and household power are therefore challenging for the strict-order contract because ties reject many raw windows. Their absolute runtime must not be interpreted as if every window entered the OP search space.

8.3 Algorithms

The evaluated implementations are:

- **BF-Oracle:** independent raw-window exhaustive enumeration, used only for correctness on small instances;
- **HJ-TopK:** compatibility hash fusion plus top- K heap, with all proposed bounds disabled;
- **SP-DUB:** HJ-TopK plus shift-profile subtree pruning;
- **SP-Full:** SP-DUB plus P-PDUB pre-alignment pruning;
- **SP-Gated:** optional SP-Full variant that skips pair-bound evaluation when a simple cost proxy predicts no advantage;

Table 3 Public datasets and strict-window coverage. Adjacent-tie rate is computed on consecutive observations; strict fraction is the fraction of length-12 windows with all values distinct.

Dataset	Domain	n	Distinct values	Adjacent-tie rate	Strict fraction ($m = 12$)
S&P 500	Finance	2,514	2,507	0.040%	99.28%
NASDAQCOM	Finance	14,014	13,486	0.093%	95.90%
SILSO	Solar activity	72,967	437	18.687%	20.05%
Beijing PM2.5	Air quality	41,757	581	4.919%	31.36%
Jena temperature	Meteorology	420,451	5,530	4.183%	52.72%
Household power	Energy	2,049,280	4,186	20.155%	17.70%

- **CAL-THRESH-HJ**: an oracle-calibrated threshold collector supplied with the true K -th support; it is a semantic baseline, not the official OPF-Miner implementation.

A direct runtime comparison with COPP-Miner would be invalid because the objective is class contrast. OPST methods solve an indexed setting; comparing them fairly in the single-shot setting requires charging index construction. The authors’ OPF-Miner source is publicly available [33], but its released program is threshold-driven and couples its original experimental I/O/parameter protocol to that objective. We therefore do not alter the published core and label CAL-THRESH-HJ only as a generator-matched semantic baseline. The supplementary material includes a cross-code protocol that would supply the true K -th score as an oracle-calibrated minimum support, collect every qualifying OPF, and then apply the present deterministic tie order. Until that external execution is completed on the same datasets and host, this paper makes no runtime-superiority claim over official OPF-Miner.

8.4 RQ1: differential and property validation

With seed 20260909, 1,000 randomized trials generate both tie-free permutation series and tie-rich discrete series. Each trial is evaluated at $\lambda \in \{0, 0.03, 0.1, 0.3, 5.0\}$, multiple K values, and $\ell_{\min} \in \{2, 3\}$. HJ-TopK, SP-DUB, SP-Full, and SP-Gated are compared rank-by-rank against BF-Oracle, yielding 20,000 ordered top- K differential comparisons. All pass. Separate exhaustive-state property checks verify $F_\lambda(x) \leq \text{SEUB}_d(p)$ and the pair-depth/P-PDUB inequalities. Unit tests also cover non-finite observations, non-finite λ , signed zero, and the score-tie total order at $\lambda = 0$.

This evidence validates agreement for tested finite instances; Theorem 3, rather than randomized testing, is the correctness argument.

8.5 RQ2–RQ3: public-series runtime and pruning

The public configuration is $K = 50$, $\ell_{\min} = 2$, $\ell_{\max} = 12$, and $\lambda = 1/n$ for every dataset. Relative to HJ-TopK, SP-Full reduces median runtime by $5.74\times$ – $19.48\times$ across the six datasets. The largest input, household power with 2,049,280 observations, falls from 4,796.75 ms to 752.66 ms ($6.37\times$); Jena temperature falls from 1,230.85 ms to 214.45 ms ($5.74\times$). Figure 2 summarizes the speedups of all pruned variants. The structural counters in Table 5 explain the gain more directly than speedup alone. Endpoint-alignment comparisons fall by 60.5%–70.9%. In the SP-Full runs, the pair bound rejects 62–100 compatible pairs before alignment and the subtree envelope still removes 83–110 live states. Isolated SP-DUB records more subtree prunes (184–205) because pair rejection is off and more states reach later levels; the 83–110 figure is therefore not the isolated SE-DUB contribution. SP-Full is faster than SP-DUB on five datasets but slightly slower on S&P 500 (1.01 versus 0.94 ms), showing that pair-bound overhead can exceed its benefit in a small cell. We therefore present P-PDUB as a safe lineage bound and report SP-DUB, SP-Full, and SP-Gated separately rather than claiming universal pair-bound acceleration.

To quantify the *marginal* contribution of P-PDUB, Table 6 compares SP-DUB and SP-Full on the same 20 measured repetitions. P-PDUB reduces alignment work by a further 7.9–13.0% after subtree pruning has already been applied. We use a two-sided paired Wilcoxon signed-rank test on per-repetition wall times and Holm correction across the six datasets; the table also

Table 4 Public-series runtime at $K = 50$, $\ell_{\max} = 12$, and $\lambda = 1/n$. Values are median $[Q_1, Q_3]$ milliseconds over 20 measured repetitions after 10 warmups. Speedup is HJ-TopK divided by SP-Full.

