Supplement for "Order-Preserving Pattern Mining and Comparison via Indexing"

Ling Li*[1], Wiktor Zuba*[2,3], Grigorios Loukides[1], Solon P. Pissis[2,4], and Maria Matsangidou[5]

[1]King's College London, London, UK
[2]CWI, Amsterdam, The Netherlands
[3]University of Warsaw, Warsaw, Poland
[4]Vrije Universiteit, Amsterdam, The Netherlands
[5]CYENS—Centre of Excellence, Nicosia, Cyprus
[1]{ling.2.li, grigorios.loukides}@kcl.ac.uk, [2] solon.pissis@cwi.nl, [3] w.zuba@mimuw.edu.pl, [5] m.matsangidou@cyens.org.cy

I. Omitted Proofs

A. Proof of ??

Proof. We first show that, for any string w over a totally ordered alphabet of size σ, there exist only up to $2\sigma+1$ possibilities for $\mathsf{LastCode}(w\cdot a)$ over all letters a.

Let us start with the trivial case, where w is the empty string and so $\mathsf{LastCode}(w\cdot a)=\mathsf{LastCode}(a)=(\bot,\bot)$, for any a. In this case, the above upper bound clearly holds.

For the rest of the proof, we assume that w is nonempty. We have the following two cases:

1) If letter a belongs to the alphabet of w, then $\mathsf{LastCode}(w\cdot a)$ is equal to (x_a,x_a), where x_a is the *single position* of the rightmost occurrence of a in w. These are σ possibilities: one for every a.
2) Otherwise let the letters of w in the sorted order be equal to a_j for $j\in[0,\sigma)$, and let x_j be the position of the rightmost occurrence of a_j in w. There exists a single value $j\in[-1,\sigma)$ such that $a_j<a<a_{j+1}$, where $a_{-1}=-\infty$ and $a_\sigma=\infty$. Then $\mathsf{LastCode}(w\cdot a)=(x_j,x_{j+1})$, where $x_{-1}=x_\sigma=\bot$. Note that, in particular, only elements x_j can occur in $\mathsf{LastCode}$. These are $\sigma+1$ possibilities: $(\bot,x_0),(x_0,x_1),(x_1,x_2),\ldots,(x_{\sigma-1},\bot)$.

The upper bound of $2\sigma+1$ follows by combining cases 1 and 2 above. By the above upper bound, any branching node of OPST cannot have more than $2\sigma+1$ outgoing edges. □

B. Proof of ??

Proof. The upper bound is obtained via a more careful analysis of the proof of **??** – if the witness occurrence representing a branching node of $\mathsf{OPST}(w)$ contains all the σ letters from w, then there are only $\sigma\leq 2\sigma-1$ possibilities of extension of such a node as the next letter must be equal to one of the previous ones of the substring. Otherwise, the representative is over a smaller alphabet $\sigma'\leq\sigma-1$, and hence by **??**, the maximal possible outdegree of this node is equal to $2\sigma'+1\leq 2\sigma-1$.

For the lower bound, we construct a string $x=0\ 1\ 2\ \ldots(\sigma-1)$ with $\mathsf{PrefCode}(x)=(\bot,\bot)(0,\bot)(1,\bot)\ldots(\sigma-2,\bot)$. For a string x' obtained by replacing the last position of x with a different letter, we have $\mathsf{PrefCode}(x')=(\bot,\bot)(0,\bot)\ldots(\sigma-3,\bot)(i,i)$, where i is the letter replacing the last letter of x. For a string x'' obtained by shifting a single letter of x to its end, we have $\mathsf{PrefCode}(x'')=(\bot,\bot)(0,\bot)\ldots(\sigma-3,\bot)(i-1,i)$, where i is the letter shifted to the end (and $-1=\bot$). All such strings share the first $\sigma-1$ positions of the $\mathsf{PrefCode}$ as the first $\sigma-1$ positions form a strictly increasing sequence. The $\mathsf{SufCode}$ of each of those $2\sigma-1$ sequences (x, $\sigma-1$ possibilities for x', $\sigma-1$ possibilities for x'') is distinct however. Hence, for a string w of length $\sigma(2\sigma-1)$, being the concatenation of all those strings, the node of $\mathsf{OPST}(w)$ reached by reading $(\bot,\bot)(0,\bot)\ldots(\sigma-3,\bot)$ has exactly $2\sigma-1$ children. Hence, $2\sigma-1$ is both an upper and a lower bound. □

