Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/120/2025.18742

Fast Constrained Sparse Dynamic Time Warping

Youngha Hwang

LG Display, DX Group
128 Yeoui-Daero, Yeongdeungpo-gu, Seoul, 07336, Republic of Korea
h.youngha@gmail.com

Abstract: Dynamic Time Warping (DTW) has been proposed to solve machine learning problems when comparing time series. Such time series can occasionally be sparse due to the inclusion of zero-values at many epochs. Since the traditional DTW does not utilize the sparsity of time series data, various fast algorithms equivalent to DTW were developed: (1) Sparse Dynamic Warping (SDTW); (2) Constrained Sparse Dynamic Time Warping (CSDTW) with the constraint on the warping path; (3) Fast Sparse Dynamic Time Warping (FSDTW) as a fast approximate algorithm of SDTW. This paper develops and analyzes a fast algorithm that approximates CSDTW, Fast Constrained Sparse Dynamic Time Warping (FCSDTW). FCSDTW significantly decreases the computational complexity compared to constrained DTW (CDTW) and also shows speed improvement against CSDTW with negligible errors. This study should add to a framework in sparsity exploitation for reducing complexity.

Keywords: dynamic time warping, time series, sparsity ratio, global constraint, sparse dynamic time warping

1. Introduction

After the first introduction of Dynamic Time Warping (DTW) in speech recognition [1], machine learning has been using it for its various problems [2–4]. Several review articles [5, 6] highlighted the potential of DTW for time series comparison. However, there was a main barrier to using DTW, computational complexity. The time complexity, $O(NM)$, provides an approximation of the work required for an algorithm based on the input sizes, N and M. Numerous algorithmic strategies to reduce the complexity of DTW have been published however they didn't solve DTW problems exactly [7, 8].

One of the real-life cases in sparse time series is managing zero values such as periods of silence in speech, inactivity or lack of food intake [9], or periods when house appliances are not running [10]. They are not missing data but explain the sparse characteristics of the source. To mitigate the challenges when using DTW for such sources, Sparse Dynamic Time Warping (SDTW) [11], Binary Sparse Dynamic Time Warping (BSDTW) [12], Constrained Sparse Dynamic Time Warping (CSDTW) [7], and Constrained Binary sparse Dynamic Time Warping (CBSDTW) [13] were developed and analyzed to speed up the DTW. The new algorithms were equivalent to DTW but had their computational complexities reduced by the sparsity ratio, s or s^2 (BSDTW, CBSDTW), where s was defined

DOI: 10.54254/2755-2721/120/2025.18742

as the arithmetic mean of the proportion of non-zero values in the data. Also, Fast Sparse Dynamic Warping (FSDTW) [8] and AWarp [14] were developed and analyzed for further speed improvement by the order of s^2 at the expense of accuracy.

The current study focuses on the faster approximation of CSDTW, Fast Constrained Sparse Dynamic Time Warping (FCSDTW). The time complexity of FCSDTW is upper bounded by about $6s^2$ times the complexity of CDTW. Even though FCSDTW is slightly slower than constrained AWarp (CAWarp) when comparing worst cases, a wide range of examples of FCSDTW is needed to provide better complexity and a more accurate estimation of CDTW. The relative error of FCSDTW (relative to CDTW) is typically $\sim 10^{-4}$, and is at least ten-thousandths that of CAWarp. Providing the most rigorous analysis, this study should be a core background for further reduction of time complexity.

The review of DTW from the perspective of dynamic programming is not included. The interested reader should refer to the previous publications [7, 8]. Section 2 presents the principles of FCSDTW and its reduction of complexity compared to CDTW. Section 3 has a description of the FCSDTW algorithm. Finally, the last section demonstrates the efficiency of FCSDTW with some numerical analysis.

2. Principles Underlying FCSDTW

This section starts with some notation and review of CSDTW [7]. Then, the FCSDTW algorithm is evident from FSDTW [8] properties and its complexity can be derived. However, explicit description of the algorithm requires some effort, so this is done in Section 3..