Dataset	n	HJ-TopK	SP-DUB	SP-Full	SP-Gated	Speedup
S&P 500	2,514	19.75 [13.71, 22.68]	0.94 [0.82, 1.42]	1.01 [0.81, 1.53]	0.94 [0.83, 1.27]	19.48×
NASDAQCOM	14,014	72.00 [71.02, 81.11]	4.04 [3.63, 4.39]	3.92 [3.72, 4.32]	4.06 [3.52, 4.21]	18.36×
SILSO	72,967	121.51 [119.23, 125.02]	14.87 [14.52, 15.59]	14.72 [14.03, 15.52]	14.68 [14.15, 15.70]	8.26×
Beijing PM2.5	41,757	103.41 [96.51, 112.59]	9.51 [9.24, 10.21]	9.04 [8.57, 9.89]	8.90 [8.60, 9.26]	11.43×
Jena temperature	420,451	1230.85 [1191.94, 1250.21]	238.59 [233.15, 259.03]	214.45 [211.37, 224.96]	219.14 [210.23, 227.04]	5.74×
Household power	2,049,280	4796.75 [4343.38, 4990.24]	851.10 [665.34, 985.10]	752.66 [643.87, 943.93]	773.27 [669.36, 883.93]	6.37×

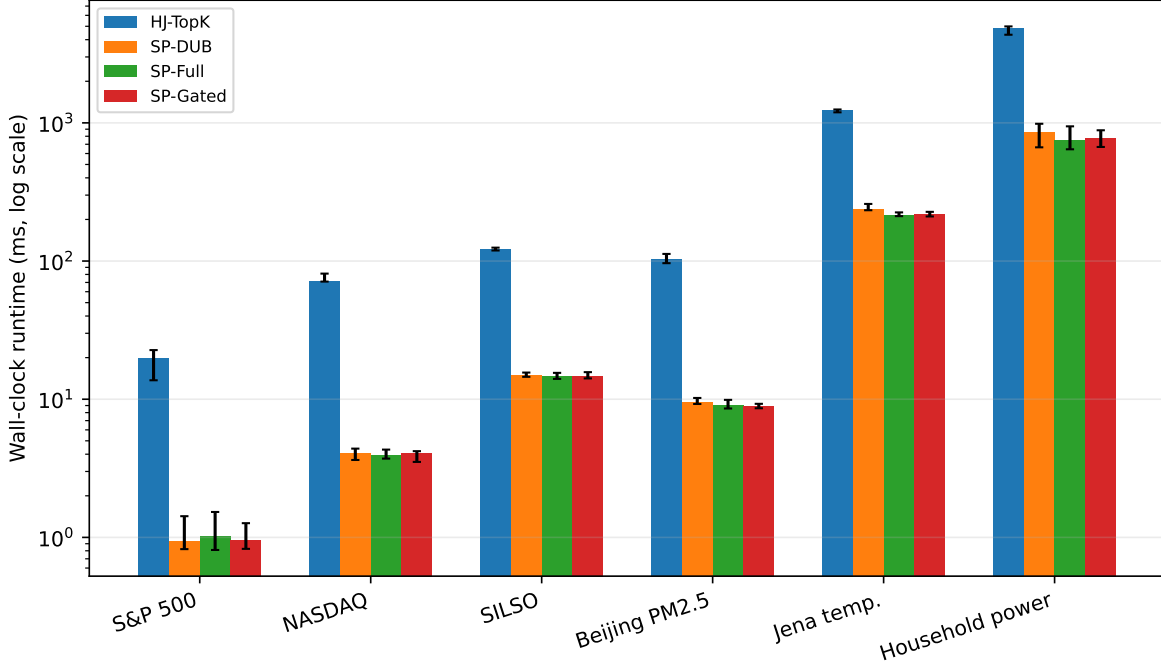


Fig. 1 Wall-clock runtime on six real datasets; error bars span Q_1 to Q_3 and the vertical scale is logarithmic. Each bar summarizes 20 measured repetitions after 10 warmups.

reports the bootstrap 95% interval for the paired median difference $t_{\text{DUB}} - t_{\text{Full}}$. Only the Jena cell remains statistically distinguishable at family-wise $\alpha = 0.05$, which supports the narrower claim that P-PDUB is structurally effective but its wall-time benefit depends on the workload and timer regime. The theoretical condition in Proposition 4 complements this ablation: pair pruning is uniquely useful when each parent is promising at some depth but the two promising-depth sets do not overlap. The result also motivates SP-Gated, which avoids paying the pair-profile scan when the upcoming alignment is already inexpensive.

The oracle-calibrated threshold baseline in Table 7 receives the true K -th score in advance, which a real thresholded user would not know. Even

under that favorable information, it must enumerate all states necessary for a complete threshold result under the present safe semantics. These three cells were measured on the auxiliary Linux host together with that host’s SP-Full times; they are not mixed with the Windows public campaign in Table 4. The table is intended as semantic context, not as a claim that SP-TK-HJ dominates the authors’ official OPF-Miner implementation.

8.6 RQ4: controlled synthetic scaling

A revision campaign on one Linux/KVM host (OpenJDK 21.0.11, five AMD EPYC 9V74 vCPUs, 5.8 GiB RAM, `-Xms1g -Xmx4g`) uses tie-free Gaussian random walks generated by Java `Random.nextGaussian`, seeds $\{101, 303, 505\}$, $\lambda =$