C. Proof of ??

Proof. We first prove that an explicit non-branching node in $\mathsf{OPST}(w)$ can appear on a root-to-leaf path with witness suffix w' only right before the first occurrence of some letter.

Let xb be a witness substring of a node directly below an explicit non-branching node u, where x is a string and b is a letter. For the explicit non-branching node to be constructed, there must exist a branching node v such that $\mathsf{SufLink}(v)=u$. Let the witnesses of any two children of v be equal to $a'x'b'$ and $a''x''b''$, respectively, where x',x'' are strings and a',a'',b',b'' are letters. We have:

- $a'x'\approx a''x''$, since both those witnesses are represented by the same node v.
- $a'x'b'\not\approx a''x''b''$, since the two children of v are distinct nodes – recall that node v is branching.
- $x'b'\approx xb\approx x''b''$, since Locus of both $x'b'$ and $x''b''$ (witnesses of children of v with their first letter deleted) must be equal to the single child of u – recall that node u is explicit but non-branching so it has only one child.

Notice that for two equal-length strings $w_1\not\approx w_2$, there must exist two positions i,j such that the relationship ($<$, $=$, $>$) between $w_1[i]$ and $w_1[j]$ is different than the relationship between $w_2[i]$ and $w_2[j]$. Hence, the relationship between a' and b' has to be different than the relationship between a''

* denotes equal contribution.

and b'' as the relationship of all the other pairs of the letters of $a'x'b'$ and $a''x''b''$ is fixed by the two listed equivalences. This proves that b' cannot appear in x' as if that was the case the relationship between a' and b' (as a letter of x') would be fixed and the same as the relationship between a'' and b'' – a contradiction. Thus b does not appear in x since $xb \approx x'b'$.

Now, since the explicit non-branching node on a root-to-leaf path can only appear just above the first occurrence of a letter, there are only at most σ such nodes. Clearly, this also implies that there can only be at most σ explicit non-branching nodes between any two branching nodes. □

D. Proof of ??

Proof. For the upper bound, it is enough to slightly refine the proof of **??**. For the two string witnesses $a'x'b'$ and $a''x''b''$, we have that $a', b' \notin x'$ and $a'', b'' \notin x''$, but at the same time we cannot have both $a' = b'$ and $a'' = b''$. Thus x' (and thus also x'') can contain only $\sigma - 2$ unique letters; i.e., the node right above the first occurrence of the last letter cannot in fact be explicit but non-branching. At the same time, the node right above the first occurrence of the first letter of a suffix is clearly the root, which is branching (it has two children labelled with $(\perp, \perp)$ and $\$$, respectively), which refines the upper bound to $\sigma - 2$ potential explicit but non-branching nodes.

For the lower bound, let $u = 0\ 0\ 1\ 2\ \cdots\ (\sigma - 1)$, and let $w = (\sigma-1)\ u\ (\sigma-2)\ u\ \cdots\ 1\ u$. In $\mathsf{OPST}(w)$, the subtree of the node $(\perp, \perp)(0, 0)$ is composed of the strings starting with two equal letters; i.e., it is generated by the suffixes starting from u. The subtree of the node $(\perp, \perp)(0, \perp)(1, 1)$ is in turn generated by the suffixes starting from $x\ 0\ 0$ for some letter x; this tree is formed of all the nodes for which the suffix link points to a node inside the former subtree.

