Proceedings of CONF-SPML 2026 Symposium: The 2nd Neural Computing and Applications Workshop 2025
DOI: 10.54254/2755-2721/2026.TJ30883

Constrained Binary Sparse Dynamic Time Warping

Youngha Hwang

LG Display, AX Group, Seoul, Republic of Korea
h.youngha@gmail.com

Abstract. Dynamic Time Warping (DTW) is employed to a great extent for comparing time series in machine learning, but is computationally expensive, especially with sparse data containing many zeros. To address this, faster DTW variants have been developed, including Sparse DTW (SDTW), Constrained Sparse DTW (CSDTW), and Binary Sparse DTW (BSDTW). This paper presents Constrained Binary Sparse DTW (CBSDTW), which adds a warping path constraint to BSDTW and significantly reduces computational complexity compared to Constrained DTW (CDTW), offering an efficient way to leverage sparsity in time series analysis.

Keywords: Dynamic Time Warping, Sparse Time Series, Binary Data, Constrained Warping Path

1. Introduction

Dynamic Time Warping (DTW), first introduced for speech recognition [1], is now widely used in machine learning for comparing time series [2–4]. Despite its effectiveness, DTW's high computational cost – $O(NM)$, based on the lengths of two time series – limits its use. Although faster versions have been proposed, most provide only approximate results [5, 6].

Many real-world time series are sparse, containing long periods of zero values, such as silence in speech, inactivity, or non-operating devices [7, 8]. These zeros represent real conditions, not missing data. Sparse and binary time series also appear in applications such as anomaly detection and event monitoring (e.g., rainfall, geyser eruptions).

To address these challenges, several efficient variants have been developed, including Sparse DTW (SDTW), Binary Sparse DTW (BSDTW), and Constrained Sparse DTW (CSDTW) [5, 6, 9]. These methods maintain DTW's accuracy while reducing computation by the sparsity ratio s (or s^2 for BSDTW). Further improvements such as Fast Sparse DTW(FSDTW), Fast Constrained Sparse DTW (FCSDTW) [10, 11], and AWarp [12] achieve additional speed gains with minor accuracy trade-offs.

This study introduces Constrained Binary Sparse Dynamic Time Warping (CBSDTW), an extension of BSDTW that adds a warping path constraint. Its time complexity is about $6s^2$ times that of CDTW. While slightly slower than CAWarp, CBSDTW preserves DTW's full accuracy. This work offers a strong basis for future reductions in computational cost. Subsequent sections outline CBSDTW's principles and algorithm, and present numerical results demonstrating its efficiency.

DOI: 10.54254/2755-2721/2026.TJ30883

2. Principles underlying CBSDTW

This section begins from notations and reviews CSDTW [6]. In addition, the CBSDTW algorithm and its complexity is derived in the same way as FCSDTW [11]. The algorithm is complicated to describe explicitly, so it is done in Section **??**.

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 fast algorithm CBSDTW with X_s and Y_s that is equivalent to CDTW in terms of the distance between X and Y. The same notions [5,6,9] 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 $m_{i-1}, m_i, n_{j-1}, n_j$ and four edges connecting them. We denote the interior $R^\circ_{i,j}$ as the dotted rectangle $R_{i,j}$ without the vertices and edges as shown in Fig. 1 of SDTW [5].

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, its calculation is necessary only inside the feasible set F. Note that CDTW calculates the accumulated distances on F, i.e., the feasible set from the Sakoe-Chiba band that has points within the width L from the diagonal of the warping path [1]. For the property of CBSDTW, the definition of the intersections between the edges and the feasible set is needed from CSDTW [6] and that of those boundaries are defined in FCSDTW [11].

For CBSDTW, 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 satisfying $R^\circ_{i,j} \cap F = \emptyset$.