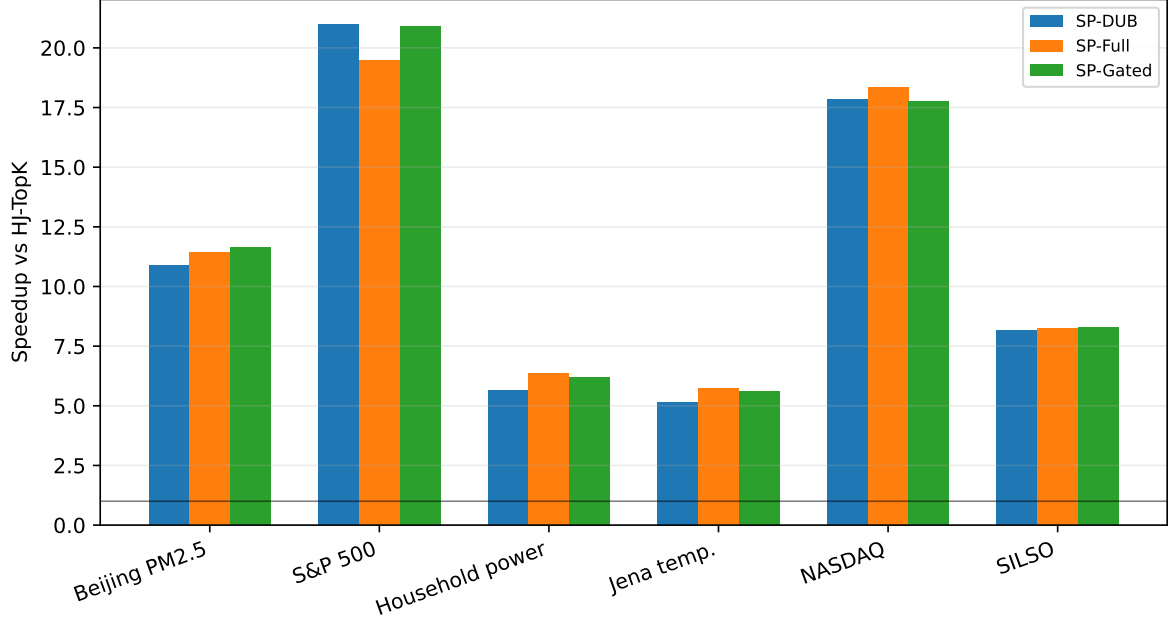


Fig. 2 Median speedup of each pruned variant relative to HJ-TopK. The horizontal line at one denotes equal runtime.

Table 5 Median structural counters on public datasets. Alignment comparisons are endpoint comparisons inside occurrence-list fusion.

Dataset	HJ align	SP-DUB align	SP-Full align	SP-Full pair prunes	SP-Full subtree prunes	Align. reduction
S&P 500	105,711	46,551	41,748	79	103	60.5%
NASDAQCOM	783,607	265,292	239,551	63	106	69.4%
SILSO	2,226,803	954,222	878,796	68	109	60.5%
Beijing PM2.5	1,681,318	690,827	633,147	62	110	62.3%
Jena temperature	27,152,079	9,091,022	7,907,219	100	83	70.9%
Household power	64,911,610	24,184,854	21,994,111	74	105	66.1%

$1/n$, $\ell_{\max} = 12$, and three warmups plus three measured repetitions per seed. Table 8 reports the median of the three per-seed medians for $K = 50$. The same host and protocol are used from $n = 1,000$ through 50,000. The monotone increase in absolute runtime with n is expected; the relative separation remains substantial over the tested range. These measurements are engineering evidence on the stated host rather than a machine-independent constant-factor guarantee.

8.7 RQ5: memory and strict ties

The OS-level RSS campaign is intentionally reported for the three redistribution-ready core snapshots available on the auxiliary Intel Xeon Linux host. For all six public datasets, the main benchmark also records JVM used-memory peaks; for SP-Full the median internal peaks are

213.3 MiB (S&P 500), 414.9 MiB (NASDAQCOM), 827.6 MiB (SILSO), 481.9 MiB (Beijing PM2.5), 1010.9 MiB (Jena temperature), and 1377.0 MiB (household power). These are runtime telemetry rather than formal space bounds and are not portable constants. The pruned variants can use less process memory because fewer live states and occurrence lists survive, although cached profiles themselves consume additional memory.

Tie statistics in Table 3 are equally important. Financial series remain mostly strict through length 12, whereas SILSO, Beijing PM2.5, and household power reject a large fraction of windows. The paper therefore does not claim weak-order support or general tie robustness; tie-aware single-shot mining is a separate problem.

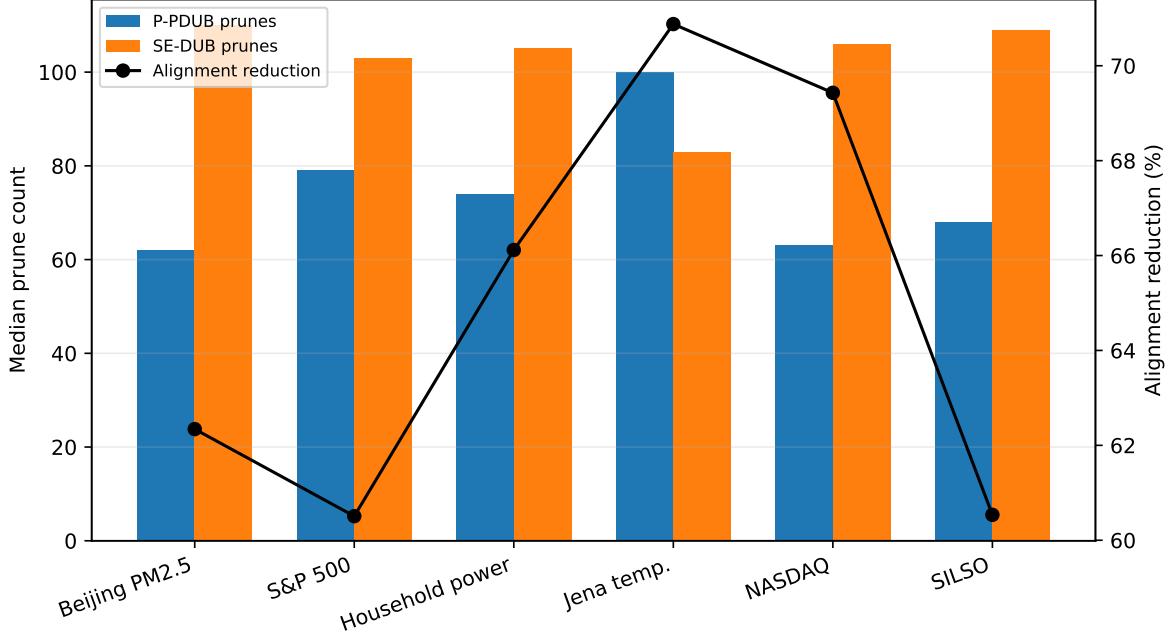


Fig. 3 Median P-DUB and SE-DUB prune counts, with the reduction in endpoint-alignment comparisons relative to HJ-TopK.