Let us have two time series such as $X = (x_1, x_2, \cdots, x_M)$ and $Y = (y_1, y_2, \cdots, y_N)$, and their subsequences of non-zero values are represented as $X_s = \left(x_{m_1}, x_{m_2}, \cdots, x_{m_{M_s}}\right)$ and $Y_s = \left(y_{n_1}, y_{n_2}, \cdots, y_{n_{N_s}}\right)$, respectively. We have developed a faster approximation of the (i.e., FCSDTW) algorithm with X_s and Y_s that approximates CDTW in terms of the distance between X and Y. The same notions [7, 8] were used when describing the geometric objects in the $m - n$ plane for the time indices of the non-zero data. A rectangle $R_{i,j}$ is made of four vertices from the consecutive nonzeros $m_{i-1}, m_i, n_{j-1}, n_j$. We have the following accumulated distance definition from [7]: $g(m, n) = \min\left[g(m, n-1) + d(m, n)w_v, g(m-1, n-1) + d(m, n)w_d, g(m-1, n) + d(m, n)w_h\right]$, where $d(m, n)$ is the local distance.

As a brief review, SDTW calculates the accumulated distances along the top and right edges of the rectangle $R_{i,j}$. For SDTW, computation at the lower left end of the interior, $(m_{i-1} + 1, n_{j-1} + 1)$ is enough because the accumulated distances at the interior of $R_{i,j}$ are the same. For CSDTW, on the other hand, the feasible set F is involved in its calculation. [Note that CDTW calculates the accumulated distances on F.] FCSDTW computes the distances only at the end points of $E_t \cap F$ and $E_r \cap F$ because the accumulated distance calculation is sufficient at their endpoints. For the property of FCSDTW, the definition of the intersections between the edges and the feasible set is needed.

Remark 1. The intersections between the edges (E_t and E_r) and the feasible region F and their boundaries (endpoints) are listed.
$E_t \cap F = \{(m, n_j) | \max(m_{i-1} + 1, n_j - L) \le m \le \min(m_i - 1, n_j + L)\}$.
$E_r \cap F = \{(m_i, n) | \max(n_{j-1} + 1, m_i - L) \le n \le \min(n_j - 1, m_i + L)\}$.
$\partial\{E_t \cap F\} = \{(\max(m_{i-1} + 1, n_j - L), n_j), (\min(m_i - 1, n_j + L), n_j)\}$.
$\partial\{E_r \cap F\} = \{(m_i, \max(n_{j-1} + 1, m_i - L)), (m_i, \min(n_j - 1, m_i + L))\}$.

Proof. The proof is shown in [7]. Their boundaries are given from the minimum and maximum. □

DOI: 10.54254/2755-2721/120/2025.18742

For FCSDTW, we have only to compute $g(\partial\{E_t \cap F\})$ and $g(\partial\{E_r \cap F\})$ instead of $g(E_t \cap F)$ and $g(E_r \cap F)$, respectively because no accumulated distances are necessary on the points between the end points. Note that we assume that the local distances are infinite outside the feasible set F, i.e., $d(F^c) = \infty$. Let us first look at the case when when $R^{\circ}_{i,j} \cap F = \emptyset$.

Proposition 1. *Let us denote* $\partial\{E_t \cap F\} = \{(t_1, n_j), (t_2, n_j)\}$, *where* $t_1 = \max(m_{i-1}+1, n_j - L)$ *and* $t_2 = \min(m_i - 1, n_j + L), n_j)$ *Similarly,* $\partial\{E_r \cap F\} = \{(m_i, r_1), (m_i, r_2)\}$, *where* $r_1 = \max(n_{j-1} + 1, m_i - L)$ *and* $r_2 = \min(n_j - 1, m_i + L)$ *Then,* $g(\partial\{E_t \cap F\} = \{g(t_1, n_j), g(t_2, n_j)\})$ *is computed as follows when* $n_j = n_{j-1} + 1$ *holds.* $g(t_1, n_j)$ *can be computed from DTW.*

$$g(t_1, n_j) = \min\{g(t_1, n_{j-1}) + d(0, n_j)w_v, g(t_1 - 1, n_j) + d(0, n_j)w_h, g(t_1 - 1, n_{j-1}) + d(0, n_j)w_d\} \quad (1)$$

