Efficient Algorithms for Numeric Data Compression

Yen-Yu Chen
Dept. Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan

Chang-Biau Yang
Dept. Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan
correspondence: cbyang@cse.nsysu.edu.tw

Abstract

In this paper, we define a new problem called the numeric data compression (*NDC*) *problem. Given an integer-valued numeric sequence* $S = \langle s_1, s_2, \ldots, s_n \rangle$*, the NDC problem seeks to find a gap constant* d *and a numeric sequence* $C = \langle c_1, c_2, \ldots, c_n \rangle$ *such that the cumulative error between* S *and* C *is minimized. Here, the difference between every two consecutive elements* $c_{i+1} - c_i$*, for* $1 \leq i \leq n-1$*, is restricted to be either* d *or* $-d$*. Then,* C *can be fully represented by* d*,* c_1 *and a fluctuation Boolean sequence* $V = \langle v_1, v_2, \ldots, v_{n-1} \rangle$*, where* $v_i = 1$ *if* $c_{i+1} - c_i = d$*, and* $v_i = -1$ *if* $c_{i+1} - c_i = -d$*. This representation significantly reduces the storage required for* S*, thereby achieving the goal of lossy data compression. We introduce four versions of the NDC problem and develop corresponding algorithms to solve them. The time complexity of our proposed algorithm for solving the third version is* $O(\min(n^3, rn))$*, where* $r = S_{\max} - S_{\min}$*, and* $S_{\max}$ *and* $S_{\min}$ *are the largest and smallest elements in* S*, respectively.*

Keywords: numeric sequence, lossy compression, dynamic programming, delta-compression

I. Introduction

Data compression techniques can be categorized into *lossless* and *lossy* methods, designed for different applications and data characteristics. *Lossless compression* preserves the original data, ensuring that decompressed data is identical to the original. This is critical in applications such as document storage, source codes, and financial computations. Notable lossless methods include Huffman coding [8], Lempel-Ziv encoding (LZ77) [13] and its derivatives, such as LZ78 [14] and LZW [12]. The sliding window concept in LZ77 has also been applied to streaming compression [7].

Lossy compression discards less significant information to achieve higher compression rates. The *discrete cosine transform* (DCT) [9], introduced in 1984, became the foundation of the JPEG standard [11] for static image compression. DCT removes high-frequency components while retaining low-frequency signals, aligning with human visual perception. MPEG standards [6], including H.264 and H.265, extend compression to dynamic images through motion estimation and vector encoding, which are essential for video streaming and surveillance. The *discrete wavelet transform* (DWT) [2], proposed in 1989, powers the JPEG 2000 standard [10], enabling efficient multi-level image compression while preserving high resolution and supporting progressive rendering.

Traditional compression methods reduce redundancy by eliminating repetitive elements but often fail to exploit sequential relationships in data. In real-world sequences, such as daily stock prices, consecutive points exhibit strong correlations that these methods do not fully utilize, limiting their compression potential. To address this, Elias introduced *predictive coding* in 1955 [3], [4], which leverages past data to predict future values and stores only the differences, effectively reducing redundancy. Building on this concept, Cleary and Witten developed *prediction by partial matching* (PPM) in 1984 [1], refining probability estimation with an order-m Markov model for arithmetic coding. This approach enhances compression efficiency by slowing the growth of computational precision requirements as data volume increases, achieving a compression ratio of approximately 2.2 bits per character. The fixed-step delta-compression algorithm, proposed by Vadim Engelson *et al.* in 2000 [5], is specifically designed for compressing large-scale numerical data from scientific simulations.

This paper introduces a lossy variant of predictive coding for numeric data, leveraging correlations between consecutive values to enhance compression rates while preserving precision. The method enhances compression efficiency in high-correlation sequential data scenarios. For example, the integer sequence $S = \langle 7, 11, 15, 11, 15, 19 \rangle$ can be represented as an initial value $c_1 = 7$, a gap constant $d = 4$, and a fluctuation Boolean sequence $V = \langle 1, 1, -1, 1, 1 \rangle$, where 1 in V indicates an increase of d and -1 in V indicates a decrease of d. This example achieves perfect compression with zero error.

In daily life, compressed data often deviates from the original data, meaning the error is not always zero. To prevent the accumulation of errors and excessive distortion, the data can be divided into fixed-length segments, with each segment being compressed individually to maintain the accuracy of the compressed data.

Given an integer sequence S of length n, the NDC problem

This research work was partially supported by National Science and Technology Council of Taiwan under contract NSTC 112-2221-E-110-026-MY2.