Table 6 Marginal P-DUB ablation. Positive paired time difference favors SP-Full. “Extra align. red.” is the additional reduction from SP-DUB to SP-Full. Holm- p corrects six paired Wilcoxon tests.

Dataset	SP-DUB med. (ms)	SP-Full med. (ms)	Extra align. red.	Pair-bound med. (ms)	Paired median Δ [95% CI] (ms)	Holm- p
S&P 500	0.941	1.014	10.32%	0.0175	0.029 [−0.144, 0.093]	1.000
NASDAQCOM	4.040	3.922	9.70%	0.0167	−0.029 [−0.169, 0.289]	1.000
SILSO	14.874	14.717	7.90%	0.0174	0.720 [0.417, 0.904]	0.133
Beijing PM2.5	9.514	9.044	8.35%	0.0185	0.321 [−0.215, 1.166]	0.568
Jena temperature	238.585	214.454	13.02%	0.0626	22.385 [16.840, 36.478]	1.14×10^{-5}
Household power	851.099	752.662	9.06%	0.0661	135.760 [−3.472, 211.948]	0.389

Table 7 Oracle-calibrated threshold baseline on the auxiliary Linux host (OpenJDK 21.0.11, Intel Xeon E5-2673 v4). Medians over 20 measured repetitions after 10 warmups. The exact K -th support is supplied in advance; this is not a reproduction of OPF-Miner, and SP-Full times are from the same Linux campaign rather than from Table 4.

Dataset	CAL-THRESH (ms)	SP-Full (ms)
S&P 500	185.13	6.36
NASDAQCOM	109.36	6.29
SILSO	204.56	26.15

8.8 RQ6: parameter sensitivity

We performed a separate sensitivity campaign on the revision Linux host (OpenJDK 21.0.11, AMD EPYC 9V74) using the three bundled real snapshots. For decay, we vary the dimensionless multiplier $c = \lambda n$ over $\{0, 0.25, 0.5, 1, 2, 4\}$ while

Table 8 Controlled synthetic scaling on the revision Linux host at $K = 50$. Runtime is the median of three per-seed medians.

n	HJ-TopK (ms)	SP-Full (ms)	HJ/SP-Full
1,000	7.50	0.57	13.26×
5,000	14.59	0.96	15.16×
10,000	30.96	1.85	16.73×
25,000	96.42	5.24	18.40×
50,000	226.70	8.92	25.42×

keeping $K = 50$ and $\ell_{\max} = 12$. For maximum length we use $\ell_{\max} \in \{6, 8, 10, 12, 14\}$ at $c = 1$. We additionally vary $K \in \{10, 50, 200\}$ on NASDAQCOM and SILSO with 10 warmups and 20 measured repetitions; the smallest S&P 500 K cell showed a process-level timing regime shift on this shared host and is conservatively omitted rather

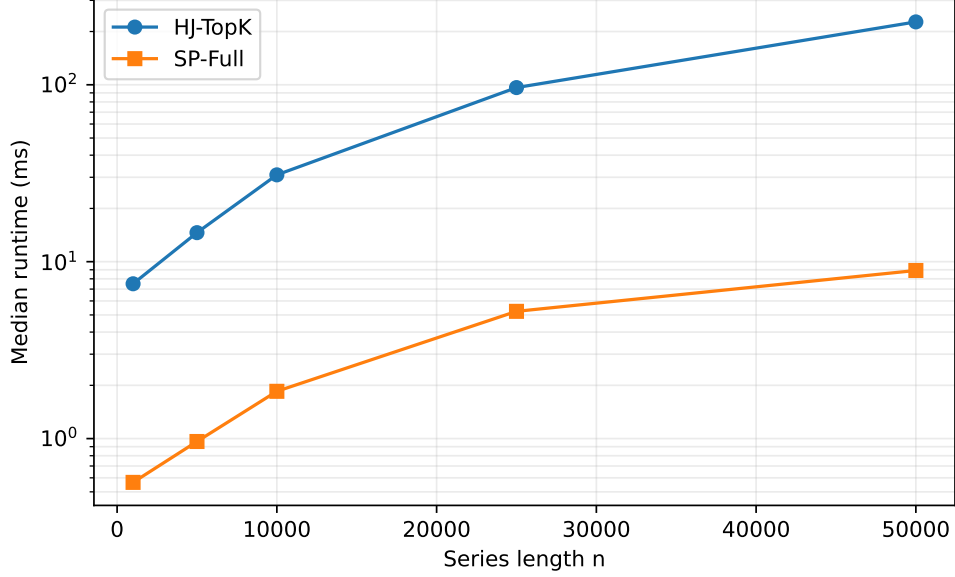


Fig. 4 Synthetic scaling at $K = 50$ on a single revision host. The vertical axis is logarithmic; each point is the median of three per-seed medians.

Table 9 One-shot maximum resident set size (MiB) on the separate Linux campaign; one JVM process per cell.

Dataset	HJ-TopK	SP-DUB	SP-Full
S&P 500	83.2	51.1	48.7
NASDAQCOM	114.9	71.4	73.6
SILSO	153.9	100.4	99.7

than cherry-picked. Table 10 reports HJ-TopK/SP-Full median speedup, so absolute times are not mixed with the Windows public campaign. Across $c = 0$ to 4 the relative separation is stable on all three series, indicating that the pruning mechanism is not tied to the default $\lambda = 1/n$. The advantage grows with $\ell_{\max}$ because the unpruned search explores increasingly expensive deeper levels, while stronger heap thresholds and shift envelopes remove more work. Increasing K weakens pruning because the heap threshold is lower; the observed speedup therefore decreases as K grows on NASDAQCOM and SILSO. These trends are consistent with the algorithmic design rather than with a single tuned parameter setting.