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,* $g(\partial\{E_t \cap F\} = \{(t_1, n_j), (t_2, n_j)\})$ *is computed as follows when* $n_j = n_{j-1} + 1$ *holds.* $g(t_1, n_j)$ *can be computed from DTW.*

$$g(t_1, n_j) = 1 + \min\{g(t_1, n_{j-1}), g(t_1 - 1, n_j), g(t_1 - 1, n_{j-1})\} \tag{1}$$

$$g(t_2, n_j) = \min \begin{cases} g(t_2, n_{j-1}) + 1 \\ g(t_2 - 1, n_{j-1}) + 1 \\ g(t_1, n_j) + (t_2 - t_1) \end{cases} \tag{2}$$

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

$$g(m_i, r_1) = 1 + \min\{g(m_i, r_1 - 1), g(m_{i-1}, r_1), g(m_{i-1}, r_1 - 1)\} \tag{3}$$

$$g(m_i, r_2) = \min \begin{cases} g(m_{i-1}, r_2) + 1 \\ g(m_{i-1}, r_2 - 1) + 1 \\ g(m_i, r_1) + (r_2 - r_1) \end{cases} \tag{4}$$

DOI: 10.54254/2755-2721/2026.TJ30883

Proof. BSDTW has the following property [9]: $g(m_i, n) = g(m_{i-1}, n)$ or $g(m_i, n) = g(m_{i-1}, n) + 1$ for $n_{j-1} + 1 \leq n \leq n_j - 1$. From the above, we know that if $g(m_{i-1}, n) = g(m_{i-1}, n-1) + 1$, then $g(m_i, n) = g(m_{i-1}, n)$. On the other hand, if $g(m_{i-1}, n) = g(m_{i-1}, n-1)$, then $g(m_i, n) = g(m_{i-1}, n) + 1$. Thus, $g(m_i, r_2) = g(m_i, r_1) + (r_2 - r_1)$ by iterations from the former case. The latter case leads to $g(m_i, r_2) = g(m_{i-1}, r_2) + 1$. and $g(m_i, r_2) = g(m_{i-1}, r_2 - 1) + 1$. $n_i = n_{i-1} + 1$ case can be proven similarly.

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

Geometry of the feasible set F leads to the above result on $g(R^{\circ}_{i,j} \cap F)$. Now, let us look at the case where $R^{\circ}_{i,j} \cap F \neq \emptyset$.

Proposition 2. *$g(\partial\{E_t \cap F\})$ is calculated as $g(R^{\circ}_{i,j} \cap F) + 1$ if $(m_{i-1} + 1, n_j) \notin F$ holds. Otherwise, i.e., $t_1 = m_{i-1} + 1$ holds, $g(\partial\{E_t \cap F\})$ is calculated in the following:*

$$g(m_{i-1} + 1, n_j) = \min \begin{cases} g(R^{\circ}_{i,j} \cap F) + 1 \\ g(m_{i-1}, n_j) + 1 \\ g(m_{i-1}, n_j - 1) + 1 \end{cases} \tag{5}$$

$$g(t_2, n_j) = \min \begin{cases} g(R^{\circ}_{i,j} \cap F) + 1 \\ g(t_1, n_j) + (t_2 - t_1) \end{cases} \tag{6}$$

$g(\partial\{E_r \cap F\})$ can be computed as $g(R^{\circ}_{i,j} \cap F) + 1$ if $(m_i, n_{j-1} + 1) \notin F$ holds. Otherwise, i.e., $r_1 = n_{j-1} + 1$ holds, the following results hold on $\partial\{E_r \cap F\}$.

$$g(m_i, n_{j-1} + 1) = \min \begin{cases} g(R^{\circ}_{i,j} \cap F) + 1 \\ g(m_i, n_{j-1}) + 1 \\ g(m_i - 1, n_{j-1}) + 1 \end{cases} \tag{7}$$

$$g(m_i, r_2) = \min \begin{cases} g(R^{\circ}_{i,j} \cap F) + 1 \\ g(m_i, r_1) + (r_2 - r_1) \end{cases} \tag{8}$$

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 $g(t_1, n_j) = g(t_2, n_j) = g(R^{\circ}_{i,j} \cap F) + 1$ 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 [10]. Thus, Eq. 6 follows. $g(\partial\{E_r \cap F\})$ can be computed similarly.

3. CBSDTW 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 like FCSDTW: $\partial\{E_t \cap F\}$, $\partial\{E_r \cap F\}$, the upper right vertex

DOI: 10.54254/2755-2721/2026.TJ30883

and a single interior point (if the interior is not empty). The boundary condition of the algorithm was discussed in [6, 10, 11]. Formally the CBSDTW algorithm can be written in Algorithm 1.