TABLE I

AN EXAMPLE OF NDC WITH AN INTEGER SEQUENCE S AND A GAP CONSTANT $d = 9$. THE GAPPED SEQUENCE C MINIMIZE THE CUMULATIVE ERROR $E(S, C)$, WHICH IS 70. C CAN BE REPRESENTED AS (c_1, d, V).

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
s_i	33	17	38	32	11	26	25	35	29	40	18	26	41	22
c_i	31	22	31	22	13	22	31	40	31	40	31	22	31	22
v_i	-1	1	-1	-1	1	1	1	-1	1	-1	-1	1	-1	
$\|e_i\|$	2	5	7	10	2	4	6	5	2	0	13	4	10	0

seeks a gap constant d that minimizes the cumulative error between S and a predicted gap sequence C. Among four proposed variants, the third can be solved in $\mathrm{O}(\min(n^3, rn))$ time, where $r = S_{\max} - S_{\min}$.

The remainder of this paper is organized as follows. Section II gives the definitions of the NDC problem. Section III presents our algorithms for solving the NDC problem. In Section IV, we illustrate some experimental results. Finally, Section V provides our conclusions.

II. THE DEFINITIONS OF THE NUMERICAL DATA COMPRESSION PROBLEM

Given a numerical sequence S, the *numerical data compression* (NDC) problem aims to determine a gap constant d and a gapped numeric sequence that minimizes the cumulative error between S and C. An example illustrating the NDC problem is shown in Figure 1, and the resulting gapped numeric sequence is presented in Table I.

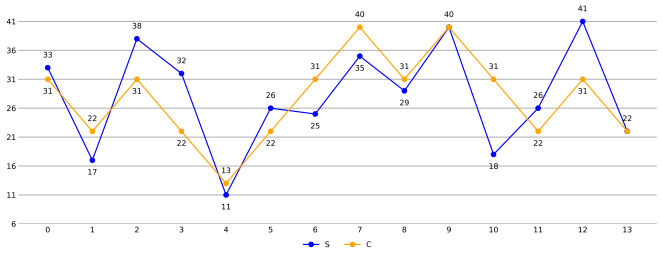

Fig. 1. An example of NDC with a given numeric sequence S and a gap constant $d = 9$. The gapped numeric sequence C minimize the cumulative error $E(S, C)$, which is 70.

Definition 1. (gapped numeric sequence) *A* gapped numeric sequence *is a numeric sequence* $C = \langle c_1, c_2, \ldots, c_n \rangle$ *in which each pair of consecutive elements differs by a fixed* gap constant d. *Specifically, for all* $1 \le i \le n-1$, *either* $c_{i+1} = c_i + d$ *or* $c_{i+1} = c_i - d$.

Definition 2. (fluctuation sequence) *A* fluctuation sequence *is a Boolean sequence* $V = \langle v_1, v_2, \ldots, v_{n-1} \rangle$ *that represents the directional variation of a gap numeric sequence* $C = \langle c_1, c_2, \ldots, c_n \rangle$, *with a gap constant* d, *defined as follows.*

$$v_i = \begin{cases} 1 & \text{if } c_{i+1} = c_i + d, \\ -1 & \text{if } c_{i+1} = c_i - d \end{cases} \quad \text{for } 1 \le i \le n-1. \qquad (1)$$

Definition 3. (error and cumulative error) *Given two numeric sequences* $S = \langle s_1, s_2, \ldots, s_n \rangle$ *and* $C = \langle c_1, c_2, \ldots, c_n \rangle$, *the* error *between every pair of corresponding elements is* $e_i = s_i - c_i$. *The* cumulative error *between* S *and* C *is* $E(S, C) = \sum_{k=1}^{n} |e_k|$.

TABLE II

FOUR VERSIONS OF THE NDC PROBLEM.

NDC problem	Given	To find	Time complexity
$\mathrm{P}(S, d, \phi, V)$	S, d, V	c_1	$\mathrm{O}(n)$
$\mathrm{P}(S, d, c_1, \phi)$	S, d, c_1	V	$\mathrm{O}(n^2)$
$\mathrm{P}(S, d, \phi, \phi)$	S, d	c_1, V	$\mathrm{O}(\min(n^3, rn))$
$\mathrm{P}(S, \phi, \phi, \phi)$	S	d, c_1, V	not available

TABLE III

AN EXAMPLE OF $\mathrm{P}(S, d, \phi, V)$ WITH A GIVEN NUMERIC SEQUENCE S, A GAP CONSTANT $d = 9$ AND A FLUCTUATION SEQUENCE V. THE GAPPED SEQUENCE C MINIMIZE THE CUMULATIVE ERROR $E(S, C)$, WHICH IS 77.

i	1	2	3	4	5	6	7
s_i	43	17	39	32	11	26	27
v_i	1	-1	-1	1	-1	1	
c_i	35	44	35	26	35	26	35
$\|e_i\|$	8	27	4	6	24	0	8

The sequence C can be completely represented by d, the initial value c_1 and a fluctuation Boolean sequence. An example is provided in Table I. This representation substantially reduces storage requirements, enabling effective data compression.