8.9 RQ7: practical parameter guidance and a representative result

The parameters have direct operational interpretations. First, the forgetting rate can be selected through a desired endpoint half-life h : because $\exp(-\lambda h) = 1/2$, choose $\lambda = \ln 2/h$. Reporting $c = \lambda n$ makes decay strength comparable across series lengths. Second, K should reflect the analyst’s review budget; increasing K lowers the final K -th score and therefore weakens pruning. Third, $\ell_{\max}$ controls the longest trend shape considered and has the strongest effect on search-space growth. A practical workflow is to start with a domain-meaningful moderate length (for example 8–12 observations for exploratory daily/regularly sampled data), inspect top- K stability and strict-window coverage, and increase $\ell_{\max}$ only while longer patterns materially change the result. The sensitivity study confirms that runtime pressure rises with $\ell_{\max}$ while the proposed pruning becomes increasingly valuable.

As a concrete descriptive example, running SP-Full on the bundled S&P 500 snapshot with $\lambda = 1/n$, $K = 5$, $\ell_{\min} = 2$, and $\ell_{\max} = 12$ returns the rank patterns shown in Table 11. These patterns are *descriptive order shapes*, not forecasts:

Table 10 Auxiliary real-data sensitivity on the revision Linux host. Entries are median HJ-TopK/SP-Full speedups.

Dataset	$c = \lambda n$													
	0	.25	.5	1	2	4	6	8	$\ell_{\max}$ 10	12	14	10	K 50	200
S&P 500	12.31	12.12	12.31	12.43	13.21	12.75	1.30	4.69	7.90	12.29	15.68	—	—	—
NASDAQCOM	17.22	16.96	17.05	16.60	17.24	17.22	1.26	3.62	9.41	17.30	23.46	27.63	16.44	10.95
SILSO	8.30	8.06	8.24	7.97	7.93	7.74	1.15	2.51	5.14	8.08	9.64	11.68	7.71	5.53

Table 11 Representative top-5 patterns on the bundled S&P 500 snapshot ($n = 2514$, $\lambda = 1/n$). Occurrence counts are unweighted; ranking uses forgetting support.

Rank	Pattern	F_λ	Occurrences
1	(1, 2)	869.390	1375
2	(2, 1)	719.328	1137
3	(1, 2, 3)	465.606	733
4	(3, 2, 1)	314.830	494
5	(1, 2, 3, 4)	248.849	390

(1, 2) denotes an adjacent increase, (2, 1) an adjacent decrease, and (1, 2, 3) / (3, 2, 1) monotone three-point moves. Exponential weighting ranks repeated shapes with more recent occurrences more strongly. The last-occurrence dates demonstrate that the top patterns remain active near the end of the snapshot, but no trading or predictive claim follows from that observation. The five most recent endpoints of (1, 2, 3) in this processed snapshot correspond to 30–31 July, 3 and 12 August, and 2 September 2026; the mapping from processed indices to dates is derived from the bundled FRED snapshot after removing missing markers. This example is included to clarify interpretation, not to establish domain utility.

9 Discussion, scope, and threats to validity

The central contribution is algorithmic: safe pruning and exact ranking for a single raw series under endpoint-based exponential forgetting. This section consolidates the boundaries of that claim instead of repeating disclaimers throughout the paper.

9.1 Numerical scope and portability

The real-valued theorems are independent of floating-point arithmetic; Corollary 1 states the implementation-level condition used by the Java artifact. `StrictMath.exp` is specified to produce reproducible results across Java platforms subject to the Java/IEEE-754 contract, all stored

weights are finite nonnegative values not exceeding one, and sums are accumulated in the same end-point order. This supports reproducible binary64 execution, not arbitrary-precision certification for adversarial near-ties. Interval or multiprecision certification is a natural extension when score separations approach rounding scale.

9.2 Strict versus weak-order semantics

Strict windows are a modeling choice, not a claim that ties are rare. The strict fraction at length 12 ranges from 17.70% to 99.28% across the six datasets (Table 3). An exploratory Spearman association between strict fraction and HJ-TopK/SP-Full speedup over these six heterogeneous datasets is positive ($\rho = 0.657$) but not statistically conclusive ($p = 0.156$); series length, value cardinality, and search structure are substantial confounders. Extending to weak orders requires equality-block canonicalization, tie-preserving prefix/suffix compatibility, extra fusion cases, and a renewed completeness proof. The endpoint-shift subset idea behind SEUB_d is likely reusable once occurrences are defined consistently, but the current theorems do not cover that model.

9.3 Baselines and external validity

HJ-TopK isolates the proposed bounds because it uses the identical generator, score, data structures, and total order. BF-Oracle supplies an implementation-independent correctness reference on small instances, and CAL-THRESH-HJ supplies favorable threshold semantics. The released official OPF-Miner code is cited and a cross-code protocol is supplied, but a same-host official execution has not been completed in this artifact; accordingly, the paper does not claim runtime superiority over OPF-Miner. COPP-Miner optimizes class contrast, and OPST amortizes index construction over repeated queries, so raw wall-clock comparisons without objective/cost normalization would be misleading. A future break-even study should measure

total cost $C_{\text{index}} + QC_{\text{query}}$ against QC_{single} as the number of queries Q increases.

