Polynomial Approximation and Critical Point Sampling for Time Series Clustering

Charles Wang*

Project Advisor: Nan Li

Abstract. Clustering time series is often computationally expensive because most methods require evaluating distances between all pairs of sequences, especially when using measures such as Dynamic Time Warping. This scaling limits the applicability of standard algorithms to large or high-resolution data sets. We propose an efficient framework that represents each time series through polynomial approximation and computes similarity using only the resulting local critical points. This yields compact, structure-preserving summaries that dramatically reduce the cost of pairwise comparisons while retaining essential temporal patterns. The resulting distance measure enhances the extraction of salient dynamical features, suppresses noise, and maintains clustering quality comparable to traditional approaches at a much lower computational cost. Numerical experiments on real data demonstrate that the method offers substantial improvements in efficiency without compromising interpretability or accuracy.

Key words. Time series clustering, dimension reduction, distance, critical points analysis, structure-preserving representations

MSC codes. 62H30, 68T10, 68Q25

1. Introduction.

Time series data arise in a wide range of applications, including finance, healthcare, sensor networks, and scientific simulations. Analyzing such data often requires unsupervised methods for grouping similar sequences, making time series clustering a fundamental task in data science. Extensive research has been conducted in either finding a new clustering algorithm, or developing time-series-specific distance measures. Comprehensive review can be found in [1, 10, 14, 23].

Despite its practical importance, clustering of time series remains computationally challenging, primarily due to the high cost of evaluating pairwise similarity measures across long sequences. Let $\{X_1, X_2, \ldots, X_m\}$ denote a set of m time series, each of length n. Standard approaches compute pairwise distances between all series, resulting in $O(m^2 \cdot distance\_cost)$ operations. When using the Euclidean distance [5], the distance cost is linear with n, and the computational cost scales as $O(m^2 n)$. For methods that account for temporal misalignment, such as Dynamic Time Warping (DTW) [19], the distance cost is quadratic of length n, and the cost increases to $O(m^2 n^2)$, rendering large-scale clustering impractical. For example, clustering thousands of sequences of moderate length can require trillions of operations, even before the actual clustering algorithm is applied.

To address this challenge, we propose a structure-preserving dimensionality reduction framework for time series clustering. Our approach represents each series as a low-degree polynomial curve, capturing its underlying trend. From this representation, we extract local critical points, i.e., locations where the curve exhibits significant changes, to summarize the series. Pairwise distances are then computed using only these critical points, reducing

*Bridgewater-Raritan High School, Bridgewater, NJ (charles.wang.v6@gmail.com).

computational complexity to $O(m^2d)$ where d is the degree of the polynomial. The proposed distance is referred to as Distance via Critical Points (DCP) throughout this paper. The advantage of doing this is two-fold: 1) By focusing on critical data points, the useful signal is amplified and the noise in the data is ignored. It is due to this reason, the developed method has comparable accuracy to other methods. 2) The computational complexity is significantly reduced compared to traditional methods, therefore it can process a large volume of data quickly.

The rest of this paper is organized as follows: Section 2 reviews related work in time series clustering. Section 3 describes the proposed framework in detail, including polynomial representation, critical point extraction, distance computation, and its application using k-medoid algorithm. Section 4 presents numerical experiments evaluating both clustering quality and computational efficiency. Finally, Section 5 concludes with a discussion of potential extensions.

2. Related Work.

In this section, we review various clustering algorithms and the traditional distance measures for time series.

2.1. Clustering Algorithms.

Clustering algorithms can be generally categorized into partitioning-based methods such as k-means [8] and k-medoids [7], hierarchical clustering [13] (agglomerative or divisive), density-based clustering, [4], and spectral clustering [6]. Each class of algorithms offers distinct advantages and limitations. Hierarchical clustering supports custom distance measures and provides intuitive structure visualizations but is computationally expensive and irreversible once merges or splits occur. Partitioning-based methods scale well but require the number of clusters in advance and struggle with non-convex shapes. Density-based methods capture arbitrarily shaped clusters and handle noise, yet depend heavily on parameter tuning and perform poorly with varying densities. Spectral clustering handles complex structures but is computationally demanding and sensitive to how the similarity graph is constructed.