Definition 4. (four versions of the numerical data compression problem) *Given an integer-valued numeric sequence* $S = \langle s_1, s_2, \cdots, s_n \rangle$, *the* numerical data compression (*NDC*) *problem seeks to determine a gap constant* d *and a gapped numeric sequence* $C = \langle c_1, c_2, \cdots, c_n \rangle$ *that minimizes the cumulative error* $E(S, C)$. *There are four versions of the NDC problem, given in Table II.*

In Table II, for the problem instance $\mathrm{P}(S, d, \phi, V)$, the values of S, d and V are already given, so the goal is to determine c_1. Since this version includes the most constraints, it is the easiest to solve among the four versions. Examples illustrating the first three versions are shown in Tables III, IV, and V.

III. OUR ALGORITHMS

Definition 5. (shifted sequence) *Given a numeric sequence* $C = \langle c_1, c_2, \ldots, c_n \rangle$ *and a shift constant* δ, *the* shifted sequence C_δ *is obtained from shifting each element of* C *by adding* δ *to it.That is,* $C_\delta = \langle c_1 + \delta, c_2 + \delta, \ldots, c_n + \delta \rangle$.

TABLE IV

AN EXAMPLE OF $\mathrm{P}(S, d, c_1, \phi)$ WITH A GIVEN NUMERIC SEQUENCE S, A GAP CONSTANT $d = 9$ AND A FLUCTUATION SEQUENCE V. THE GAPPED SEQUENCE C MINIMIZE THE CUMULATIVE ERROR $E(S, C)$, WHICH IS 47.

i	1	2	3	4	5	6	7
s_i	43	17	39	32	11	26	27
v_i	-1	1	-1	-1	1	-1	
c_i	43	34	43	34	25	34	25
$\|e_i\|$	0	17	4	2	14	8	2

TABLE V
AN EXAMPLE OF P(S, d, ϕ, ϕ) WITH A GIVEN NUMERIC SEQUENCE S, A GAP CONSTANT $d = 9$ AND A FLUCTUATION SEQUENCE V. THE GAPPED SEQUENCE C MINIMIZE THE CUMULATIVE ERROR $E(S, C)$, WHICH IS 39.

i	1	2	3	4	5	6	7
s_i	43	17	39	32	11	26	27
v_i	-1	1	-1	-1	1	-1	
c_i	39	30	39	30	21	30	21
$\lvert e_i \rvert$	4	13	0	2	10	4	6

Definition 6. *The terms $e^+(S, C)$, $e^-(S, C)$, $e^=(S, C)$ denote the number of indices for which s_i is greater than, less than, or equal to c_i, respectively. That is,*

$$e^+(S, C) = |\{i \mid s_i - c_i > 0, 1 \le i \le n\}| \quad (2)$$

$$e^-(S, C) = |\{i \mid s_i - c_i < 0, 1 \le i \le n\}| \quad (3)$$

$$e^=(S, C) = |\{i \mid s_i - c_i = 0, 1 \le i \le n\}| \quad (4)$$

For example, in Table III, we have $e^+(S, C) = 3$, $e^-(S, C) = 3$, and $e^=(S, C) = 1$.

A. Algorithm for P(S, d, ϕ, V)

Theorem 1 (alignment of P(S, d, ϕ, V))**.** *There exists an aligned solution C to P(S, d, ϕ, V) such that there is an index i, with $1 \le i \le n$, where $s_i = c_i$.*

Proof. Omitted here. □

For solving P(S, d, ϕ, V), since V is given, $E(S, C)$ is fixed. Based on Theorem 1, we can shift C to another aligned solution C', with the same cumulative error. This can be achieved by finding the median of all e_i, and making its error to be zero using the shift concept. Then, the value of c_1 can be determined accordingly. Thus, the time complexity for solving P(S, d, ϕ, V) is O(n).

