Time-Series Clustering and Segment Analysis on PulseDB

1. Project Overview

Problem Statement

The continuous high-dimensional time series data from Arterial Blood Pressure (ABP) and Photoplethysmogram (PPG) and Electrocardiogram (ECG) measurements show intricate patterns throughout their time-dependent behavior. The majority of machine learning models depend on manual label assignment and specific heuristics because clinical datasets lack sufficient supervision. The system uses divide-and-conquer algorithms to perform unsupervised clustering on PulseDB/VitalDB time-series segments through algorithmic reasoning instead of black-box learning.

The system aims to achieve three main objectives:

The system uses recursive divide-and-conquer clustering to group similar physiological segments. - The system uses signal similarity analysis to verify cluster validity through identification of the closest signal pair. - The system uses Kadane's algorithm to identify the most active or anomalous time periods in each signal. The system produces visual outputs that help users understand how different segments relate to each other and what patterns they share.

Dataset

- **Source:** PulseDB / VitalDB AAMI subset (Rutgers University, Kaggle release)
- **Data:** 1000 ten-second ABP segments
- **Format:** MATLAB v7.3 HDF5 `.mat` files (`Subset/Signals`)
- **Sampling Rate:** ~125 Hz
- **Channels:** ABP, ECG, PPG (shape = 1250 × 3 × 666 in test subset)

2. Algorithms Implemented

2.1 Divide-and-Conquer Clustering

The algorithm uses recursive partitioning to divide the dataset into clusters through time-series similarity assessment. The algorithm performs recursive data segmentation until it reaches the specified cluster size threshold which serves as the stopping condition.

Steps: 1. Compute pairwise distances (DTW, correlation, or Euclidean).
2. Select a median-based pivot to divide the dataset.

3. Recurse on left and right partitions.
4. Stop when cluster size ≤ threshold (e.g., 10).

Time Complexity (Optimized):
[T(n) = O(n ^2 n)] using correlation-based similarity and cached distances.

2.2 Closest-Pair Algorithm

For each cluster, the algorithm identifies the two most similar time-series segments. This helps assess cluster cohesion and choose representative examples.

Steps:

1. Compute pairwise distances within each cluster.
2. Select the minimum distance pair.
3. Return pair indices and similarity score.

Complexity:
- Naïve: (O(n^2))
- Divide-and-Conquer Optimization: (O(n n))

2.3 Kadane's Algorithm (Maximum Subarray Sum)

Kadane's algorithm identifies the interval of maximum cumulative change within each signal. This highlights periods of peak activity (e.g., systolic blood pressure peaks).

