APPENDIX A
PROOF OF PROPOSITION

A. Proposition 1

Proposition. *The optimal solution of operation list must satisfy the Non-overlapping Constraint.*

Proof. The proof of this proposition is as follows. Consider a group of operations $v_1 = [o_1, o_2]$, where $o_i = (p_i, l_i, s_i)$ and $p_1 < p_2$. Suppose $\sigma_1\{n\}$ is transformed into $\sigma_2\{m\}$ by these two operations in order, and there exists an overlap between these two operations, which can be expressed as $p_1 < p_2 < p_1 + \text{len}(s_1)$. According to the formula in Definition 1, σ_1 is transformed as follows:

If $\text{len}(s_1) > p_2 - p_1 + l_2$:

$$\begin{aligned} o_2 : \sigma_1 \to \sigma_1[0, (p_1 - 1)] + s_1[0, (p_2 - p_1 - 1)] + s_2 \\ + s_1[(p_2 - p_1 + l_2), (\text{len}(s_1) - 1)] + \\ + \sigma_1[(p_1 + l_1), (n - 1)]. \end{aligned}$$

Else, $\text{len}(s_1) \leq p_2 - p_1 + l_2$:

$$\begin{aligned} o_2 : \sigma_1 \to \sigma_1[0, p_1 - 1] + s_1[0, p_2 - p_1 - 1] + s_2 \\ + \sigma_1[p_1 + l_1 + (l_2 - \text{len}(s_1) + p_2 - p_1), n - 1]. \end{aligned}$$

There exists a new operation that is equivalent to the aforementioned two substitutions. Suppose that there is an operation $o' = (p', l', s')$ such that:

If $\text{len}(s_1) > p_2 - p_1 + l_2$:

$$\begin{aligned} p' = p_1,\ l' = l_1, \\ s' = s_1[0, p_2 - p_1 - 1] + s_2 + s_1[p_2 - p_1 + l_2, \text{len}(s_1) - 1]. \end{aligned}$$

Else, $\text{len}(s_1) \leq p_2 - p_1 + l_2$:

$$\begin{aligned} p' = p_1,\ l' = l_1 + (l_2 - \text{len}(s_1) + p_2 - p_1), \\ s' = s_1[0, p_2 - p_1 - 1] + s_2. \end{aligned}$$

It can be easily verified by definition that o' completes the transformation of the two operations $v_1 = (o_1, o_2)$. Treat o' as a single operation list, denoted $v_2 = [o']$. Consider the cost of two groups v_1 and v_2:

$$\begin{aligned} C(v_1) = C(p_1) + C(p_2) + C(l_1) + C(l_2) + C(s_1) + C(s_2), \\ C(v_2) = C(p') + C(l') + C(s'). \end{aligned}$$

Note that typically different positions occupies the same space, implying that they should get equal cost values. This principle also holds for length. Regarding string information, the space occupied (i.e., the cost) ought to be proportional to its length. On account of $\text{len}(s') \leq \text{len}(s_1) + \text{len}(s_2)$, it can be obtained that $C(v_1) > C(v_2)$.

This indicates that an operation list with less cost is discovered, which accomplishes the same effect as the initial group of operations. For any group of operations with overlap, it can be transformed into a list with reduced cost. Hence, the group of operations within the optimal solution should satisfy the Non-overlapping Constraint, demonstrating that optimal solution can be converted into an operation list. □

B. Proposition 2

Proposition. *Given that l_1, l_2 are the lengths of σ_1, σ_2,the time complexity of Algorithm 2 is $O(l_1^2 \times l_2^2)$.*

Proof. The Algorithm 1 initializes a 2D array of size $x \times y$ and then employs nested loops iterating over x and y. Since the conditional checks and assignments within the loops can be completed in constant time, and the maximum values of x and y are bounded to l_1 and l_2, the overall complexity for Algorithm 1 is $O(l_1 \times l_2)$. Furthermore, Algorithm 2 creates a 2D array of size $l_1 \times l_2$, costs $O(l_1 \times l_2)$. Each iteration of the nested loop involves a call to Algorithm 1, a minimum value search and two assignments. As previously analyzed, the complexity of Algorithm 1 is $O(l_1 \times l_2)$. The range for the minimum value search corresponds to the return value of Algorithm 1, with a maximum size of $l_1 \times l_2$, making its time complexity also $O(l_1 \times l_2)$. Therefore, the total time complexity is $O(l_1 \times l_2) \times O(l_1 \times l_2) = O(l_1^2 \times l_2^2)$. □

C. Proposition 3

Proposition. *Minimizing the cost of the Variable-Length Substitution List is equivalent to maximizing the cost of the Dual Identification List.*

Proof. The proof is quite intuitive, as the cost of Identification is defined based on the cost generated Variable-Length Substitution and the changes in cost are exactly opposite. This indicates that an increase in the cost of Dual Identification List is equivalent to a decrease in the cost of Variable-Length Substitution List, thereby making the two optimization problems equivalent. □