Now, since all the suffixes of the first subtree start from the same $\sigma+1$ letters, there is no branching node in this subtree at string depth between 2 (there is no other suffix starting from two equal letters) and σ – the top of the subtree forms a single long edge. At the same time, for suffixes $s_1 = x\ u\ \cdots$ and $s_2 = (x+1)\ u\ \cdots$, we have $s_1[0\,..\,x+1] \approx s_2[0\,..\,x+1]$ and $s_1[0\,..\,x+2] \not\approx s_2[0\,..\,x+2]$ (the first mismatch appears when $s_1[0] = x = s_1[x+2]$, but $s_2[0] = x+1 > x = s_2[x+2]$). This inequality produces a branching node at string depth $x+2$ with a suffix link to a node in the mentioned branch at depth $x+1$. As this works for every $x \in \{1, \ldots, (\sigma-2)\}$, the edge contains $\sigma - 2$ explicit nodes that are not branching. Hence, $\sigma - 2$ is both an upper and a lower bound. □

E. Proof of Lemma ??

Proof. We mimic the proof of the complexity of McCreight's algorithm [1]. Therein, the bound on the number $k(w)$ of moves was proved based on the *node depth* of the nodes u and v: when going from u to its branching parent u', the node depth decreases by 1, and then upon using the suffix link of u' it can again decrease by at most 1. Thus, we can do at most $2n$ moves upward, and since the node depth is bounded by n (the maximal node depth in the tree), at most $3n$ moves downward.

In the OP setting, by **??** and the above analysis of McCreight's algorithm, the node depth can decrease by at most $\sigma + 2$ when going upwards (which also bounds the possible number of moves) due to the explicit non-branching nodes that do not correspond to any explicit or branching node on the path from root to $v = \mathsf{SufLink}(u)$. This means that we can make at most $n(\sigma+2)$ moves upward and at most $n(\sigma+2)+n$ moves downward; that is, $k(w) \leq n(2\sigma + 5) = \mathcal{O}(n\sigma)$. □

II. Experimental Evaluation: Details and Additional Results

A. Baseline Algorithms

The baseline algorithms BA-CP and BA-MP take $\mathcal{O}(nk \log k)$ time, where k is the length of the longest frequent OP pattern in the output. They start with computing the PrefCode of all the substrings of the input string w in the order of increasing lengths. Given $\mathsf{PrefCode}(w[i\,..\,j-1])$ and the ordered balanced binary search tree on $w[i], \cdots, w[j-1]$ (i.e., the nodes of the tree are these letters) we can compute the $\mathsf{PrefCode}(w[i\,..\,j])$ in $\mathcal{O}(\log(j-i))$ time by finding the location of $w[j]$ in the order using the tree. We can then easily update the tree by adding the element $w[j]$ and removing the element $w[i]$, allowing for the computation of $\mathsf{PrefCode}(w[i+1\,..\,j+1])$. Each time we compute the hash of this $\mathsf{PrefCode}(w[i\,..\,j])$ in $\mathcal{O}(1)$ time (using Karp-Rabin hash [2], and knowing the hash of $\mathsf{PrefCode}(w[i\,..\,j-1])$), and store it in the element $A[i][j]$ of a 2D array A. We can now group all the substrings with the same hash using a hash table. This way we know for each substring $w[i\,..\,j]$ if it is frequent and can easily check if the pattern $w[i\,..\,j-1]$ is right-closed or right-maximal.

Once we reach length $k+1$ when no pattern is frequent, we can stop the computation, and hence we only consider the $\mathcal{O}(nk)$ substrings of length at most $k+1$, spending $\mathcal{O}(\log k)$ time for each one.