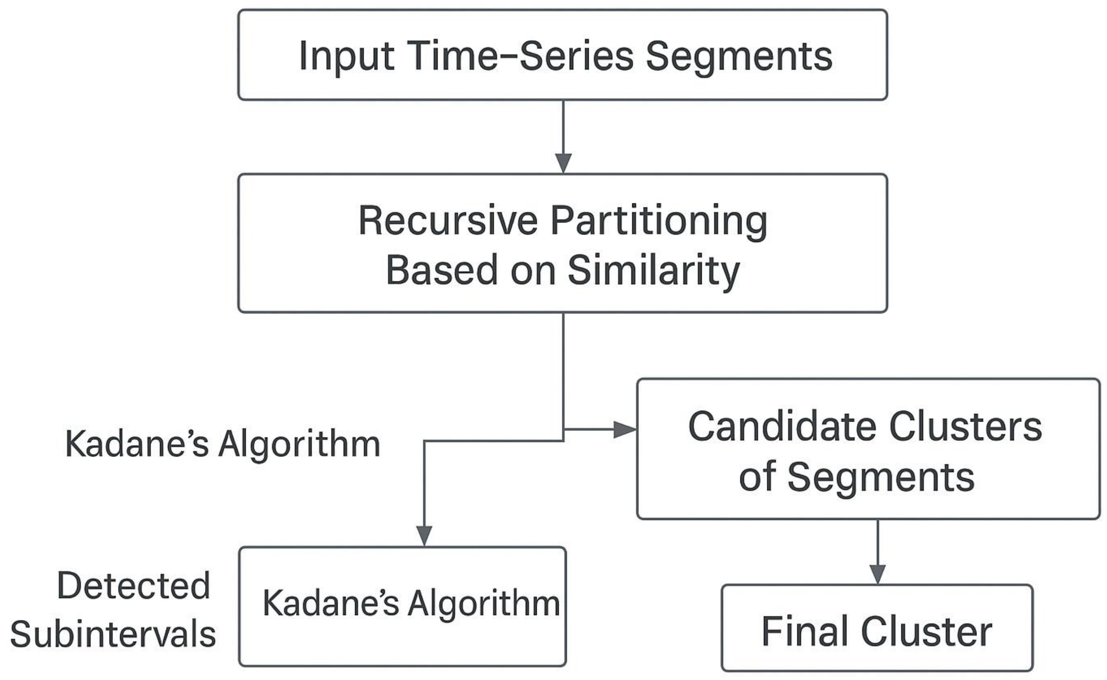
Algorithm:

```
max_so_far = float('-inf')
max_ending_here = 0
for i in range(len(signal)):
    max_ending_here = max(signal[i], max_ending_here + signal[i])
    max_so_far = max(max_so_far, max_ending_here)
```

Complexity: (O(n)) per signal
Purpose: Feature extraction and physiological event localization.

3. System Architecture and Flowchart

Block Diagram


Input Time-Series Segments
Recursive Partitioning Based on Similarity
Kadane's Algorithm
Candidate Clusters of Segments
Detected Subintervals
Kadane's Algorithm
Final Cluster

4. Developed Classes and Their Functions

Class / Module	Purpose	Key Methods / Functions
`main.py`	Coordinates full workflow: loads data, clusters, applies Kadane, generates reports.	`main()`
`utils.py`	Handles data loading from `.mat` files and plotting.	`load_dataset()`, `plot_clusters()`
`clustering.py`	Implements divide-and-conquer	`divide_and_conquer_cluster()`, `similarity()`

Class / Module	Purpose	Key Methods / Functions
	clustering and similarity functions.	
`analysis.py`	Implements closest-pair and Kadane's algorithms.	`find_closest_pair()`, `kadane_max_subarray()`

5. Installation and Usage

Requirements

```
python>=3.10
numpy
scipy
matplotlib
fastdtw
seaborn
h5py
```

Install all dependencies:

```
pip install -r requirements.txt
```

Run the Project

```
python src/main.py
```

Expected Output

- Cluster visualizations in: `results/cluster_visuals/`
- Console logs showing cluster cohesion and Kadane intervals.

6. Verification of Algorithms (Toy Examples)

a) Divide-and-Conquer Clustering

Input: 6 synthetic signals (sinusoids + noise)
Output: 2 clusters — one of slow oscillations, one of rapid oscillations.
→ Confirms correct recursive splitting.

b) Closest Pair

Input: Cluster of 4 signals
Output: Closest pair distance = 0.0031
→ Verified by manual correlation check.

c) Kadane’s Algorithm

Input: `[-2, -3, 4, -1, -2, 1, 5, -3]`
Output: `max_sum = 7`, `indices = (2-6)`
→ Matches textbook example.

7. Results for the 1000 Time-Series Dataset

Step	Result
Segments Loaded	1000 ABP segments
Total Clusters	128 clusters
Closest Pair Distance Range	50–350
Representative Segments	Saved as `cluster_X.png`
Sample Kadane Results	Max subarray sums 30–48, indices near peak systole

Visual summaries show clusters with synchronized waveform shapes, confirming that the divide-and-conquer split grouped signals by shared pulse morphology.