if $t_1 > m_{i-1} + 1$,

$$g(t_2, n_j) = \min \begin{cases} g(t_1, n_j) + d(0, n_j)(t_2 - t_1)w_h \\ g(t_2, n_{j-1}) + d(0, n_j)w_v \\ g(t_2 - 1, n_{j-1}) + d(0, n_j)w_d \\ g(t_1 - 1, n_{j-1}) + d(0, n_{j-1})(t_2 - t_1)w_h + d(0, n_j)w_d \end{cases} \quad (2)$$

otherwise, i.e., $t_1 = m_{i-1} + 1$,

$$g(t_2, n_j) = \min \begin{cases} g(t_1, n_j) + d(0, n_j)(t_2 - t_1)w_h \\ g(t_2, n_{j-1}) + d(0, n_j)w_v \\ g(t_2 - 1, n_{j-1}) + d(0, n_j)w_d \\ g(t_1, n_{j-1}) + d(0, n_{j-1})(t_2 - t_1 - 1)w_h + d(0, n_j)w_d \\ g(t_1, n_{j-1}) + d(0, n_j)(t_2 - t_1 - 1)w_h + d(0, n_j)w_d \end{cases} \quad (3)$$

$g(\partial\{E_r \cap F\} = \{g(m_i, r_1), g(m_i, r_2)\})$ *is computed as follows when* $m_i = m_{i-1} + 1$ *holds.* $g(m_i, r_1)$ *can be computed from DTW.*

$$g(m_i, r_1) = \min\{g(m_i, r_1 - 1) + d(m_i, 0)w_v, g(m_{i-1}, r_1) + d(m_i, 0)w_h, g(m_{i-1}, r_1 - 1) + d(m_i, 0)w_d\} \quad (4)$$

if $r_1 > n_{j-1} + 1$,

$$g(m_i, r_2) = \min \begin{cases} g(m_i, r_1) + d(m_i, 0)(r_2 - r_1)w_v \\ g(m_{i-1}, r_2) + d(m_i, 0)w_h \\ g(m_{i-1}, r_2 - 1) + d(m_i, 0)w_d \\ g(m_{i-1}, r_1 - 1) + d(m_{i-1}, 0)(r_2 - r_1)w_v + d(m_i, 0)w_d \end{cases} \quad (5)$$

otherwise, i.e., $r_1 = n_{j-1} + 1$,

$$g(m_i, r_2) = \min \begin{cases} g(m_i, r_1) + d(m_i, 0)(r_2 - r_1)w_v \\ g(m_{i-1}, r_2) + d(m_i, 0)w_h \\ g(m_{i-1}, r_2 - 1) + d(m_i, 0)w_d \\ g(m_{i-1}, r_1) + d(m_{i-1}, 0)(r_2 - r_1 - 1)w_v + d(m_i, 0)w_d \\ g(m_{i-1}, r_1) + d(m_i, 0)(r_2 - r_1 - 1)w_v + d(m_i, 0)w_d \end{cases} \quad (6)$$

DOI: 10.54254/2755-2721/120/2025.18742

Proof. Let us first look into $g(m_i, r_2)$ when $m_i = m_{i-1} + 1$ holds. We know that $g(m_i, r_2)$ always depend on $g(m_i, r_1)$. Computation of $g(m_i, r_2)$ depends on whether $r_1 = n_{j-1} + 1$ holds and $r_2 = n_j - 1$ holds. If $r_1 = n_{j-1} + 1$ holds, then $g(m_i, r_2)$ depends on $g(m_{i-1}, r_1)$ like Eq. 8 of the unconstrained FSDTW [8]. Otherwise, It depends on $g(m_{i-1}, r_1 - 1)$ and it does not have the term $g(m_{i-1}, r_1 - 1) + d(m_i, 0)w_d + \cdots$ like Eq. 6 because $g(m_i, r_1) \leq g(m_{i-1}, r_1 - 1) + d(m_i, 0)w_d$ from its definition. Similarly, $g(m_i, r_2)$ depends on $g(m_{i-1}, r_2)$ if $r_2 = n_j - 1$ holds. Otherwise, It depends on $g(m_{i-1}, r_2 - 1)$. For the former case, we have $g(m_{i-1}, r_2 - 1) = \infty$ and vice versa for the other case. Thus we can add both terms simultaneously and compute $g(m_i, r_2)$ without checking if $r_2 = r_m - 1$ or not. $g(t_2, n_j)$ can be derived similarly when $n_j = n_{j-1} + 1$ holds. □