Since it is enough to store the array A and the hash tables only for the two next lengths of the substrings (i.e., upon processing the length-$(\ell+1)$ substrings we get to know if the length-ℓ ones are closed/maximal, and when we start processing the length-$(\ell+2)$ substrings we have already reported the patterns of length ℓ and will never access them again) the solution requires only $\mathcal{O}(n)$ words of space using a hash table of $\mathcal{O}(n)$ size and $\mathcal{O}(1)$-time operations [3].

B. Details of the Datasets

In this section, we provide the details of the 6 real-world, large-scale datasets used in our experiments. See Table **??** in the main paper for links to these datasets except ECG that is proprietary and cannot be made public for privacy reasons. The datasets that we discuss below except ECG are also available at https://github.com/LINGLI97/OPST_MS.

HOU is derived from the Individual household electric power consumption dataset (IHEPC), which comprises approximately 2.07 million measurements of electric power consumption for a single house located in Sceaux (7 km from

Paris, France). These measurements were recorded with a one-minute sampling rate over a period of almost 4 years, spanning from December 2006 to November 2010 (a total of 47 months). To construct a large-scale dataset, we concatenated the measurements from 3 sub-meterings.

SOL is derived from the records of solar power production in 2006, with a sampling rate of every 10 minutes from 137 synthetic solar photovoltaic power plants in Alabama State, US. It was preprocessed by [4] and we concatenated the data from all 137 power plants.

ECG is a dataset derived from collected ECG responses during exercise-induced pain in 40 participants. During the experiment, participants sat on a chair with their elbow at a 90º angle and their wrist joint 20 cm above a table surface. They were instructed to hold their Baseline Weight (calculated based on the heaviest weight they could lift) in an isometric contraction for as long as possible. The ECG and Time to Exhaustion (TTE) of participants was measured, based on the duration of the weight-holding task. Additionally, participants' Pain Intensity [5] and their Rating of Perceived Exertion were recorded for every minute during the exercise [6]. As mentioned in the main paper, we used ECG in two ways. In the context of experiments about mining patterns from a single string, we concatenated the data from all participants, producing a string with length $n = 22,973,535$ and alphabet size $\sigma = 31,731$. In the context of experiments about pattern-based clustering, we used ECG as is (i.e, as a collection of strings, one for each participant). The average and maximum length of the strings was $574,338$ and $1,241,745$, respectively, and the alphabet sizes of the strings are in $[6270, 30248]$.

TRA is derived from 48 months of hourly data (2015-2016) provided by the California Department of Transportation. The data describe the road occupancy rates (between 0 and 1) measured by different sensors on San Francisco Bay area freeways. The data was preprocessed by [4] and we concatenated the data from all different sensors.

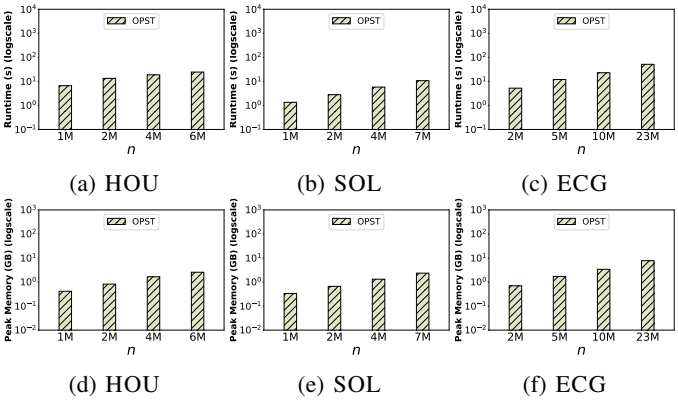
TEM is derived from temperature measurements, captured using an 8×8 infrared array sensor (Panasonic Grid-EYE) over a period of 3 weeks in 2018, located in Bucharest, Romania. The data were recorded at a frequency of 1 Hz, with each frame comprising 64 temperature values in degrees Celsius, corresponding to the 64 cells of the sensing grid. We concatenated the data in the order they were recorded.