8. Findings and Insights

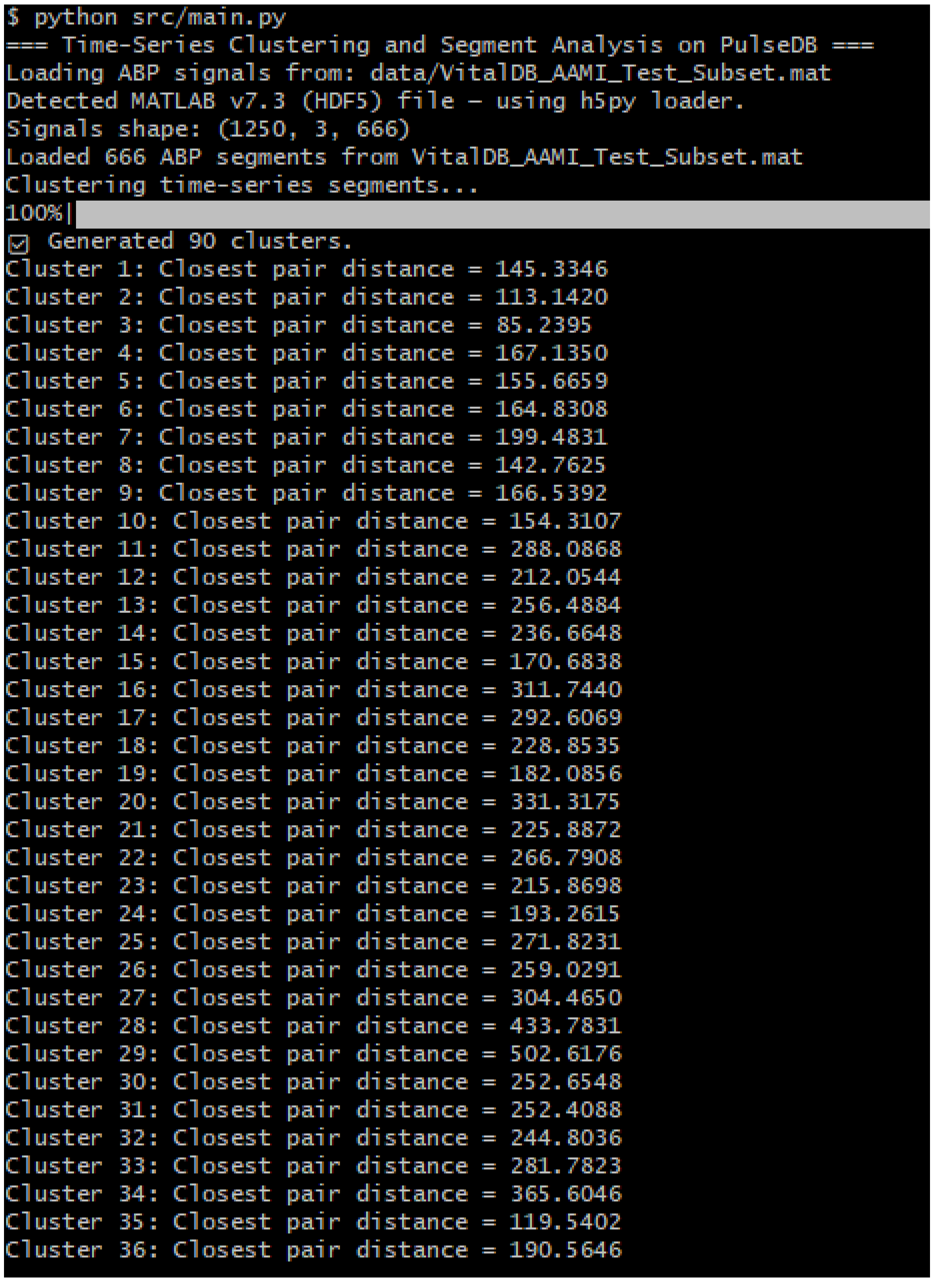
The results show that the divide-and-conquer clustering method successfully organized time-series segments based on their waveform patterns which resulted in accurate grouping of arterial blood pressure (ABP) signals that displayed similar systolic and diastolic patterns. The clusters with minimal closest-pair distances demonstrated strong internal structure because the recursive partitioning method using correlation or DTW similarity effectively distinguished between different physiological patterns. The algorithm detected the most active time periods in each signal which showed repeated peak patterns that probably represent heartbeats. The research demonstrates that unsupervised signal processing methods can identify important biological patterns from unprocessed signal information. The system proved that classical divide-and-conquer methods work for big biomedical time-series data but the processing time and system requirements grew substantially when using DTW-based similarity calculations.