Remark 2. Geometry of the feasible set F leads to the following result on $g(R^\circ_{i,j} \cap F)$.

$$g(R^\circ_{i,j} \cap F) = \begin{cases} g(n_{j-1} - L, n_{j-1}) & \text{if } m_{i-1} + L < n_{j-1}, \\ g(m_{i-1}, m_{i-1} - L) & \text{if } m_{i-1} - L > n_{j-1}, \\ g(m_{i-1} + 1, n_{j-1} + 1) & \text{else.} \end{cases} \tag{7}$$

Given the above remark, let us look at the case where $R^\circ_{i,j} \cap F \neq \emptyset$.

Proposition 2. *$g(\partial\{E_t \cap F\})$ is equal to $g(R^\circ_{i,j} \cap F) + d(0, n_j) \min\{w_v, w_d\}$ when $(m_{i-1}+1, n_j) \notin F$. Otherwise, i.e., $t_1 = m_{i-1} + 1$ holds, it is computed as follows:*

$$\begin{aligned} &g(t_1, n_j) \\ &= \min\{g(R^\circ_{i,j} \cap F) + d(0, n_j)w_v, g(m_{i-1}, n_j) + d(0, n_j)w_h, g(m_{i-1}, n_j - 1) + d(0, n_j)w_d\} \end{aligned} \tag{8}$$

$$g(t_2, n_j) = \min\{g(R^\circ_{i,j} \cap F) + d(0, n_j) \min\{w_d, w_v\}, g(t_1, n_j) + d(0, n_j)(t_2 - t_1)w_h\} \tag{9}$$

$g(\partial\{E_r \cap F\})$ is equal to $g(R^\circ_{i,j} \cap F) + d(m_i, 0) \min\{w_h, w_d\}$ when $(m_i, n_{j-1}+1) \notin F$. Otherwise, i.e., $r_1 = n_{j-1} + 1$ holds, it is computed as follows:

$$\begin{aligned} &g(m_i, r_1) \\ &= \min\{g(R^\circ_{i,j} \cap F) + d(m_i, 0)w_h, g(m_i, n_{j-1}) + d(m_i, 0)w_v, g(m_i - 1, n_{j-1}) + d(m_i, 0)w_d\} \end{aligned} \tag{10}$$

$$g(m_i, r_2) = \min\{g(R^\circ_{i,j} \cap F) + d(m_i, 0) \min\{w_d, w_h\}, g(m_i, r_1) + d(m_i, 0)(r_2 - r_1)w_v\} \tag{11}$$

Proof. Let us look at $g(\partial\{E_t \cap F\})$ first. When $(m_{i-1} + 1, n_j) \notin F$, $t_1 = n_j - L > n_{i-1} + 1$. So $g(t_1, n_j) = g(t_2, n_j) = g(R^\circ_{i,j} \cap F) + d(0, n_j) \min\{w_v, w_d\}$ holds. Otherwise, $t_1 = m_{i-1} + 1$ holds and we compute $g(t_1, n_j)$ from DTW. $g(t_2, n_j)$ can be computed from either $g(R^\circ_{i,j} \cap F)$ or $g(t_1, n_j)$ like Eq. 5 of the unconstrained FSDTW [8]. So Eq. 9 follows. $g(\partial\{E_r \cap F\})$ can be computed similarly. □

3. FCSDTW Algorithm and Complexity

The results in Section 2. show that it is possible to compute the DTW distance by scanning through the intersections of the feasible set and the rectangles $R_{i,j}$ row or column-wise, and only computing the accumulated distances at six points, namely the end points of the top edge, those of the right edge, the upper right vertex and a single interior point (if the interior is not empty). Indeed, the Propositions and the subsequent Remarks show these accumulated distances can be expressed in terms of the