9.4 Benchmark and statistical validity

The public campaign uses randomized method order, 10 warmups, 20 measured repetitions, Q_1/Q_3 intervals, and bootstrap intervals; the P-PDUB ablation additionally uses paired Wilcoxon tests with Holm correction. JVM JIT, GC, scheduling, and thermal effects remain possible sources of variance. Absolute timings are therefore interpreted only within a host/campaign, while structural counters (pruned lineages, endpoint comparisons) provide a machine-independent explanation of the main effect. The separate sensitivity and synthetic campaigns are reported as within-host evidence rather than combined with the Windows timings.

9.5 Responsible interpretation

The financial and environmental series are public, non-personal data. Mined OP shapes summarize relative-order recurrence and recency; they do not establish causation, expected return, health risk, or predictive accuracy. Financial patterns should not be interpreted as investment advice, and PM2.5 or climate patterns should not replace calibrated environmental or public-health models. Potential social benefit lies in transparent exploratory summarization and hypothesis generation, followed by domain-specific validation before any consequential use.

9.6 Remaining research directions

The most important extensions are weak-order semantics, certified near-tie comparison, stronger adaptive scheduling of pair bounds, a same-host official OPF-Miner conversion study, indexed/single-shot break-even analysis, parallel execution, and downstream task evaluation. Proposition 4 also suggests a direct adaptive heuristic: estimate overlap of the two promising-depth sets before paying the full pair-bound scan.

10 Conclusion

We formulated exact global top- K strict order-preserving pattern mining under endpoint-based

exponential forgetting and presented SP-TK-HJ, a single-shot compatibility-fusion search. Right extension can increase the weights of surviving occurrences, invalidating ordinary support anti-monotonicity. The proposed depth profile instead follows each feasible endpoint shift directly, its envelope supports safe subtree pruning, and a depth-aligned combination of two parent profiles supports pre-alignment pair pruning. We proved exact global top- K completeness and gave explicit characterizations of depth-bound slack and of the conditions under which pair pruning is strictly additional to parent-level pruning. The data-sensitive complexity analysis is complemented by a coarse worst-case bound in n , K , and ℓ_{max} , making clear that pruning improves practical search rather than changing the combinatorial worst case.

The Java artifact passes 20,000 ordered differential comparisons against an independent raw-window oracle. On six public series with up to 2,049,280 observations, SP-Full reduces endpoint-alignment comparisons by 60.5–70.9% and is 5.74–19.48 $\times$ faster in median time than the same fusion engine with the proposed bounds disabled. Beyond the subtree bound, P-PDUB removes an additional 7.9–13.0% of alignment work; paired timing tests show that this structural gain translates into statistically clear wall-time improvement only on some workloads, so the paper makes no universal pair-bound speed claim. Sensitivity experiments across forgetting strength, maximum length, and K , together with multi-seed scaling to $n = 50,000$, support the robustness of the main pruning effect. The current result is an exact strict-window, single-shot mining method. Future work should extend the semantics to weak orders, certify adversarial near-ties, complete a same-host comparison with official OPF-Miner, quantify the indexed/single-shot break-even point, and evaluate whether the discovered recency-weighted order shapes improve specific downstream tasks.

Supplementary information. The Java 21 implementation, public dataset snapshots, and archived measurement files are provided as Supplementary Software S1 and in the public artifact [32].

Acknowledgements. During preparation of this work, the authors used OpenAI ChatGPT, OpenAI Codex, and xAI Grok for manuscript restructuring, language editing, code assistance,

consistency checking, number audit against archived CSVs, and scripts used to reproduce tables and visualizations. The AI systems were not treated as authors, did not select or fabricate experimental observations, and did not determine the reported numerical results. All mathematical arguments, citations, source code, raw outputs, tables/figures, and experimental interpretations were independently inspected and approved by the authors, who take full responsibility for the final publication.

Declarations

Funding

The authors received no specific funding for this work.

Competing interests

The authors declare no competing interests.

Ethics approval and consent to participate

Not applicable. The study uses public, non-personal time-series datasets and synthetic data.

Consent for publication

Not applicable.

Data availability

The public datasets are cited in the bibliography [25–31]. Processed snapshots, checksums, and the archived measurement files underlying the tables and figures are in the public artifact [32] at `results/published/`. Absolute wall-clock times remain host-specific; those archived CSVs, not a later rerun on a different machine, are the sources of the reported numbers. Official dataset providers retain their original licenses.

Materials availability

Not applicable.

Code availability

The Java 21 source for SP-TK-HJ and all compared variants, the independent BF-Oracle, and

the build/validation/campaign scripts are provided as Supplementary Software S1 and in the public artifact [32]. The official OPF-Miner repository is cited separately [33] and is not part of our artifact.

Author contributions

K.-C.N.: conceptualization, methodology, software, investigation, writing—original draft. N.-M.B.: validation, writing—review and editing. C.-P.T.: validation, writing—review and editing. All authors reviewed and approved the submitted version.

Appendix A Proof details

This appendix records proof details that are intentionally shortened in the main text to improve readability.

A.1 Child-realization completeness

For Lemma 3, necessity follows because any observed child occurrence ending at j has the length- m prefix occurrence at $j - 1$ and the length- m suffix occurrence at j ; the two-pointer scan must therefore encounter that adjacent endpoint pair. Conversely, each adjacent pair identifies one raw length- $(m + 1)$ window with the required overlapping normalized parents. When FUSETEMPLATES yields one template, compatibility determines every rank relation. When it yields two, only the relation between the non-overlapping first and last raw values is undetermined; their direct comparison selects exactly one strict child and equality is rejected. Since each adjacent endpoint pair is consumed once, every returned occurrence list is complete, sorted, and duplicate-free.