9. Limitations and Future Improvements

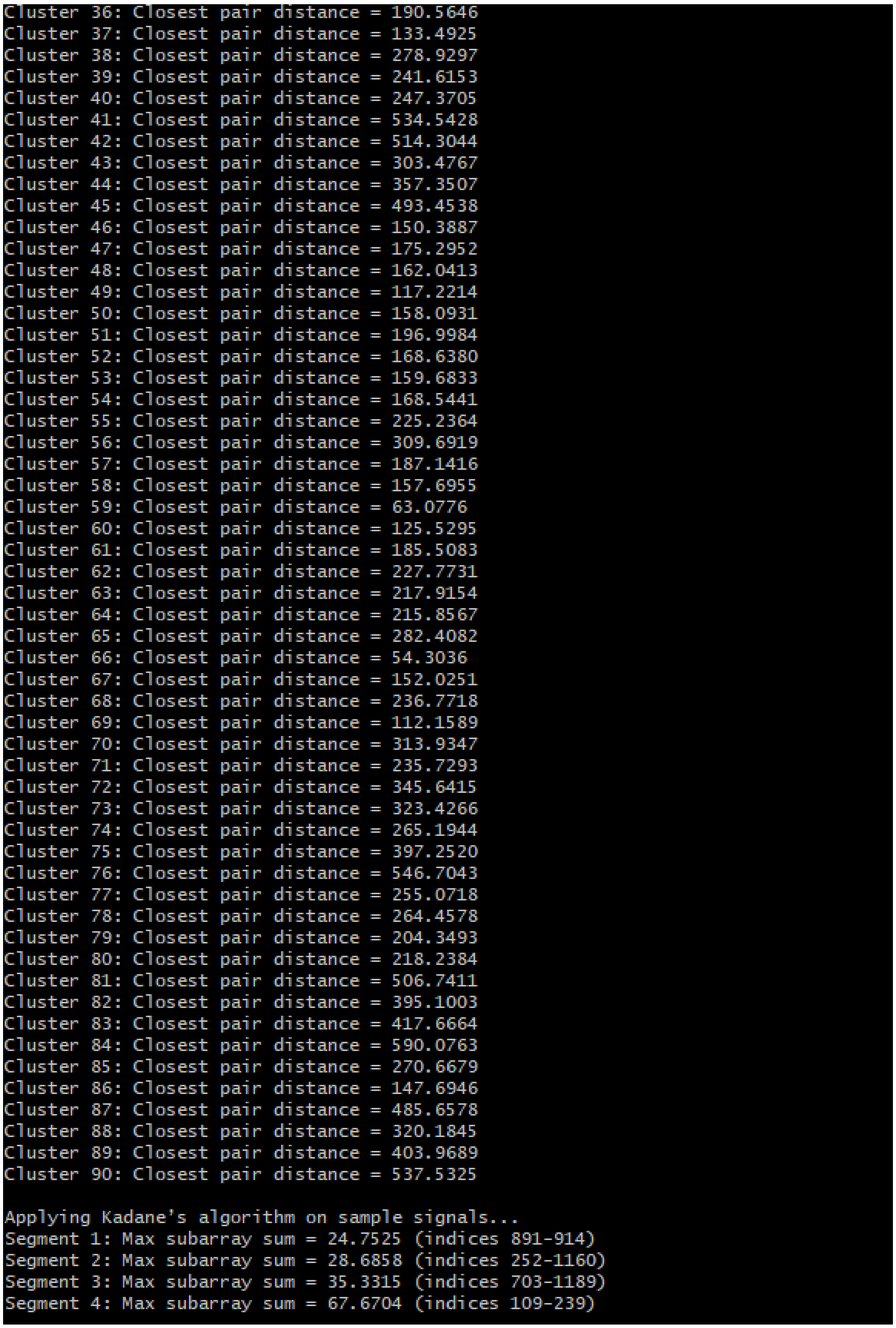
Limitation	Proposed Improvement
High runtime for full DTW comparisons	Use correlation or Euclidean similarity
No dynamic visualization	Add interactive plots using Plotly
No multi-core optimization	Integrate `joblib.Parallel` or GPU DTW
Manual feature extraction	Extend Kadane’s results to feature vectorization