Clustering with time series data is not much different from normal clustering. Because the idea of "shape" is not defined rigorously, people have devoted most of their efforts into creating various distance measures built to suit their own needs. As such, most research has focused on developing time-series-specific distance measures. It is commonly believed that finding a good distance measure is as important as a good clustering algorithm [12]. Therefore, time series clustering relies mostly on common clustering algorithms while implementing a fitting distance measure.

2.2. Distance Measures in Time Series Clustering.

Many distance measures exist for time-series comparison, each with advantages and limitations. Among them, Euclidean distance (ED) [5] remains one of the most widely used due to its simplicity, interpretability, and computational efficiency. By directly comparing point-to-point differences, ED is easy to implement but highly sensitive to temporal misalignments. The L_∞ distance [10] offers an alternative perspective by capturing the maximum absolute difference between corresponding points. This makes it robust to small fluctuations but prone to overemphasizing single outliers and overlooking overall shape similarity.

Dynamic Time Warping (DTW) [19] overcomes temporal misalignment by stretching or

compressing segments in time, allowing for robust distance estimation even under phase shifts. Because of this flexibility, DTW is often considered the gold standard for distance measures. However, the quadratic cost in sequence length makes DTW prohibitively expensive for large datasets. To reduce the complexity of computation, DTW is generally constrained by limiting warping to a specific part of the time series, helping preserve local structure. This is called constrained DTW (cDTW). Numerous approximation and acceleration techniques have also been proposed to further improve scalability [20, 16, 11].

Complementing these alignment-based strategies, polynomial approximation and critical point analysis address the complexity issue from a different angle. Polynomial curves offer smooth, compact summaries of continuous signals [17], while critical points capture salient structural changes [9]. Our method integrates these ideas to represent time series as polynomial curves and compute distances using their critical points, improving scalability while preserving essential patterns.

3. Proposed Method.

The proposed representation reframes how similarity is measured for clustering. While static data clustering relies on spatial proximity, time series clustering must capture evolving temporal dynamics. Rather than performing pointwise comparisons like DTW, we compare only the critical points that summarize each series' key structural features.

3.1. Polynomial Approximation.

To obtain a smooth and noise-reduced representation, each time series is approximated by a polynomial.

$$y(x) = \sum_{i=0}^{d} \beta_i x^i$$

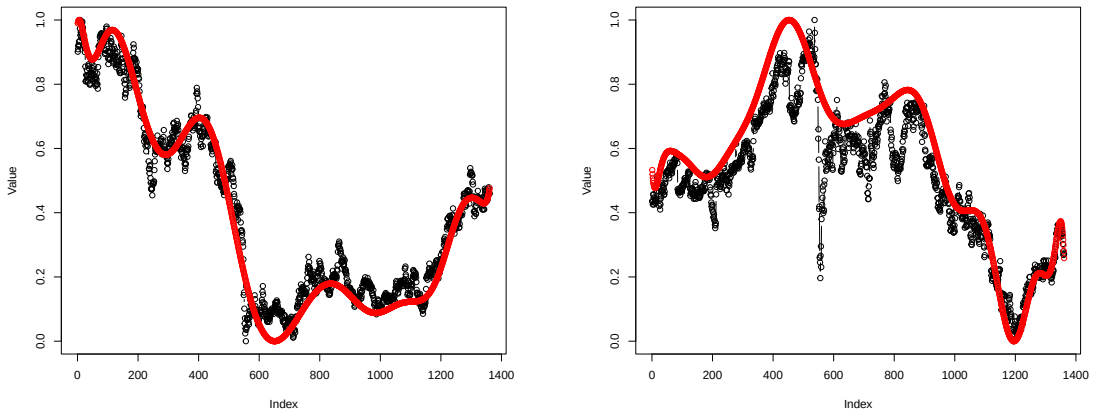
where:

d	is the degree of the polynomial,
β_i	is the coefficient for the i-th term.

This approximation serves two purposes: it yields a compact functional representation of the signal and enables the extraction of meaningful structural features, such as local extrema and inflection points, which are later used as critical points for distance computation. By capturing the dominant shape of the series, the polynomial fit highlights the underlying trend while suppressing small, high-frequency fluctuations that typically reflect noise rather than substantive variation.

Figure 3.1 shows the daily prices of two randomly selected stocks during the COVID period, a dataset that will also be used in our numerical experiments for evaluating results in Section 4. The polynomial approximation smooths the sharp, short-lived dip while preserving the overall trend of the series.