D. Proposition 4

Proposition. *Given that l_1, l_2 are the lengths of σ_1, σ_2,the time complexity of Algorithm 3 is average $O(l_1 + l_2)$.*

Proof. The algorithm first calculates the Q-gram lists and their intersection, which can be completed in $O(l_1 + l_2)$ time. Although the core of the algorithm consists of two nested loops, it effectively reduces the number of iterations using i_1 and i_2. Intuitively, using $i_1 + i_2$ as a checkpoint, each iteration increases at least one of i_1 or i_2 by 1, resulting in a total loop count of $O(l_1 + l_2)$. In each iteration, the algorithm needs to look up q (line 7). The average complexity of this search is $O(1)$, while the worst-case complexity is $O(l_2)$ when a hash table is maintained [19]. Therefore, the average complexity of the algorithm is $O(l_1 + l_2) + O(l_1 + l_2) \times O(1)$, simplifying to $O(l_1 + l_2)$. □

APPENDIX B
ADDITIONAL EXPERIMENTS

A. Data Size

Data Size refers to the number of log entries in a file, indicating the volume of log data. The *Data Size* values range from $1,000$ to $20,000$, increasing by 500 entries to simulate the case of small batch data. The evaluation dataset

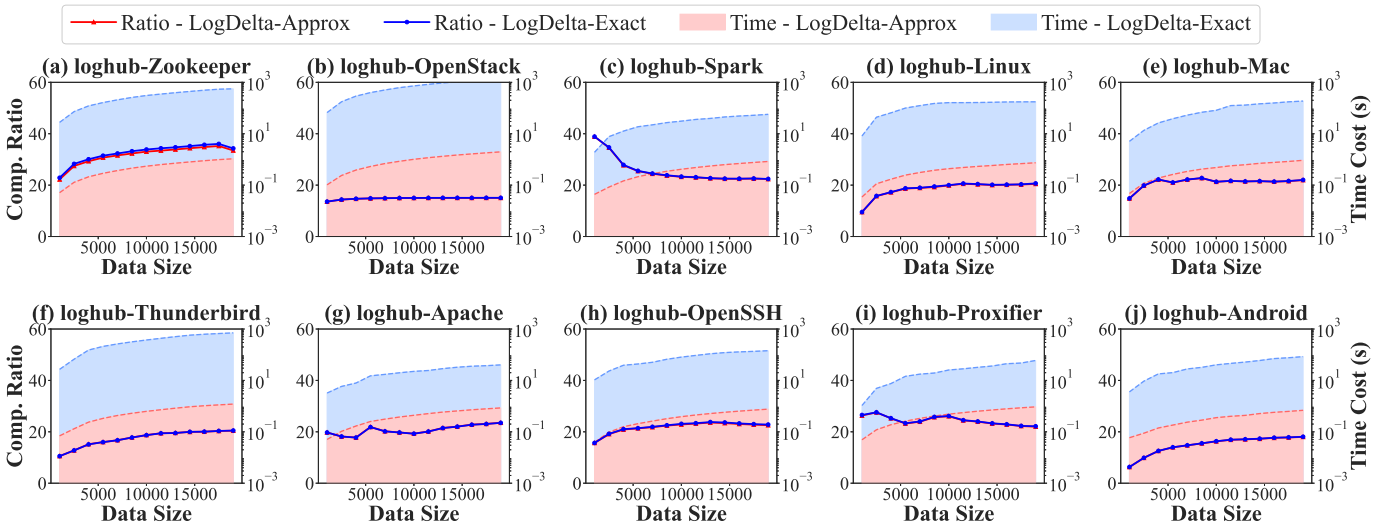

Fig. 17: Varying Data Sizes.

is generated by selecting a set number of entries from the start of each dataset.

Figure 17 illustrates the variation of compression effectiveness with data size. The solid line represents the compression ratio, while the shaded area indicates compression time.

In terms of compression ratio, LogDelta-Approx and LogDelta-Exact perform similarly across all datasets, with minor fluctuations. These variations occur because different dataset sections exhibit varying local similarity, and log data often has an irregular distribution. Consequently, the redundant information varies, causing compression ratio fluctuations, while overall performance remains stable. A notable exception is the Zookeeper dataset, where a significant difference in compression ratio exists between the Approx and Exact methods. This discrepancy arises from misaligned matches in certain strings, where high similarity does not translate to accurate matching positions. As a result, the approximate algorithm redundantly classifies longer substrings as operations, increasing encoding information compared to the exact algorithm.

Regarding time cost, LogDelta-Approx consistently outperforms LogDelta-Exact by more than two orders of magnitude in speed across all datasets. This substantial difference underscores the efficiency of the approximate method. Achieving similar compression ratios while significantly reducing time cost demonstrates the practical benefits of LogDelta-Approx, especially in real-time or resource-constrained environments.