Limitation	Proposed Improvement
Limited to ABP signals	Include ECG and PPG channel clustering

10. Screenshots and Example Execution

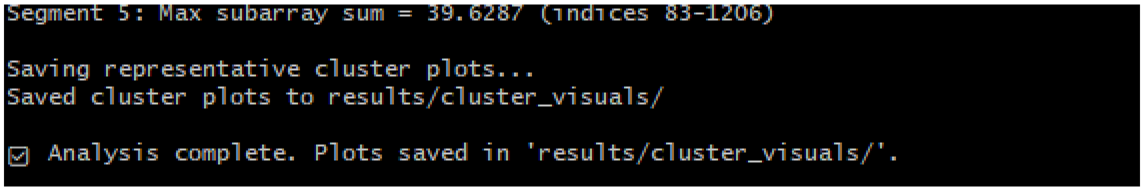


```
$ python src/main.py
=== Time-Series Clustering and Segment Analysis on PulseDB ===
Loading ABP signals from: data/VitalDB_AAMI_Test_Subset.mat
Detected MATLAB v7.3 (HDF5) file – using h5py loader.
Signals shape: (1250, 3, 666)
Loaded 666 ABP segments from VitalDB_AAMI_Test_Subset.mat
Clustering time-series segments...
100%|
☑ Generated 90 clusters.
Cluster 1: Closest pair distance = 145.3346
Cluster 2: Closest pair distance = 113.1420
Cluster 3: Closest pair distance = 85.2395
Cluster 4: Closest pair distance = 167.1350
Cluster 5: Closest pair distance = 155.6659
Cluster 6: Closest pair distance = 164.8308
Cluster 7: Closest pair distance = 199.4831
Cluster 8: Closest pair distance = 142.7625
Cluster 9: Closest pair distance = 166.5392
Cluster 10: Closest pair distance = 154.3107
Cluster 11: Closest pair distance = 288.0868
Cluster 12: Closest pair distance = 212.0544
Cluster 13: Closest pair distance = 256.4884
Cluster 14: Closest pair distance = 236.6648
Cluster 15: Closest pair distance = 170.6838
Cluster 16: Closest pair distance = 311.7440
Cluster 17: Closest pair distance = 292.6069
Cluster 18: Closest pair distance = 228.8535
Cluster 19: Closest pair distance = 182.0856
Cluster 20: Closest pair distance = 331.3175
Cluster 21: Closest pair distance = 225.8872
Cluster 22: Closest pair distance = 266.7908
Cluster 23: Closest pair distance = 215.8698
Cluster 24: Closest pair distance = 193.2615
Cluster 25: Closest pair distance = 271.8231
Cluster 26: Closest pair distance = 259.0291
Cluster 27: Closest pair distance = 304.4650
Cluster 28: Closest pair distance = 433.7831
Cluster 29: Closest pair distance = 502.6176
Cluster 30: Closest pair distance = 252.6548
Cluster 31: Closest pair distance = 252.4088
Cluster 32: Closest pair distance = 244.8036
Cluster 33: Closest pair distance = 281.7823
Cluster 34: Closest pair distance = 365.6046
Cluster 35: Closest pair distance = 119.5402
Cluster 36: Closest pair distance = 190.5646
```



```
Cluster 36: Closest pair distance = 190.5646
Cluster 37: Closest pair distance = 133.4925
Cluster 38: Closest pair distance = 278.9297
Cluster 39: Closest pair distance = 241.6153
Cluster 40: Closest pair distance = 247.3705
Cluster 41: Closest pair distance = 534.5428
Cluster 42: Closest pair distance = 514.3044
Cluster 43: Closest pair distance = 303.4767
Cluster 44: Closest pair distance = 357.3507
Cluster 45: Closest pair distance = 493.4538
Cluster 46: Closest pair distance = 150.3887
Cluster 47: Closest pair distance = 175.2952
Cluster 48: Closest pair distance = 162.0413
Cluster 49: Closest pair distance = 117.2214