To select an appropriate polynomial degree, we evaluate the mean absolute percentage error (MAPE) [15] across different approximation orders. Using a 10% MAPE threshold as a baseline for an acceptable fit, we find that both the median and mean optimal degrees for stocks during this period cluster around five. This suggests that a fifth-degree polynomial generally provides a suitable balance between representational accuracy and model complexity, capturing essential dynamics without overfitting.



(a) Random Stock 1 (b) Random Stock 2

Figure 3.1: Polynomial approximation of two randomly selected stocks during the COVID period.

Next, we normalize the time series to ensure they are on a common scale, independent of their absolute values. Without normalization, series with large magnitudes could dominate the distance calculations, obscuring meaningful similarities in overall shape or temporal patterns. By scaling each series to a fixed range, we emphasize relative changes and trends, enabling more robust and interpretable clustering results.

The following min-max normalization is used to scale the time series values in the range of $[0, 1]$.

$$f(x) = \frac{y(x) - \min\limits_{x} y(x)}{\max\limits_{x} y(x) - \min\limits_{x} y(x)}$$

3.2. Proposed DCP Algorithm. The proposed distance measure, Distance via Critical Points (DCP), is defined as follows. For each time series after normalization $f(x)$, where $x \in [a, b]$, we determine its set of critical points

$$CP_f = \{x : f'(x) = 0\} \cup \mathcal{A}.$$

where $\mathcal{A}$ is the auxiliary set usually set to be $\mathcal{A} = \{a, b\}$ or $\mathcal{A} = \{a, (a + b)/2, b\}$ to include the starting, ending and/or mid-point of the curve, in case $\{x : f'(x) = 0\}$ is empty.

Similarly, for another time series $g(x)$, we compute the set of critical points as CP_g. The DCP between f and g is then given by the average absolute difference at all critical points of either series:

$$\mathrm{DCP}(f, g) = \frac{1}{|CP_f \cup CP_g|} \sum_{x \in CP_f \cup CP_g} \big|f(x) - g(x)\big|.$$

By focusing only on these critical points, DCP is essentially comparing those "turning points" of the curves, which characterize the trends of the time series. The advantage is that by capturing these key features of the data, the computation becomes much simpler and faster than other distance measures, making it well-suited for large-scale time series analysis.

To obtain the critical points of a time series $x[i], i = 1, ...n$, a numerical method is to calculate the difference of two adjacent data points, i.e., $\Delta x[i] = x[i+1] - x[i]$, and find the places where the signs of $\Delta x[i]$ change. The complexity of this process is linear with the length of the time series n. However, since the curves are now represented by polynomials, there is an alternative way of calculating the critical points. Assume the time series is approximated by a polynomial $f(x)$ of degree d, i.e.,

$$f(x) = \beta_d x^d + \beta_{d-1} x^{d-1} + ... + \beta_1 x + \beta_0.$$

The derivative of $f(x)$ is simply:

$$f'(x) = \beta_d d x^{d-1} + \beta_{d-1}(d-1)x^{d-2} + ... + \beta_1.$$

Closed-form solutions exist in finding the roots of $f'(x)$ if $d - 1 \leq 4$ and the complexity is almost negligible. If $d-1 \geq 5$, various numerical methods can be used to find the roots [21, 22]. The general complexity is $O((d-1)^3)$. Therefore, the complexity of finding the critical points is $O(\min(n, (d-1)^3))$. It is noted that under the proposed DCP distance, this calculation is performed only once for all time series at the preprocessing stage. The burden of calculation is shifted to this stage from the pairwise distance calculation, making the clustering much faster than other methods. The details of the analysis will be shown in Section 4.3.

The defined DCP distance, however, is not applicable to the popular k-means clustering algorithm. The reason is that the k-means algorithm calculates an average distance, generally in the form of Euclidean distance. But since our DCP distance relies on the critical points of each time series, we have to use actual data when calculating the centroids. The constraint naturally leads to the employment of k-medoids algorithm, which is a partitional clustering algorithm similar to k-means, but it selects actual data points—called *medoids*—to serve as cluster centers.