DOI: 10.54254/2755-2721/120/2025.18742

Algorithm 1 FCSDTW Algorithm

Append a dummy one if necessary to get time series X and Y which end in a one. Build X_s and Y_s of lengths M_s and N_s, respectively, from X and Y of lengths M and N, respectively, by deleting the zeros.
for $j = 1 : N_s$ **do**

  for $i = 1 : M_s$ **do**

    if $R^{\circ}_{ij} \neq \emptyset$ **then**
      Compute $g(R^{\circ}_{i,j} \cap F)$ from Eq. 7.
      if $(m_{i-1} + 1, n_j) \notin F$ **then**
        Compute $g(\partial\{E_t \cap F\})$ from $g(R^{\circ}_{i,j} \cap F) + d(0, n_j) \min\{w_v, w_d\}$
      else
        Compute $g(\partial\{E_t \cap F\})$ from Eqs. 8, 9.
      end if
      if $(m_i, n_{j-1} + 1) \notin F$ **then**
        Compute $g(\partial\{E_r \cap F\})$ from $g(R^{\circ}_{i,j} \cap F) + d(m_i, 0) \min\{w_h, w_d\}$
      else
        Compute $g(\partial\{E_r \cap F\})$ from Eqs. 10, 11.
      end if
      Compute $g((m_i, n_j) \cap F)$ in the same way as DTW
    else

      if $m_i = m_{i-1} + 1$ **then**
        Compute $g(\partial\{E_r \cap F\})$ from Eqs. 4, 5, 6.
        Compute $g((m_i, n_j) \cap F)$ in the same way as DTW
      else if $n_j = n_{j-1} + 1$ **then**
        Compute $g(\partial\{E_t \cap F\})$ from Eqs. 1, 2, 3.
        Compute $g((m_i, n_j) \cap F)$ in the same way as DTW
      end if
    end if
  end for
end for
Compute the DTW distance as $D(X, Y) = \frac{1}{K^*} h(M^*, N^*)$, where $M^* = M$ if a one is appended to get X and $M^* = M - 1$ otherwise, and $N^* = N$ if a one is appended to get Y and $N^* = N - 1$ otherwise.

DOI: 10.54254/2755-2721/120/2025.18742

corresponding accumulated distances at previously scanned rectangles. The boundary condition of the algorithm was discussed in [7, 8]. Formally the FCSDTW algorithm can be written in Algorithm 1.

Bound on the time complexity C_{FCSDTW} of FCSDTW was computed by counting the necessary computation of the accumulated distances. It is proportional to that of FSDTW [8]. Its proof is omitted.

Proposition 3. $C_{FSDTW} \leq \frac{2L}{N}\{6(N_s+1)(M_s+1) - 2N - 1\}$ *on average.*

Remark 3. The expression of the bound is $C_{FCSDTW} \leq \frac{2L}{N}\{6s_1s_2N^2 + 12sN - 2N + 5\}$ and further upper bounded by $C_{FCSDTW} \leq \frac{2L}{N}\{6s^2N^2 + 12sN - 2N + 5\}$, where as above $s_1 = \frac{N_s}{N}$, $s_2 = \frac{M_s}{N}$ and $s = \frac{1}{2}(s_1 + s_2)$, the sparsity ratio. Note that $C_{FCSDTW} = O(s^2N^2)$ as $N \to \infty$. The complexity of CDTW is $2LN$ Thus, the complexity of FCSDTW over CDTW is calculated as $6s^2 + 12s/N - 2/N + 5/N^2$.

4. Numerical Analysis

This section shows efficiency comparisons using synthetic and experimental data between FCSDTW and CDTW and between FCSDTW with CAWarp and CSDTW. Other research has shown a significant reduction in time complexity but a large increase in errors [7, 11]. For FCSDTW we have the same hyperparameters as CSDTW [7] with the same computer but different versions of software such as MATLAB R2018b, and OS X 10.11.6. Following the same method in [7], the UCR collection was used to generate sparse time series data [15], specifically ECG, Trace, Car, MALLAT, and CinC ECG torso. Their abbreviations were ECG, TR, Car, ML, and CET, respectively.