Bound on the time complexity C_{CBSDTW} of CBSDTW was computed by counting the necessary computation of the accumulated distances. It turns out to be the same as FCSDTW because both computes the distances at six points in each rectangle $R_{i,j}$. So the result can be referred in Proposition 3 of FCSDTW [11].

Algorithm 1 CBSDTW Algorithm

If needed, add a placeholder value so that the time series X and Y both end with a 1. Next, form X_s and Y_s, of lengths M_s and N_s, by removing all zero entries from the original series X and Y, which have lengths M and N, respectively.
for $j = 1 : N_s$ **do**

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

    if $R_{ij}^{\circ} \neq \emptyset$ **then**
      Compute $g(R_{i,j}^{\circ} \cap F)$ from Eq. 5.
      if $(m_{i-1} + 1, n_j) \notin F$ **then**
        Compute $g(\partial\{E_t \cap F\})$ from $g(R_{i,j}^{\circ} \cap F) + 1$.
      else
        Compute $g(\partial\{E_t \cap F\})$ from Eqs. 5, 6.
      end if
      if $(m_i, n_{j-1} + 1) \notin F$ **then**
        Compute $g(\partial\{E_r \cap F\})$ from $g(R_{i,j}^{\circ} \cap F) + 1$.
      else
        Compute $g(\partial\{E_r \cap F\})$ from Eqs. 7, 8.
      end if
      Compute $g((m_i, n_j) \cap F)$ just as DTW
    else

      if $m_i = m_{i-1} + 1$ **then**
        Compute $g(\partial\{E_r \cap F\})$ from Eqs. 3, 4.
        Compute $g((m_i, n_j) \cap F)$ just as DTW

      else if $n_j = n_{j-1} + 1$ **then**
        Compute $g(\partial\{E_t \cap F\})$ from Eqs. 1, 2.
        Compute $g((m_i, n_j) \cap F)$ just as DTW
      end if
    end if
  end for
end for
Calculate the DTW distance using $D(X, Y) = \frac{1}{K^*} h(M^*, N^*)$. Here, M^* equals M if an additional 1 was appended to the sequence X; otherwise, M^* is $M - 1$. Likewise, N^* equals N if a 1 was added to Y, and N^* is $N - 1$ if not.

4. Numerical analysis

This section presents numerical results comparing the efficiency of CBSDTW with CDTW, as well as comparisons of CBSDTW with constrained AWarp (CAWarp) and CSDTW. The experiments use both synthetic and real-world data. The width L of the Sakoe-Chiba band, measured from the diagonal of the constraint region, is set to 10% of the time series length - a commonly used value [1]. The weights w_h, w_v, and w_d are all set to one, and the Euclidean norm is used to compute local distances. The experiments were conducted on a computer with an Intel Core i-5 1.3 GHz processor, 8 GB of memory, MATLAB R2018b, running macOS 10.11.6.

DOI: 10.54254/2755-2721/2026.TJ30883

Table 1: Ratio of time complexity of CBSDTW over CDTW under differing sparsity conditions

dataset	sparsity ratio					
	0.05	0.10	0.15	0.20	0.25	0.30
SC	0.686	0.748	0.809	0.943	1.014	1.123
ECG	0.390	0.382	0.471	0.529	0.622	0.709
50W	0.133	0.181	0.244	0.322	0.416	0.517
Car	0.171	0.205	0.253	0.322	0.393	0.491
ML	0.123	0.149	0.190	0.241	0.297	0.360
CET	0.127	0.156	0.193	0.245	0.303	0.371

Table 2: Regression coefficients showing the relative time complexity of CBSDTW compared to CDTW as a function of the sparsity ratio.

dataset	length	b_2	b_0
SC	60	4.236	0.722
ECG	96	3.500	0.386
50W	270	3.866	0.153
Car	577	3.343	0.180
ML	1024	2.366	0.135
CET	1639	2.420	0.139

Although there are publicly available sparse time series data [8], it is often preferable to control the sparsity in experimental data. In this study, sparse time series are created by selecting time indices according to a geometric distribution and setting all other values to zero. For the equal-length time series considered here, this distribution has a mean of $\frac{1}{s}$, where s represents the sparsity ratio. The resulting sparse series are then converted into binary form by assigning a value of one to all nonzero entries. The time series datasets are sourced from the UCR collection [13]. The generated sparse series from the UCR database include Synthetic Control, ECG, 50Words, Car, MALLAT, and CinC ECG torso, abbreviated as SC, ECG, 50W, Car, ML, and CET, respectively.