Let $X = \{X_1, X_2, \ldots, X_m\}$ be the set of m time series. Each of X_i in the set is a vector of length n. Let $DCP(X_i, X_j)$ be a distance function defined between X_i and X_j. The goal is to choose a set of k medoids $M = \{m_1, m_2, \ldots, m_k\} \subseteq X$ that minimizes the total dissimilarity:

$$\sum_{i=1}^{m} D(X_i, m_{c(i)})$$

where $m_{c(i)}$ is the medoid with a centroid index of $c(i)$ ssociated with point X_i.

The steps of the algorithm are detailed in Algorithm 3.1.

4. Results.

Unlike supervised learning, clustering typically lacks ground truth labels, making validation a challenging process. Very often, validation has to rely on the data itself without using external labels. While numerous techniques exist for cluster validation, internal validation indices such as the Silhouette Score [18] and Calinski–Harabasz Index [3] are among the most commonly used to evaluate compactness and separation of clusters.

Algorithm 3.1 k-Medoid Clustering with DCP Distance

Input: Data points $X = \{X_1, X_2, \ldots, X_m\}$, number of clusters k
Output: Cluster assignments and medoids $M = \{m_1, \ldots, m_k\}$
Initialization: Randomly select k data points as initial medoids $M = \{m_1, \ldots, m_k\}$
repeat
  Distance: Compute the DCP distance from each point X_i to each medoid in M
  Assignment: Assign each X_i to the nearest medoid
  for each cluster c **do**
    Update: Select the point in c that minimizes the total DCP distance to all other points in c and set it as the new medoid
  end for
until cluster assignments do not change or the cost function converges

4.1. Cluster Validation. In this study, we use the Silhouette Score to evaluate our clusterings. The silhouette score is a metric used to evaluate the quality of a clustering solution. It measures how similar an object is to its own cluster compared to other clusters. Values range from -1 to 1. A score of 1 indicates that the point is well matched to its own cluster and poorly matched to neighboring clusters (ideal). A score of 0 indicates that the point lies on the boundary between clusters, and a score of -1 infers that the point may have been assigned to the wrong cluster.

The score is computed as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i),\, b(i)\}}, \qquad S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

where:

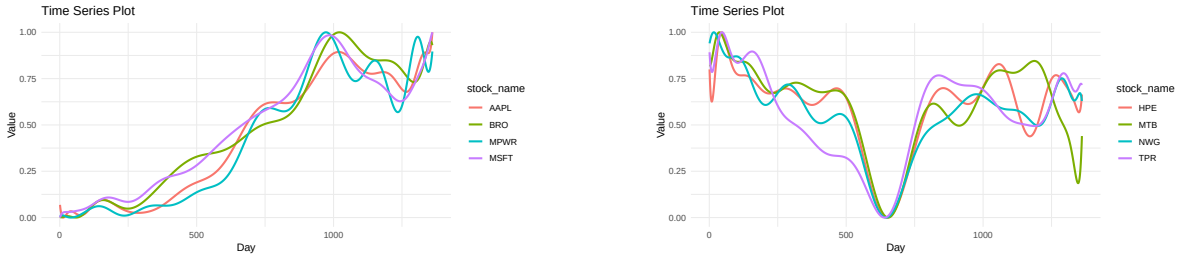
$a(i)$ is the average distance from point i to all other points in the same cluster,
$b(i)$ is the smallest average distance from point i to all points in any other cluster,
n is the total number of samples,
S is the overall silhouette score for the dataset.

4.2. Numerical Experiments with Stocks. We collect stocks listed in Nasdaq from Jan 1, 2018 to May 31, 2023, and use the daily close price as the data points in the time series. The length of the time series is $n = 1361$.

We select a set of 20 stocks that can be visually grouped into five c lusters. Clustering is performed using the k-medoid algorithm with both the proposed DCP measure and the traditional DTW. The number of clusters is assumed to be known a priori as five.

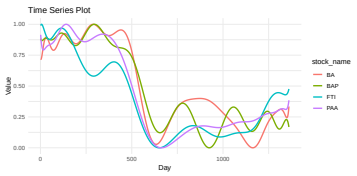
Figure 4.1 shows the clustering results using DCP, highlighting a clear separation of the stocks. Notably, the DCP-based clustering preserves the natural grouping and temporal patterns of the series compared to DTW, demonstrating the effectiveness of the proposed distance measure in capturing the intrinsic structure of the data.