B. Algorithm for P(S, d, c_1, ϕ)

In P(S, d, c_1, ϕ), the initial value c_1 is fixed. Let $D_{i,j}$ represent the minimal cumulative error $\sum_{k=1}^{i} |e_k|$ where $c_i = c_1 + j \times d$. The following dynamic programming formula can solve P(S, d, c_1, ϕ).

$$D_{i,j} = \min \begin{cases} D_{i-1,j-1} + |c_1 + j \times d - s_i|, \\ D_{i-1,j+1} + |c_1 + j \times d - s_i| \end{cases}$$
$$\text{for } 2 \le i \le n, -(n-1) \le j \le n-1.$$
$$D_{1,0} = |c_1 - s_1|;$$
$$D_{1,j} = \infty, \quad \text{for } j \ne 0. \quad (5)$$

Equation 5 compute all possible $D_{i,j}$, each of which requires constant time. Therefore, the time complexity for solving P(S, d, c_1, ϕ) is O(n^2).

C. First Algorithm for P(S, d, ϕ, ϕ)

Theorem 2 (alignment of P(S, d, ϕ, ϕ))**.** *There exists a solution C to P(S, d, ϕ, ϕ) such that there is an index i where $s_i = c_i$ for $1 \le i \le n$.*

Proof. Omitted here. □

According to Theorem 2, the solution C aligns at a specific index k. We can try every possible value of k in the range from 1 to n. Consequently, the sequence S can be split into two parts: $S_{1:k}$ and $S_{k..n}$. By reversing the first part $S_{1:k}$ to become $\overline{S_{1..k}}$, P(S, d, ϕ, ϕ) can be solved by solving P$(\overline{S_{1..k}}, d, s_k, \phi)$ and P$(S_{k..n}, d, s_k, \phi)$, where the initial values of both parts are set to s_k. The overall algorithm for solving P(S, d, ϕ, ϕ) is presented in Algorithm 1. Since there are n possible values for k, the overall time complexity of Algorithm 1 is O(n^3).

Algorithm 1 The dynamic programming method for solving P(S, d, ϕ, ϕ)

Input: An integer-valued numeric sequence $S = \langle s_1, s_2, \ldots, s_n \rangle$ along with a gap constant d.
Output: A gapped numeric sequence C, minimizing $E(S, C)$.

```
E^min ← ∞
for k = 1 to n do
    C_L ← solution of P(overline(S_{1..k}), d, s_k, φ)      ▷ Equation 5
    C_R ← solution of P(S_{k..n}, d, s_k, φ)                ▷ Equation 5
    C ← concatenate C_L and C_R with removing one of the duplicated value c_k
    if E(S, C) < E^min then
        E^min ← E(S, C)
        C^best ← C
Return C^best
```

D. Second Algorithm for P(S, d, ϕ, ϕ)

We propose another algorithm for solving P(S, d, ϕ, ϕ).

Theorem 3. (range of c_i in P(S, d, ϕ, ϕ)) *Let S_{min} and S_{max} denote the minimum and maximum elements in S, respectively. The possible value for each c_i of a solution C to P(S, d, ϕ, ϕ) lies within the range $[S_{min} - d, S_{max} + d]$. That is,*

$$S_{min} - d \le c_i \le S_{max} + d, \quad 1 \le i \le n.$$

Proof. Omitted here. □

Based on Theorem 3, we can apply the dynamic programming formula to all possible grid points within the range $[S_{\min} - d, S_{\max} + d]$ for each index i.

Let $D'_{i,j}$ denote the cumulative error from index 1 to index i, where $i \in [1, n]$ and $j \in [S_{\min} - d, S_{\max} + d]$. Here, the value of $c_{i,j}$ is set to j. Note that the second index j of $c_{i,j}$ represents each possible value of c_i.

The DP formula for calculating $D'_{i,j}$ is given as follows.

$$D'_{1,j} = |s_1 - j|, \quad \text{for } S_{min} - d \le j \le S_{max} + d. \quad (6)$$

$$D'_{i,j} = \min \begin{cases} D'_{i-1,j-d} + |j - s_i| \\ D'_{i-1,j+d} + |j - s_i| \end{cases}$$
$$\text{for } 2 \le i \le n, S_{min} - d \le j \le S_{max} + d. \quad (7)$$

Equation 7 for solving P(S, d, ϕ, ϕ) requires O(rn) time, where r is the size of the range $[S_{\min} - d, S_{\max} + d]$. The time complexity of solving the same problem in Algorithm 1

TABLE VI
COMPRESSION RATIOS AND ERROR RATIOS FOR THREE STOCK PRICE SEQUENCES WITH VARIOUS SEGMENT LENGTHS.