WHA is derived from underwater microphone signals, analyzing the energy within the 360-370 Hz frequency band, as released by the National Centers for Environmental Information. It contains passive acoustic recordings of cetaceans collected from areas within the Pacific Islands Region from April 2005 to September 2019. It was preprocessed by [7].

TABLE I: UCR clustering datasets characteristics

Dataset	Sequence Length	# of Sequences	# of Ground Truth Clusters
CinCECGTorso (CCT)	1639	40	4
Wafer (WAF)	152	1000	2

We used 2 time series datasets from different domains for clustering and visualization, obtained from the UCR Archive [8]. Their characteristics are shown in Table I. The sequences in each dataset are of equal length. CCT has two ground truth clusters and WAF four (the time series in each ground truth cluster have the same label).



(a) HOU (b) SOL (c) ECG

(d) HOU (e) SOL (f) ECG

Fig. 1: OPST (construction algorithm): (a-c) Runtime and (d-f) respective peak memory consumption for varying n.

We used 4 stock time series obtained from Yahoo Finance [9]. Their characteristics are shown in Table II.

TABLE II: Stock datasets characteristics

Dataset	n	σ
Apple (AAPL)	6434	6020
Meta (META)	3320	3153
Tesla (TSLA)	3797	3586
Google (GOOGL)	5272	5175

C. Clustering Quality Measures

In the following, we discuss the three widely adopted measures of clustering quality we used, namely Normalized Mutual Information (NMI) [10], Homogeneity (h)[11], and the Rand Index (RI) [12].

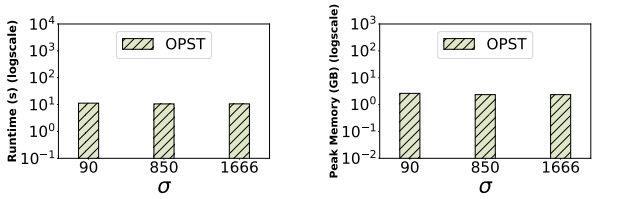
A clustering is a partition of a collection of strings into sets called clusters. We consider two clusterings C, C' with sizes $|C|$ and $|C'|$, respectively, and a collection of N strings. C is the ground truth clustering. NMI, h, and RI quantify how similar C' is to the ground truth clustering. The values in all these measures are in $[0, 1]$, where a higher value indicates that C' is closer to the ground truth clustering.

The NMI between C and C' is defined as:

$$\mathrm{NMI}(C, C') = \frac{2 \cdot \sum_{i=1}^{|C|} \sum_{j=1}^{|C'|} \frac{n_{ij}}{N} \log\left(\frac{n_{ij/N}}{n_i \cdot \hat{n}_j / N^2}\right)}{-\sum_{i=1}^{|C|} \frac{n_i}{N} \log \frac{n_i}{N} - \sum_{j=1}^{|C'|} \frac{\hat{n}_j}{N} \log \frac{\hat{n}_j}{N}},$$

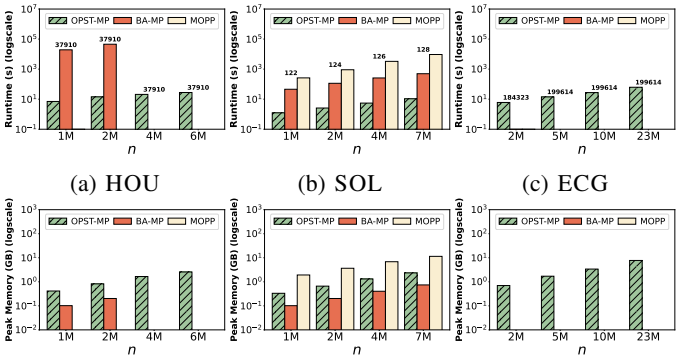
where n_i denotes the number of strings in the ith cluster in C, $\hat{n}_j$ denotes the number of strings in the jth cluster in C', and n_{ij} denotes the number of clusters belonging both in the ith cluster in C and in the jth cluster in C'.