Furthermore, Silhouette scores are computed to compare the clustering performance of DCP and DTW. The evaluation scores in Table 4.1 show that DCP achieves a Silhouette Score

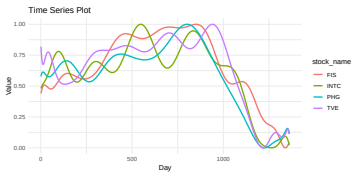


(a) Cluster 1

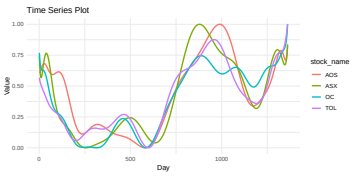
(b) Cluster 2



(c) Cluster 3



(d) Cluster 4



(e) Cluster 5

Figure 4.1: Clustering Results of the 20 Stocks

Table 4.1: Comparison of the highest silhouette scores for DCP and DTW.

Method	Silhouette Score
DCP	0.5928
DTW	0.5963

of 0.5928, which is very close to DTW's score of 0.5963, demonstrating its competitiveness in clustering quality.

In scenarios where the number of clusters is unknown, the optimal value of k can be identified by evaluating the clustering performance across different candidate values and selecting the one that maximizes the Silhouette score. Figure 4.2 presents the Silhouette scores corresponding to various choices of k. The peak score occurs at $k = 5$, which aligns with the underlying structure of the dataset, confirming that the 20 stocks naturally partition into five distinct clusters.

4.3. Complexity Analysis. The proposed DCP method runs significantly faster than both Dynamic Time Warping (DTW) and Euclidean Distance (ED). This improvement is evident from the complexity comparison in Table 4.2, which reports the computational cost of constructing an $m \times m$ distance matrix. Here, m denotes the number of time series, each containing n data points, and d represents the degree of the polynomial used in the approximation.

The complexity of DCP, $O(min(n, (d-1)^3)m + dm^2)$, consists of two terms. The first term, $min(n, (d-1)^3)m$, corresponds to the cost of identifying the zero derivatives for each time series. The second term, dm^2, accounts for the pairwise distance computations. The

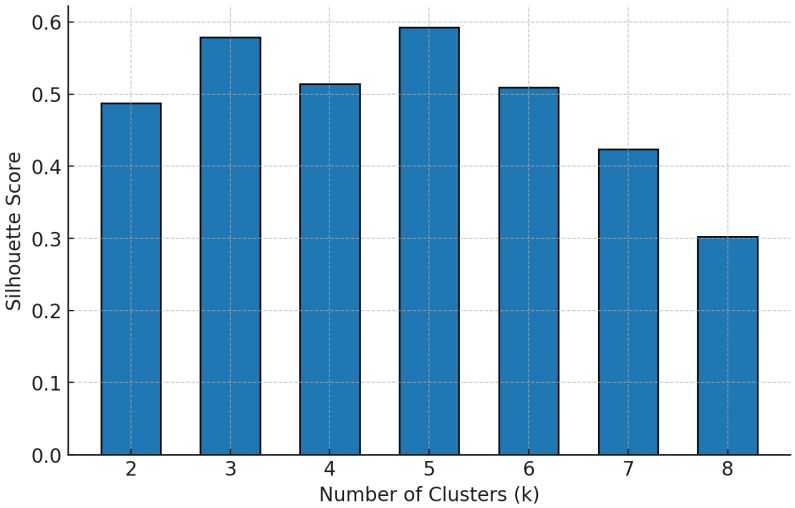


Figure 4.2: Silhouette Scores for DCP with Varying Number of Clusters

Distance Measure	**Time Complexity**	**Notes**
DCP	$O(min(n,(d-1)^3)m + dm^2)$	$d \ll n$, uses only critical points
L_∞ approx.	$O(min(n,(d-1)^3)m^2)$	L_∞ with polynomial approximation
ED	$O(nm^2)$	Requires element-wise comparison
L_∞	$O(nm^2)$	Standard L_∞
DTW	$O(nbm^2)$	Dynamic programming with a band of b

Table 4.2: Time complexity of various distance measures, where the m^2 term reflects the cost of computing all pairwise distances.

$min(n,(d-1)^3)$ factor reflects two possible approaches for detecting zero derivatives. If the extrema are located via a linear scan of local minima and maxima — by checking sign changes in $x[j+1]-x[j]$ for each $1 \le j < n$ - the complexity is $O(n)$. Alternatively, when polynomial approximation is used, the zero derivatives can be obtained by finding the roots of the derivative of a degree-d polynomial. This requires $O((d-1)^3)$ operations [22]. For small d, the polynomial-based method can yield a substantial performance improvement.