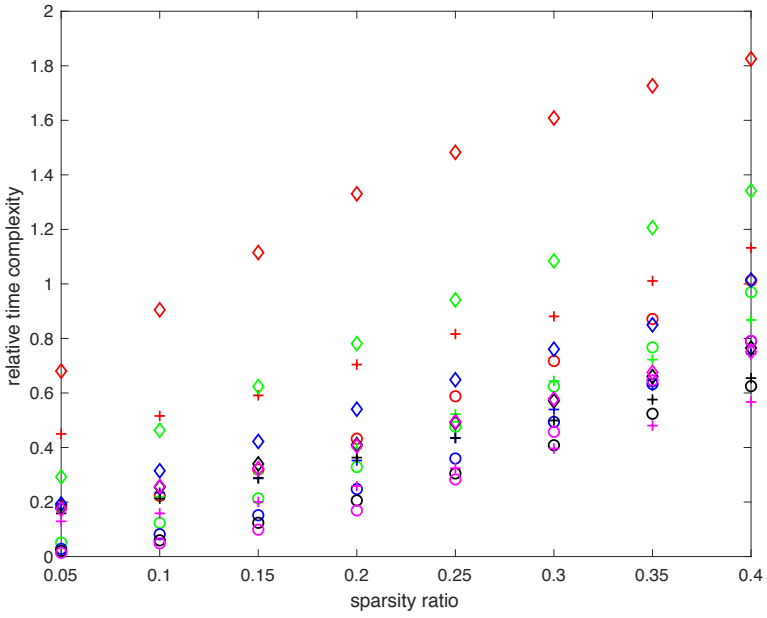

Figure 1: Relative time complexity of CSDTW, FCSDTW, and CAWarp against CDTW are shown by diamonds, crosshairs, and circles, respectively. The colors red, green, blue, black, and magenta denote the data ECG, TR, Car, ML, and CET, respectively.

DOI: 10.54254/2755-2721/120/2025.18742

Table 1: Regression coefficients for relative time complexity of FCSDTW over CDTW versus sparsity ratio. b_2 and b_0 represent the coefficients of s^2 and the constant.

data set	length	b_2	b_0
ECG	96	4.290	0.496
TR	275	4.332	0.205
Car	577	3.463	0.205
ML	1024	3.036	0.205
CET	1639	2.729	0.140

Table 2: Relative errors of the cost of FCSDTW over CDTW with varying sparsity ratio in 10^{-4}. The symbol 'eps' stands for errors less than double precision accuracy in MATLAB.

data set	sparsity ratio							
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
ECG	eps	eps	eps	0.0017	0.1039	0.0172	0.1733	1.2845
TR	eps	eps	eps	0.0769	0.0940	0.0789	0.0345	0.0474
Car	eps	eps	0.0673	0.0586	0.2367	0.2328	0.7437	0.3674
ML	eps	0.0250	0.0464	0.0364	0.1336	0.5085	0.1859	0.3958
CET	eps	eps	0.0023	0.0293	0.0384	0.0710	0.1065	0.2563

Figure 1 demonstrates the relative time complexity - the ratio of the computing times - of CSDTW, FCSDTW, and CAWarp over CDTW. In longer data records, the speed of CSDTW, FCSDTW, and CAWarp increases (see Table 1), presumably from the lower overhead when the algorithm was initialized. A similar trend was observed with the SDTW, BSDTW, and FSDTW [8, 11, 12]. The estimated time complexity of CAWarp lies between 0.3 and 1.3 against FCSDTW in Figure 1. Their theoretical ratio was 2/3.

Table 1 presents a second-order polynomial regression analysis comparing the relative time complexity of algorithm FCSDTW to CDTW with the sparsity ratio from the sparse time series datasets. The obtained regression is $b_2 s^2 + b_0$. We have the values of b_2 between two and four, satisfying the upper bound six in the Remark following Proposition 3. However, We have the values of b_0 much higher than FSDTW, probably because of the computing load to find the region $R_{i,j} \cap F$. Such examples are displayed in [7]. Note that s term is not negligible for ECG time series because its length N is only 96. So b_0 is estimated around 0.5. In contrast, b_0 values are around 0.14 for CET because their lengths are larger than 1600. Note that the values of b_0 of CAWarp are between 0.02 and 0.2, which are between two times and six times lower than FCSDTW. Such differences in the values of b_0 result in the time complexity differences between FCSDTW and CAWarp in Figure 1.

Table 2 shows the relative errors of the cost by comparing FCSDTW to CDTW in ten-thousandth (10^{-4}) scale. The relative errors of the cost of CAWarp are between 10^4 and 10^5 times those of FCSDTW in Table 3. Therefore when considering FCSDTW's small errors, the speed of FCSDTW is more significant compared to CAWarp. In addition, FCSDTW doesn't have errors like FSDTW when the low sparsity ratio is 0.05 or 0.1 because they nearly have no consecutive non-zeros, the cause of errors.