Homogeneity (h) measures how much the strings in a cluster of the clustering C' "agree" with those in a cluster of the ground truth clustering C. A cluster in C' is considered homogeneous (i.e., completely "agrees" with a cluster in C)



(a) SOL (b) SOL

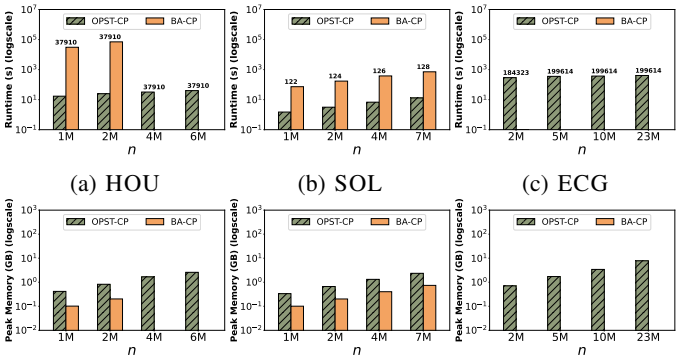
Fig. 2: OPST (construction algorithm): (a) Runtime and (b) peak memory consumption for varying σ.



(a) HOU (b) SOL (c) ECG

(d) HOU (e) SOL (f) ECG

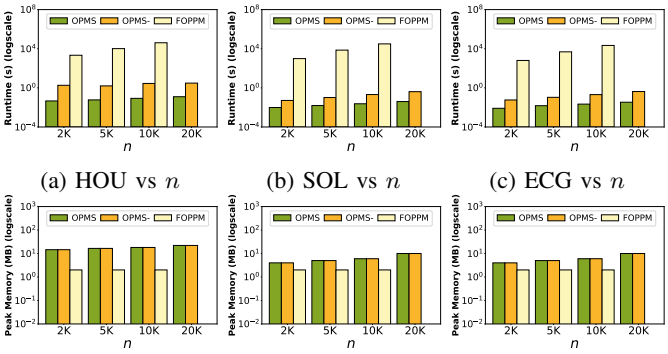
Fig. 3: OPST-MP, BA-MP and MOPP: (a – c) Runtime and (d – f) respective peak memory consumption for varying n. Missing bars for BA-MP and MOPP indicate that they *did not finish within 24 hours.* The value above each pair of bars in (a – c) represents the maximum length k of all τ-maximal τ-frequent OP patterns. τ is set to 3 for HOU, and $\tau = 10$ for the other datasets.



(a) HOU (b) SOL (c) ECG

(d) HOU (e) SOL (f) ECG

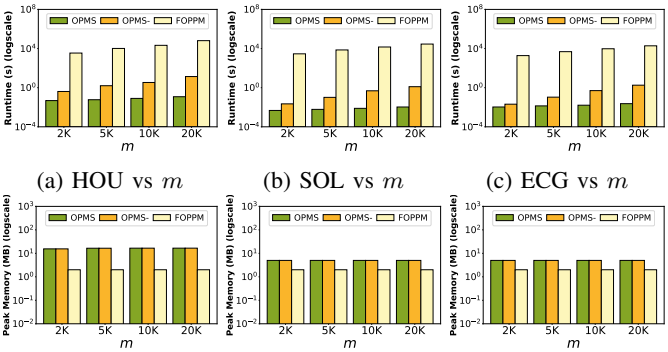
Fig. 4: OPST-CP and BA-CP: (a – c) Runtime and (d – f) respective peak memory consumption for varying n. Missing bars for BA-CP indicate that it *did not finish within 24 hours.* The value above each pair of bars in (a – c) represents the maximum length k of all closed τ-frequent OP patterns. τ is set to 3 for HOU, and $\tau = 10$ for the other datasets.