Seg. len. / Stock ID		7	10	15	20
2317	compression	9.76	12.12	13.10	13.85
	error	0.0051	0.0059	0.0061	0.0065
2330	compression	8.33	9.89	11.82	13.04
	error	0.0055	0.0060	0.0065	0.0073
2603	compression	10.20	11.27	12.75	13.59
	error	0.0081	0.0093	0.0104	0.0109

is $\mathrm{O}(n^3)$. By combining these two algorithms, we obtain a hybrid approach for solving $\mathrm{P}(S, d, \phi, \phi)$ in $\mathrm{O}(\min(n^3, rn))$ time.

IV. EXPERIMENTAL RESULTS

In practice, we divide a data sequence into multiple segments, and compress each of them independently, with its own initial value and gap constant. As a result, only a small number of integers need to be stored, while the remaining values can be stored using less space.

To evaluate the effect of various segment lengths on compression performance, we define the *error ratio* to quantify the loss introduced by compression as follows.

$$\text{error ratio} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{s_i}\right| \tag{8}$$

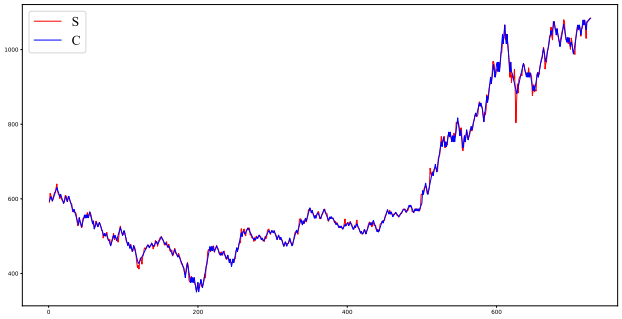

Fig. 2. An example of applying the NDC algorithm to compress stock data. The original sequence S consists of 726 closing prices of TSMC stock from January 1, 2022, to December 31, 2024.

We conducted experiments using 726 daily closing prices of three stocks in Taiwan Stock Market Index, from January 1, 2022, to December 31, 2024. The data was compressed using various segment lengths to analyze the impact on performance. The experimental results are summarized in Table VI.

The original data of stock 2330 (TSMC) and its corresponding gapped numeric sequence compressed with a segment length of 20 are illustrated in Figure 2. Although the NDC algorithm introduces some loss, the errors primarily appear in short-term irregular fluctuations. The gapped numeric sequence effectively preserves the overall trend of the stock price.

V. CONCLUSION

In this paper, we define a new problem, the numeric data compression (NDC) problem, which focuses on relationships between data elements for compression. By recording these relationships, we eliminate the need to store the original data.

We define four versions of the NDC problem. Our algorithm to solving the third version has a time complexity of $\mathrm{O}(\min(n^3, rn))$, where $n = |S|$ and $r = S_{\max} - S_{\min}$. For solving the fourth version $\mathrm{P}(S, \phi, \phi, \phi)$, a straightforward approach is to try each possible value for d in the range $[S_{\min}, S_{\max}]$. Then the time complexity will be $\mathrm{O}(\min(rn^3, r^2n))$. One direction of the future works is to investigate the properties of the gap constant d, which may lead to the design of a more efficient algorithm.

Another potential direction for future work is to extend the NDC problem by incorporating more generalized relationships to achieve further data compression. For instance, we can consider patterns such as increasing or decreasing by one or more units of d. Additionally, multiple interval constants $d_1, d_2, \ldots, d_k$ can be utilized simultaneously to capture complex relationships among data elements. Exploring these extensions could lead to more flexible and efficient compression methods.

REFERENCES

[1] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[2] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on pure and applied mathematics*, vol. 41, no. 7, pp. 909–996, 1988.

[3] P. Elias, "Predictive coding–I," *IRE Transactions on Information Theory*, vol. IT-1, no. 1, pp. 16–24, 1955.

[4] ——, "Predictive coding–II," *IRE Transactions on Information Theory*, vol. IT-1, no. 1, pp. 24–33, 1955.

[5] V. Engelson, D. Fritzson, and P. Fritzson, "Lossless compression of high-volume numerical data from simulations," *Data Compression Conference*, p. 574, 2000.

[6] D. L. Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, pp. 46–58, 1991.

[7] P. R. Geethakumari and I. Sourdis, "Streamzip: Compressed sliding-windows for stream aggregation," *International Conference on Field-Programmable Technology*, pp. 1–9, 2021.

[8] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[9] G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, no. 1, pp. 135–147, 1999.

[10] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, 1st ed. Harcourt Brace Jovanovich, 2012.

[11] G. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[12] Welch, "A technique for high-performance data compression," *The Computer Journal*, vol. 17, no. 6, pp. 8–19, 1984.

[13] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[14] ——, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.