Table 1 presents the average relative time complexity of CBSDTW compared to CDTW, expressed as the ratio of their computation times. The speed advantage of CBSDTW becomes more pronounced for longer data records (see Table 2 for data lengths), likely due to the algorithm's initialization overhead, including the extraction of nonzero data and their corresponding time indices.

Table 2 presents a second-order polynomial regression analysis of the relative time complexity of CBSDTW compared to CDTW as a function of the sparsity ratio for the sparse time series datasets. The regression model is given by $b_2 s^2 + b_0$. The coefficient b_2 ranges between 2 and 5, which is lower than the upper bound of 6 reported in Remark 3 of FCSDTW [11]. Notably, the b_0 values are significantly higher than those for unconstrained BSDTW, likely due to the additional computational effort required to determine the region $R_{i,j} \cap F$. Note that s term does not disappear for SC time series because its length N is only 60. So b_0 is estimated over 0.7. On the other hand, b_0 values are about 0.13 for ML and CET because their lengths are larger than 1000. These facts match the property from Remark 3 that both b_1 and b_0 are proportional to $O(1/N)$ [11]. Table 3 presents the average relative time complexity of CAWarp compared to CDTW across different sparsity conditions. It shows that CAWarp is faster than CBSDTW, surpassing the theoretical value of the time complexity ratio, $\frac{2}{3}$. Note that the relative time complexity of CAWarp is upper-bounded by $4s^2$ and CBSDTW is upper-

DOI: 10.54254/2755-2721/2026.TJ30883

Table 3: Relative time complexity of CAWarp over CDTW under differing sparsity conditions

dataset	sparsity ratio					
	0.05	0.10	0.15	0.20	0.25	0.30
SC	.237	.330	.413	.555	.683	.862
ECG	.144	.197	.289	.404	.554	.647
50W	.0494	.122	.217	.403	.794	.946
Car	.0275	.0773	.150	.245	.349	.484
ML	.0153	.0497	.101	.167	.249	.340
CET	.0142	.0488	.102	.173	.260	.359

Table 4: Relative error of the CAWarp cost compared to the constrained DTW cost under differing sparsity conditions, measured in 10^{-3}. The symbol 'eps' indicates errors smaller than MATLAB's double-precision accuracy.

data set	sparsity ratio					
	0.05	0.10	0.15	0.20	0.25	0.30
SC	eps	eps	eps	eps	eps	eps
ECG	0.7	2.3	3.1	2.8	2.2	2.6
50W	0.336	0.582	0.703	0.713	0.668	0.578
Car	0.426	0.288	0.316	0.192	0.172	0.086
ML	8.439	4.234	2.290	1.045	0.793	0.619
CET	eps	eps	eps	eps	eps	eps

bounded by $6s^2$. However, CBSDTW is faster than CAWarp for a time series dataset such as 50W when the sparsity ratio is 0.25 or 0.30. Table 4 shows the relative errors of the cost of CAWarp against CDTW, which are about thousand times lower than those of non-binary results in CFSDTW [11]. Such drastic smaller error apparently occurred because AWarp is completely equivalent to DTW for binary data in the unconstrained case [9].

To demonstrate the performance of CBSDTW on real-world data, the time complexities of CAWarp, CBSDTW, and CSDTW are compared with that of CDTW using house power data [8]. The readers that are interested in data preparation can refer to Sec. 4 of FCSDTW [11]. Although CAWarp is four to six times faster than CBSDTW, its error relative to CDTW is significant, for example, from 5% up to 9% of CDTW distance. Thus one has to be cautious before applying CAWarp to real data.

5. Conclusion

CBSDTW efficiently exploits binary sparse time series with repeated zeros, offering the same results as CDTW but with lower computational cost. Its complexity scales with the square of the sparsity ratio, leading to substantial speed gains for sufficiently sparse and long sequences. Experiments on synthetic and real data confirm that CBSDTW outperforms CSDTW, providing a foundation for future integration with other complexity reduction methods.

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.

DOI: 10.54254/2755-2721/2026.TJ30883

[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.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] 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.

[11] Youngha Hwang. Fast constrained sparse dynamic time warping. *Applied and Computational Engineering*, 120(1):50–58, 2025.

[12] 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.

[13] 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.