The second term, dm^2, assumes the number of zero derivatives is on the order of the polynomial degree d, which is generally an overestimate. Consequently the overall complexity expression should be interpreted as an upper bound rather than the exact cost.

The same principle can be applied to the L_∞ distance to accelerate its computation. We denote the resulting method by L_∞ approx. to distinguish it from the standard L_∞ distance. With polynomial approximation, the L_∞ distance is defined as

$$\mathrm{L}_\infty(f,g) = \max\big|f(x) - g(x)\big|.$$

where $f(x) - g(x)$ is a polynomial. As discussed above, the maximum of this polynomial can

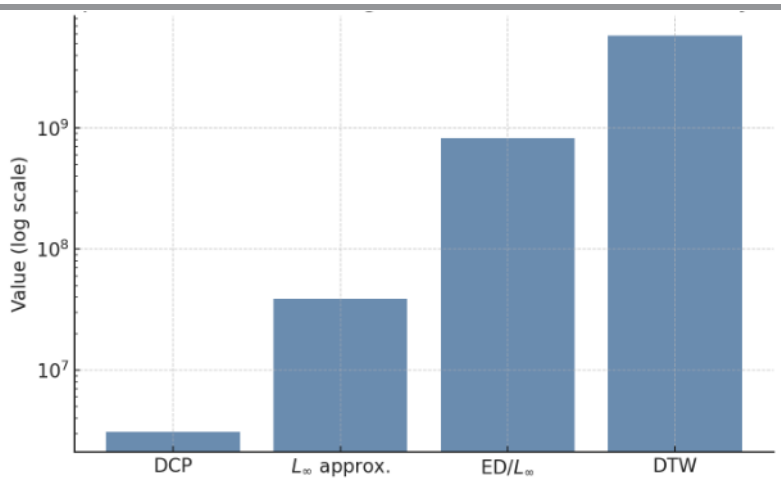


Figure 4.3: Comparison of complexity for various distance measures, with m=800, n=1361, d=5, b=7.

be found either by a linear search ($O(n)$) or by locating its zero derivatives ($O((d-1)^3)$). Therefore, the complexity of L_∞ approx. is $O(min(n,(d-1)^3)m^2)$.

The complexity of Euclidean and the standard L_∞ distance are both nm^2 as they require evaluating all n data points for each pair of time series.

For DTW, the complexity in Table 4.2 assumes a Sakoe-Chiba band of b with $b < n$. Without this constraint, i.e., $b = n$, the complexity increases to $O(n^2m^2)$.

Figure 4.3 compares the computational complexities of the different schemes. The proposed DCP method is significantly faster than all of the alternatives. Notably, L_∞ approx. can be substantially faster than the standard L_∞ when the polynomial degree d is small.

4.4. Stability Improvement. K-medoids, like k-means, can be sensitive to the initial choice of medoids. This can lead to different cluster assignments across runs, even when using the same dataset. This is especially prominent when clusters are not well-separated. This instability makes results harder to interpret and reproduce. To address this, the following stability-focused improvements can be made.

1. **Multiple initializations** – Run k-medoids several times with different random seeds, computing the total clustering cost (i.e., the sum of distances to the medoids) for each run. The solution with the lowest cost is then selected. Because the DCP distance can be computed efficiently, this strategy is computationally affordable.
2. **Refined initialization** – Instead of selecting medoids at random, initial medoids can be chosen using a distance-aware procedure that favors diverse and well-separated starting points, similar to the approach in [2]. This improves initialization quality while adding only modest computational overhead.

4.5. Accuracy-Efficiency Trade-offs. Several factors influence the balance between computational efficiency and accuracy.

Auxiliary Evaluation Points. The proposed method evaluates curves only at critical

points, which has been shown to be effective. If there is concern that differences between curves may occur in intermediate regions, auxiliary evaluation points can be introduced. For example, inserting midpoints between consecutive critical points enables a finer-grained comparison. More generally, subdividing each interval into additional segments increases the number of evaluation points proportionally, improving accuracy at the cost of additional computation.