(a) HOU vs n (b) SOL vs n (c) ECG vs n

(d) HOU vs n (e) SOL vs n (f) ECG vs n

Fig. 5: OPMS, OPMS-, and FOPPM: (a – c) Runtime and (d – f) respective peak memory consumption for varying n ($m = 5$K). Missing bars for FOPPM indicate that it *did not finish within* 24 *hours*.



(a) HOU vs m (b) SOL vs m (c) ECG vs m

(d) HOU vs m (e) SOL vs m (f) ECG vs m

Fig. 6: OPMS, OPMS-, and FOPPM: (a – c) Runtime and (d – f) respective peak memory consumption for varying m ($n = 5$K).

when it consists entirely of strings belonging to one cluster of C. The h measure between C and C' is defined as:

$$h(C, C') = 1 - \frac{-\sum_{i=1}^{|C|}\sum_{j=1}^{|C'|}\frac{n_{ij}}{N}\log\left(\frac{n_{ij}/N}{\hat{n}_j/N}\right)}{-\sum_{i=1}^{|C|}\frac{n_i}{N}\log\frac{n_i}{N}},$$

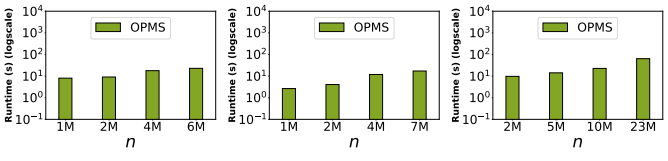
where n_i, n_{ij}, $\hat{n}_j$, and N are as defined above.

The Rand Index (RI) measures similarity between the clustering C' and the ground truth clustering C by counting the number of pairs of strings that are placed in the same clusters in C and C', in different clusters in C and C', or in the same clusters in one of the C and C' and in different clusters in the other. $\text{RI}(C, C')$ is defined as follows:

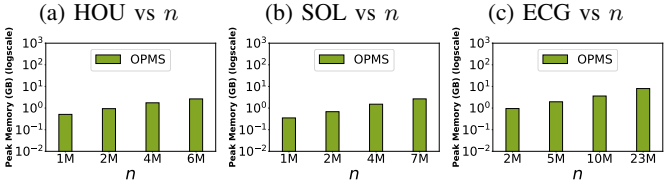
$$\text{RI}(C, C') = \frac{a+b}{a+b+c+d},$$

where:

- a is the number of pairs of strings that are in the same cluster in C and C'.

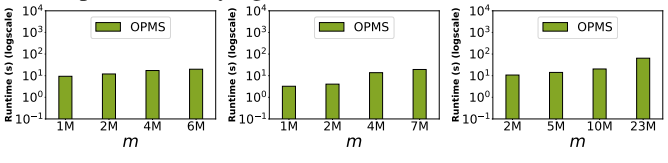


(a) HOU vs n (b) SOL vs n (c) ECG vs n

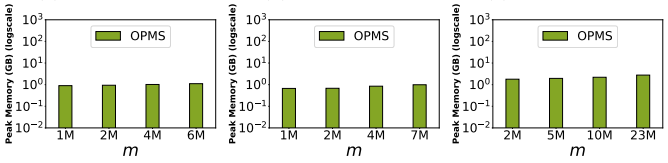


(d) HOU vs n (e) SOL vs n (f) ECG vs n

Fig. 7: OPMS: (a – c) Runtime and (d – f) peak memory consumption for varying n.



(a) HOU vs m (b) SOL vs m (c) ECG vs m



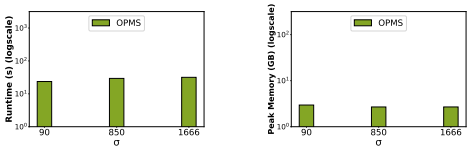
(d) HOU vs m (e) SOL vs m (f) ECG vs m

Fig. 8: OPMS: (a – c) Runtime and (d – f) peak memory consumption for varying m.