Table 4 provides a comparison of the time complexities in CAWarp, FCSDTW, and CSDTW against that of CDTW for washing machine power usage [10] to demonstrate the better performance of FCSDTW in real datasets. The dataset was downsampled by 40 for fast processing and its length

DOI: 10.54254/2755-2721/120/2025.18742

Table 3: Relative errors of the cost of CAWarp over CDTW cost with varying sparsity ratio.

data set	sparsity ratio							
	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
ECG	0.3053	0.2553	0.2363	0.1898	0.1681	0.1608	0.1634	0.1494
TR	0.3998	0.3534	0.3273	0.2938	0.2878	0.2673	0.2470	0.2467
Car	0.2476	0.1664	0.1671	0.1342	0.1476	0.1407	0.1497	0.1312
ML	0.2004	0.1402	0.1092	0.1018	0.0910	0.0810	0.0689	0.0714
CET	0.3627	0.3101	0.2927	0.2851	0.2089	0.2300	0.2126	0.2065

was reduced to 270 (from 10800). Although CAWarp is faster, but its error of the cost is much bigger than that of FCSDTW: The relative errors in class 3 are 717.64 for CAWarp and 2.2052 for FCSDTW; those in class 7 are 482.85 for CAWarp and 0.4922 for FCSDTW. Thus application of CAWarp to real data can be impractical because its relative error amounts to more than 7% of the CDTW cost.

Table 4: Comparison of the relative time complexities of CAWarp, CSDTW, and FCSDTW against CDTW

class	3	7
sparsity ratio	0.0578	0.02697
CSDTW	0.2204	0.1672
FCSDTW	0.1426	0.1362
CAWarp	0.0350	0.0194

5. Conclusion

This study developed and analyzed FCSDTW to compare sparse time series with zero data in many research applications. FCSDTW closely approximated CDTW with much lower time complexities for time series with many zero values and especially large lengths. FCSDTW is a distinct algorithm from other approaches producing huge complexity reduction with significant errors or no change in complexity without error. Further numerical experiments support the efficiency of FCSDTW compared against CSDTW and CAWarp.

References

[1] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[2] Claus Bahlmann and Hans Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):299–310, 2004.

[3] Zsolt Miklos Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

[4] Samsu Sempena, Nur Ulfa Maulidevi, and Peb Ruswono Aryan. Human action recognition using dynamic time warping. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–5. IEEE, 2011.

DOI: 10.54254/2755-2721/120/2025.18742

[5] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[6] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[7] Youngha Hwang and Saul B Gelfand. Constrained sparse dynamic time warping. In *International Conference on Machine Learning and Applications*, pages 216–222. IEEE, 2018.

[8] Youngha Hwang and Saul B Gelfand. Fast sparse dynamic time warping. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3872–3877. IEEE, 2022.

[9] Heather A Eicher-Miller, Saul Gelfand, Youngha Hwang, Edward Delp, Anindya Bhadra, and Jiaqi Guo. Distance metrics optimized for clustering temporal dietary patterning among us adults. *Appetite*, 144:104451, 2020.

[10] David Murray and L Stankovic. Refit: electrical load measurements. *URL= http://www.refitsmarthomes.org/*, 2017.

[11] Youngha Hwang and Saul B Gelfand. Sparse dynamic time warping. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 163–175. Springer, 2017.

[12] Youngha Hwang and Saul B Gelfand. Binary sparse dynamic time warping. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 748–759. Springer, 2019.

[13] Youngha Hwang. Constrained binary sparse dynamic time warping. In *manuscript*.

[14] Abdullah Mueen, Nikan Chavoshi, Noor Abu-El-Rub, Hossein Hamooni, Amanda Minnich, and Jonathan MacCarthy. Speeding up dynamic time warping distance for sparse time series data. *Knowledge and Information Systems*, 54(1):237–263, 2018.

[15] Eamonn Keogh, Xiaopeng Xi, Li Wei, and Chotirat Ann Ratanamahatana. The ucr time series classification/clustering homepage. *URL= http://www.cs.ucr.edu/˜ eamonn/time_series_data*, 2006.