A.2 Binary64 bound preservation

For Corollary 1, each stored weight returned by `StrictMath.exp` is finite and lies in $[0, 1]$; underflow to $+0.0$ is allowed. The array-backed implementation has $n \leq \text{Integer.MAX_VALUE}$, so a support or bound sum cannot approach binary64 overflow. Child and bound folds traverse endpoint positions in increasing order. For finite nonnegative binary64 values under round-to-nearest, addition is monotone in each operand. Scanning the union of child endpoints and shifted parent endpoints, the bound either adds the same term as the child or adds an additional nonnegative term. Induction

over the common endpoint order therefore preserves the implemented child-sum $\leq$ implemented bound-sum relation.

A.3 Global top- K completeness

For Theorem 3, soundness follows from observed length-2 seeds and Lemma 3; only requested lengths are offered to the heap. For generation without pruning, take any observed strict length- $(m+1)$ pattern. Its normalized length- m prefix and suffix are observed, compatible parents. Lemma 2 emits their pair and Lemma 3 reconstructs the child, so induction from length 2 generates every observed pattern through $\ell_{\max}$.

Now let x be in the final top- K result. Before the heap is full, no score-based pruning is performed. After it is full, let θ_τ be the current primary threshold at any earlier time τ . Lemma 4 gives $\theta_\tau \leq \theta_{\text{final}}$, and membership of x in the final top- K gives $F_\lambda(x) \geq \theta_{\text{final}}$. Along every fusion step on the lineage of x , Lemma 5 places both required parent envelopes at or above $F_\lambda(x)$, so neither parent satisfies the strict prune test. Theorem 2 likewise places the relevant P-PDUB at or above $F_\lambda(x)$, so the pair cannot be discarded. Hence every step on a final-answer lineage survives. Because equality with the threshold is retained and the heap uses the complete score/length/lexicographic total order, the final capacity- K heap contains exactly the best $\min(K, |\mathcal{P}|)$ observed candidates.

Appendix B Reproducibility checklist

The public repository [32] includes machine-readable measurements underlying the tables (results/published/): the 20-repetition public campaign, bootstrap intervals, P-PDUB marginal paired statistics, parameter-sensitivity CSVs, strict-window statistics, memory telemetry, calibrated-threshold timings, and the representative S&P 500 top-pattern trace. The same repository includes the Java source, build scripts, test suite, independent brute-force oracle, and dataset preparation documentation. These materials are intended to make each quantitative statement in the paper traceable to a stored output rather than to manually transcribed numbers.

References

- [1] Kim, J., Eades, P., Fleischer, R., Hong, S.-H., Iliopoulos, C.S., Park, K., Puglisi, S.J., Tokuyama, T.: Order-preserving matching. *Theoretical Computer Science* **525**, 68–79 (2014) <https://doi.org/10.1016/j.tcs.2013.10.006>
- [2] Crochemore, M., Iliopoulos, C.S., Kociumaka, T., Kubica, M., Langiu, A., Pissis, S.P., Radoszewski, J., Rytter, W., Walen, T.: Order-preserving incomplete suffix trees and order-preserving indexes. In: *SPIRE. LNCS*, vol. 8214, pp. 84–95 (2013). https://doi.org/10.1007/978-3-319-02432-5_13
- [3] Wu, Y., Hu, Q., Li, Y., Guo, L., Zhu, X., Wu, X.: Opp-miner: Order-preserving sequential pattern mining for time series. *IEEE Transactions on Cybernetics* **53**(5), 3288–3300 (2023) <https://doi.org/10.1109/TCYB.2022.3169327>
- [4] Wu, Y., Zhao, X., Li, Y., Guo, L., Zhu, X., Fournier-Viger, P., Wu, X.: Opr-miner: Order-preserving rule mining for time series. *IEEE Transactions on Knowledge and Data Engineering* **35**(11), 11722–11735 (2023) <https://doi.org/10.1109/TKDE.2022.3224963>
- [5] Li, Y., Liu, J., Guo, Y., Liu, J., Wu, Y.: Aop-miner: Approximate order-preserving pattern mining for time series. In: *2022 IEEE International Conference on Knowledge Graph (ICKG)*, pp. 149–156 (2022). <https://doi.org/10.1109/ICKG55886.2022.00026>
- [6] Wu, Y., Wang, Z., Li, Y., Guo, Y., Jiang, H., Zhu, X., Wu, X.: Co-occurrence order-preserving pattern mining with keypoint alignment for time series. *ACM Transactions on Management Information Systems* **15**(2), 1–27 (2024) <https://doi.org/10.1145/3658450>
- [7] Wu, Y., Lou, S., Li, Y., Guo, L., Fournier-Viger, P., Wu, X.: Outo-miner: Detecting outlying occurrences in maximal frequent order-preserving patterns in time series. *Information Sciences* **720**, 122497 (2025) <https://doi.org/10.1016/j.ins.2025.122497>