- b is the number of pairs of strings that are in different clusters in both C and C'.
- c is the number of pairs of strings that are in the same cluster in C but in different clusters in C'.
- d is the number of pairs of strings that are in different clusters in C but in the same cluster in C'.

D. Additional Experimental Results

Figs. 1 to 4 show the impact of σ and n on the runtime and peak memory consumption (construction space) of all methods using the datasets that are not included in the main paper. The results are consistent with those presented in the main paper. Experiments measuring the runtime and peak memory consumption for varying σ on HOU and TEM were not conducted because σ is only 88 and 76, respectively, and these values are too small to be meaningfully discretized.



(a) SOL vs σ (b) SOL vs σ

Fig. 9: OPMS: (a) Runtime and (b) peak memory consumption for varying σ.

Figs. 5 to 8 show the impact of n, m, and σ on the runtime and peak memory consumption (construction space) of all methods using the datasets that are not included in the main paper. The results are analogous to those in the main paper.

E. Details of Stock Sequences

Here are the detailed dates of all sequences shown in Fig. **??** of the main paper.

For AAPL: $S1$ is from 2010-03-19 to 2010-04-09; $S2$ is from 2024-11-19 to 2024-12-10; $S3$ is from 2002-03-01 to 2002-03-21; and $S4$ is from 2002-05-23 to 2002-06-13.

For META: $S1$ is from 2015-08-28 to 2015-09-18; $S2$ is from 2025-01-10 to 2025-01-31; $S3$ is from 2012-09-13 to 2012-10-03; and $S4$ is from 2012-10-01 to 2012-10-19.

For TSLA: $S1$ is from 2020-12-18 to 2021-01-11; $S2$ is from 2023-05-22 to 2023-06-12; $S3$ is from 2010-09-08 to 2010-09-28; and $S4$ is from 2014-06-27 to 2014-07-18.

For GOOGL: $S1$ is from 2012-09-11 to 2012-10-01; $S2$ is from 2025-07-09 to 2025-07-29; $S3$ is from 2005-09-19 to 2005-10-07; and $S4$ is from 2006-01-26 to 2006-02-15.

REFERENCES

[1] E. M. McCreight. "A Space-Economical Suffix Tree Construction Algorithm". In: *J. ACM* (1976), pp. 262–272.

[2] R. M. Karp and M. O. Rabin. "Efficient randomized pattern-matching algorithms". In: *IBM J. R&D* 31.2 (1987), pp. 249–260.

[3] M. A. Bender et al. "Iceberg Hashing: Optimizing Many Hash-Table Criteria at Once". In: *J. ACM* (2023), 40:1–40:51.

[4] G. Lai et al. "Modeling long-and short-term temporal patterns with deep neural networks". In: *SIGIR*. 2018, pp. 95–104.

[5] D. B. Cook et al. "Naturally occurring muscle pain during exercise: assessment and experimental evidence". In: *Med Sci Sports Exerc.* 29.8 (1997), pp. 999–1012.

[6] G. Borg. *Introduction to Time Series and Forecasting*. 1998.

[7] M. Ceccarello and J. Gamper. "Fast and Scalable Mining of Time Series Motifs with Probabilistic Guarantees". In: *PVLDB* (2022), pp. 3841–3853.

[8] H. A. Dau et al. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Oct. 2018.

[9] https://finance.yahoo.com/.

[10] N. X. Vinh, J. Epps, and J. Bailey. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance". In: *JMLR* 11 (2010), pp. 2837–2854.

[11] A. Rosenberg and J. Hirschberg. "V-Measure: A conditional entropy-based external cluster evaluation measure". In: *EMNLP-CoNLL*. 2007, pp. 410–420.

[12] W. M. Rand. "Objective criteria for the evaluation of clustering methods". In: *JASA* 66.336 (1971), pp. 846–850.