Polynomial Degree. Different applications of the proposed method may require different degrees. Applications such as short-term financial time series data, which typically exhibits locally smooth trends with modest curvature over short horizons, may only need a degree 2-3 polynomial for effective results. Long-term financial time series data may require degree of 4 or 5, as data often exhibits multiple phases, such as growth, stagnation, and decline. Data that often exhibits more frequent fluctuations and localized variations, such as hourly or daily weather data, may need higher-degree polynomials to capture the additional curvature and multiple inflection points present in the data.

Increasing the polynomial degree may improve approximation accuracy and typically produces more critical points for comparison. However, these improvements often exhibit diminishing returns as the degree increases. If acceptable MAPE cannot be achieved even with relatively high degrees ($d > 10$), this suggests that polynomial regression may not be well suited for the underlying data. In such cases, preprocessing techniques such as moving averages can provide an alternative approximation. Critical points can then be approximated by identifying sign changes in the discrete time series, as discussed in Section 4.3.

5. Conclusion.

This paper introduces a fast and effective approach for time series clustering based on polynomial approximation and critical point extraction. By emphasizing the most informative temporal structures and suppressing redundant or noisy data, the method delivers clustering performance comparable to computationally intensive measures such as DTW at a fraction of the cost. This efficiency makes the approach particularly well suited for large-scale datasets and applications such as financial time series analysis, sensor monitoring, and biomedical signal processing.

The proposed DCP distance is not limited to k-medoids. Its compact, structure-preserving representation of time series allows seamless integration with other clustering paradigms, including hierarchical, density-based, and graph-based methods. Exploring these extensions—and assessing their theoretical and practical properties—constitutes an important direction for future work.

REFERENCES

[1] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, *Time-series clustering – a decade review*, Information Systems, 53 (2015), pp. 16–38.

[2] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[3] T. Caliński and J. Harabasz, *A dendrite method for cluster analysis*, Commun. Stat. Theory Methods, 3 (1974), pp. 1–27.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, in Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96), AAAI Press, 1996, pp. 226–231.

[5] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, *Fast subsequence matching in time-series databases*, in SIGMOD, 1994, pp. 419–429.
[6] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, *A survey of kernel and spectral methods for clustering*, Pattern Recognition, 41 (2008), pp. 176–190.
[7] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, 1990.
[8] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: An introduction to cluster analysis*, vol. 344, John Wiley & Sons, 2009.
[9] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, *Derivative-based time series representations*, Data Mining and Knowledge Discovery, 7 (2003), pp. 1–30.
[10] E. Keogh and S. Kasetty, *On the need for time series data mining benchmarks: A survey and empirical demonstration*, Data Mining and Knowledge Discovery, 7 (2003), pp. 349–371.
[11] E. Keogh and C. A. Ratanamahatana, *Exact indexing of dynamic time warping*, Knowledge and Information Systems, 7 (2005), pp. 358–386.
[12] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, *An efficient complexity-invariant distance for time series*, in Proc. of the 31st Int. Conf. Very Large Data Bases (VLDB), VLDB Endowment, 2005, pp. 882–893.
[13] G. N. Lance and W. T. Williams, *A general theory of classificatory sorting strategies: 1. hierarchical systems*, The Computer Journal, 9 (1967), pp. 373–380.
[14] T. W. Liao, *Clustering of time series data – a survey*, Pattern Recognition, 38 (2005), pp. 1857–1874.
[15] A. D. Myttenaere, B. Golden, B. L. Grand, and F. Rossi, *Mean absolute percentage error for regression models*, Neurocomputing, 192 (2016), pp. 38–48, https://doi.org/10.1016/j.neucom.2015.12.114.
[16] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, *Searching and mining trillions of time series subsequences under dynamic time warping*, in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 262–270.
[17] J. O. Ramsay and B. W. Silverman, *Functional Data Analysis*, Springer, New York, 2 ed., 2005.
[18] P. J. Rousseeuw, *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*, J. Comput. Appl. Math., 20 (1987), pp. 53–65.
[19] H. Sakoe and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing, 26 (1978), pp. 43–49.
[20] S. Salvador and P. Chan, *Toward accurate dynamic time warping in linear time and space*, Intelligent Data Analysis, 11 (2007), pp. 561–580.
[21] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997.
[22] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.
[23] Z. Xing, J. Pei, and E. Keogh, *A brief survey on sequence classification*, ACM SIGKDD Explorations Newsletter, 12 (2010), pp. 40–48.