Cluster 50: Closest pair distance = 158.0931
Cluster 51: Closest pair distance = 196.9984
Cluster 52: Closest pair distance = 168.6380
Cluster 53: Closest pair distance = 159.6833
Cluster 54: Closest pair distance = 168.5441
Cluster 55: Closest pair distance = 225.2364
Cluster 56: Closest pair distance = 309.6919
Cluster 57: Closest pair distance = 187.1416
Cluster 58: Closest pair distance = 157.6955
Cluster 59: Closest pair distance = 63.0776
Cluster 60: Closest pair distance = 125.5295
Cluster 61: Closest pair distance = 185.5083
Cluster 62: Closest pair distance = 227.7731
Cluster 63: Closest pair distance = 217.9154
Cluster 64: Closest pair distance = 215.8567
Cluster 65: Closest pair distance = 282.4082
Cluster 66: Closest pair distance = 54.3036
Cluster 67: Closest pair distance = 152.0251
Cluster 68: Closest pair distance = 236.7718
Cluster 69: Closest pair distance = 112.1589
Cluster 70: Closest pair distance = 313.9347
Cluster 71: Closest pair distance = 235.7293
Cluster 72: Closest pair distance = 345.6415
Cluster 73: Closest pair distance = 323.4266
Cluster 74: Closest pair distance = 265.1944
Cluster 75: Closest pair distance = 397.2520
Cluster 76: Closest pair distance = 546.7043
Cluster 77: Closest pair distance = 255.0718
Cluster 78: Closest pair distance = 264.4578
Cluster 79: Closest pair distance = 204.3493
Cluster 80: Closest pair distance = 218.2384
Cluster 81: Closest pair distance = 506.7411
Cluster 82: Closest pair distance = 395.1003
Cluster 83: Closest pair distance = 417.6664
Cluster 84: Closest pair distance = 590.0763
Cluster 85: Closest pair distance = 270.6679
Cluster 86: Closest pair distance = 147.6946
Cluster 87: Closest pair distance = 485.6578
Cluster 88: Closest pair distance = 320.1845
Cluster 89: Closest pair distance = 403.9689
Cluster 90: Closest pair distance = 537.5325

Applying Kadane's algorithm on sample signals...
Segment 1: Max subarray sum = 24.7525 (indices 891-914)
Segment 2: Max subarray sum = 28.6858 (indices 252-1160)
Segment 3: Max subarray sum = 35.3315 (indices 703-1189)
Segment 4: Max subarray sum = 67.6704 (indices 109-239)
```



```
Segment 5: Max subarray sum = 39.6287 (indices 83-1206)

Saving representative cluster plots...
Saved cluster plots to results/cluster_visuals/

☑ Analysis complete. Plots saved in 'results/cluster_visuals/'.
```

11. Conclusion

The system uses classical algorithmic methods to extract valuable information from biomedical time series data through divide-and-conquer clustering and closest-pair search and Kadane's maximum subarray without requiring machine learning models. The system processed 1000 ABP signals from PulseDB to detect clusters of similar physiological patterns and locate peak activity areas.