- [8] Zhang, Y., Wu, Y., Li, Y.: Sopp-miner: Stable distribution order-preserving pattern mining for time series. *IEEE Transactions on Knowledge and Data Engineering* **38**(8), 5406–5419 (2026) <https://doi.org/10.1109/TKDE.2026.3692778>
- [9] Zhou, Y., Wang, F., Tang, Y.: Order-preserving pattern matching: Review. *IEEE Transactions on Knowledge and Data Engineering* **38**(3), 1885–1904 (2026) <https://doi.org/10.1109/TKDE.2025.3649808>
- [10] Li, Y., Ma, C., Gao, R., Wu, Y., Li, J., Wang, W., Wu, X.: Opf-miner: Order-preserving pattern mining with forgetting mechanism for time series. *IEEE Transactions on Knowledge and Data Engineering* **36**(12), 8981–8995 (2024) <https://doi.org/10.1109/TKDE.2024.3438274>
- [11] Li, L., Zuba, W., Loukides, G., Pissis, S.P., Matsangidou, M.: Scalable order-preserving pattern mining. In: 2024 IEEE International Conference on Data Mining (ICDM), pp. 211–220 (2024). <https://doi.org/10.1109/ICDM59182.2024.00028>
- [12] Li, L., Zuba, W., Loukides, G., Pissis, S.P., Matsangidou, M.: Order-preserving pattern mining and comparison via indexing. *IEEE Transactions on Knowledge and Data Engineering*, 1–18 (2026) <https://doi.org/10.1109/TKDE.2026.3710634> . Published online 6 July 2026
- [13] Wu, Y., Meng, Y., Li, Y., Guo, L., Zhu, X., Fournier-Viger, P., Wu, X.: Copp-miner: Top-k contrast order-preserving pattern mining for time series classification. *IEEE Transactions on Knowledge and Data Engineering* **36**(6), 2372–2387 (2024) <https://doi.org/10.1109/TKDE.2023.3321749>
- [14] Wang, J., Han, J., Lu, Y., Tzvetkov, P.: Tfp: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering* **17**(5), 652–664 (2005) <https://doi.org/10.1109/TKDE.2005.81>
- [15] Amagata, D., Hara, T.: Mining top-k co-occurrence patterns across multiple streams. *IEEE Transactions on Knowledge and Data Engineering* **29**(10), 2249–2262 (2017) <https://doi.org/10.1109/TKDE.2017.2728537>
- [16] Wu, Y., Wang, Y., Li, Y., Zhu, X., Wu, X.: Top-k self-adaptive contrast sequential pattern mining. *IEEE Transactions on Cybernetics* **52**(11), 11819–11833 (2022) <https://doi.org/10.1109/TCYB.2021.3082114>
- [17] Cho, S., Na, J.C., Park, K., Sim, J.S.: A fast algorithm for order-preserving pattern matching. *Information Processing Letters* **115**(2), 397–402 (2015) <https://doi.org/10.1016/j.ipl.2014.10.018>
- [18] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 2–11 (2003). <https://doi.org/10.1145/882082.882086>
- [19] Wu, Y., Tian, S., Li, Y., Liu, J., Li, J., Wang, W., Wu, X.: Oip-miner: One-off incremental sequential pattern mining with forgetting factor. *Science China Information Sciences* **68**, 209101 (2025) <https://doi.org/10.1007/s11432-024-4496-9>
- [20] Cafaro, M., Pulimeno, M., Epicoco, I., Aloisio, G.: Mining frequent items in the time fading model. *Information Sciences* **370–371**, 221–238 (2016) <https://doi.org/10.1016/j.ins.2016.07.077>
- [21] Leung, C.K.-S., Jiang, F.: Frequent pattern mining from time-fading streams of uncertain data. In: Data Warehousing and Knowledge Discovery (DaWaK 2011). Lecture Notes in Computer Science, vol. 6862, pp. 252–264 (2011). https://doi.org/10.1007/978-3-642-23544-3_19
- [22] Alkan, O.K., Karagoz, P.: CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction. *IEEE Transactions on Knowledge and Data Engineering* **27**(10), 2645–2657 (2015) <https://doi.org/10.1109/TKDE.2015.2438537>

1109/TKDE.2015.2420557

- [23] Zhang, C., Yang, Y., Du, Z., Gan, W., Yu, P.S.: HUSP-SP: Faster utility mining on sequence data. *ACM Transactions on Knowledge Discovery from Data* **18**(1), 1–21 (2023) <https://doi.org/10.1145/3597935>
- [24] Buffett, S.: Dramatically reducing search for high utility sequential patterns by maintaining candidate lists. *Information* **11**(1), 44 (2020) <https://doi.org/10.3390/info11010044>
- [25] Federal Reserve Bank of St. Louis: S&P 500 (SP500). FRED. Accessed 18 August 2026 (2026). <https://fred.stlouisfed.org/series/SP500>
- [26] Federal Reserve Bank of St. Louis: NASDAQ Composite (NASDAQCOM). FRED. Accessed 18 August 2026 (2026). <https://fred.stlouisfed.org/series/NASDAQCOM>
- [27] WDC-SILSO, Royal Observatory of Belgium: Daily Total Sunspot Number, Version 2.0. Accessed 18 August 2026; CC BY-NC 4.0 (2026). <https://doi.org/10.24414/qnza-ac80> . https://www.sidc.be/SILSO/DATA/SN_d_tot_V2.0.csv
- [28] Chen, S.: Beijing PM2.5. UCI Machine Learning Repository. Hourly observations from 2010 to 2014; CC BY 4.0 (2015). <https://doi.org/10.24432/C5JS49> . <https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data>
- [29] Attri, P., Sharma, Y., Takach, K., Shah, F.: Timeseries Forecasting for Weather Prediction: Jena Climate Dataset. Keras. Updated 22 November 2023; accessed 10 September 2026 (2020). https://keras.io/examples/timeseries/timeseries_weather_forecasting/
- [30] Max Planck Institute for Biogeochemistry: Weather Data from the Beutenberg Campus Station, Jena. Data released under CC BY 4.0; accessed 10 September 2026 (2026). https://www.bgc-jena.mpg.de/wetter/weather_data.html
- [31] Hebrail, G., Berard, A.: Individual Household Electric Power Consumption. UCI Machine Learning Repository. One-minute measurements from 2006 to 2010; CC BY 4.0 (2006). <https://doi.org/10.24432/C58K54> . <https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption>
- [32] Nguyen, K.-C., Bui, N.-M., Tran, C.-P.: SP-TK-HJ: Exact Top-K Order-Preserving Pattern Mining under Exponential Forgetting. GitHub repository. Accessed 14 September 2026 (2026). <https://github.com/Porcupine1805/sp-tk-hj>
- [33] OPF-Miner authors: OPF-Miner Algorithms and DataSets. GitHub repository. Accessed 10 September 2026 (2024). <https://github.com/wuc567/Pattern-Mining/tree/master/OPF-Miner>