Appendix

The Sub-column determination problem is essentially one of bit width β search. Our proposed algorithm is a cost-model-based algorithm to find optimal bit width of sub-columns. The unique challenges specific to this problem is to find optimal storage cost of sub-columns with less compression time. We replace the exhaustive search for the optimal sub-column bit width with a Particle-Swarm-Optimization (PSO) based procedure in Algorithm 4. In Lines 4-9, algorithm initial i-th particle's position p_i, its velocity v_i, the best i-th position $p_{best,i}$, the minimum cost $Cost_{\min,i}$, the global best position p_{best}, and the global minimum cost $C_{\min}$. Then, after $t_{\max}$ iterations, the algorithm updates the minimum cost for each particle and the global minimum cost. Particles update their velocities and positions using the standard PSO rule with inertia w and cognitive/social coefficients c_1, c_2 in Lines 12-13. The global minimum cost $C_{\min}$ is updated whenever any particle finds a lower cost in Lines 14-18. Finally, in the vicinity of the optimal solution β' found by the global search of PSO, another local refinement is performed in Lines 19-23.

Since Algorithm 4 updates s particles with $t_{\max}$ iterations, the total runtime is approximately $s * t_{\max} * n$ and the time complexity is $O(n)$. The parameters s and $t_{\max}$ could be tuned to trade off computational cost against solution quality, which influences compression ratio. This method requires far fewer time cost evaluations than a full exhaustive search for large M in Algorithm 1, when $s * t_{\max}$ is far smaller than M.

Algorithm 4: Sub-column Determination with PSO

Input: Series $X = (x_1, x_2, \dots, x_n)$, PSO hyperparams: swarmSize s, maxIter $t_{\max}$, inertia w, cognitive coefficients c_1, social coefficients c_2, localRadius R

Output: Bit width β'

1 $M = \lceil \log(x_{\max} - x_{\min} + 1) \rceil$;
2 **for** $i \leftarrow 1$ **to** s **do**
3     $p_i = U(1, M)$;
4     $v_i = U(-(M-1)/2, (M-1)/2)$;
5     $p_{best,i} = p_i$;
6     $Cost_{\min,i} = Cost(X, round(p_i))$;
7 $p_{best} = \arg\min_i Cost_{\min,i}$;
8 $C_{\min} = \min Cost_{\min,i}$;
9 $v_{max} = M$;
10 **for** $t \leftarrow 1$ **to** $t_{\max}$ **do**
11   **for** $i \leftarrow 1$ **to** s **do**
12         $r_1 = U(0,1), r_2 = U(0,1)$, $v_i = w \cdot v_i + c_1 r_1 (p_{best,i} - p_i) + c_2 r_2 (p_{best} - p_i)$, clamp v_i to $[-v_{\max}, v_{\max}]$;
13         $p_i = p_i + v_i$, clamp p_i to $[1, M]$;
14         $\beta = round(p_i)$;
15   **if** $C(X, \beta) < p_{best,i}$ **then**
16             $Cost_{\min,i} = C(X, \beta)$, $p_{best,i} = p_i$;
17   **if** $C(X, \beta) < C_{\min}$ **then**
18                 $C_{\min} = C(X, \beta)$, $p_{best} = p_i$;
19 $\beta' = round(p_{best})$;
20 **for** $b \leftarrow \beta' - R$ **to** $\beta' + R$ **do**
21   **if** $1 \le b \le M$ **then**
22   **if** $Cost(X, b) < C_{\min}$ **then**
23             $C_{\min} = Cost(X, b)$, $\beta' = b$;
24 **return** β';
