APPENDIX

A. Proof of Proposition 1

As shown in Algorithm 2, the first step (Line 1) constructs distance matrices for all attributes by calling the algorithm PDM. For each attribute A_j, PDM divides the rows of the distance matrix among the $|\mathcal{P}|$ processors so that each processor computes the distances for all tuple pairs in its assigned rows. After combining partial results, this yields the entire distance matrix. Since there are m attributes and $n \times n$ tuple pairs, the overall cost is $O\left(\frac{m \cdot n^2}{|\mathcal{P}|}\right)$.

Next, Algorithm 2 invokes TAGC (Line 3) to generate clusters for each attribute, as presented in Algorithm 1. In particular, Line 1 sorts the thresholds Θ_j in constant time, because the size of Θ_j is small, while Line 2 must traverse the entire distance matrix in $O\left(n^2\right)$ time to obtain $C_j[\theta_j^{\max}]$. Then, in Lines 3–5, TAGC refines that maximum-threshold clustering without a second full traversal, instead iterating over the previously computed clusters and thus costing less than $O\left(n^2\right)$. Throughout this process, the distance matrix rows are partitioned among the $|\mathcal{P}|$ processors, so that each processor handles clustering for all relevant thresholds Θ_j within its assigned rows. Consequently, generating clusters for every attribute in parallel over m attributes requires $O\left(\frac{m \cdot n^2}{|\mathcal{P}|}\right)$ time overall.

Afterward, in Lines 4–16, Algorithm 2 conducts the complete error detection process for every constraint $\varphi \in \Sigma$. Specifically, in Line 6, the algorithm ASPT partitions the tuple pair sets from all relevant attribute clusters of φ across $|\mathcal{P}|$ processors in parallel. Since these tuple pair sets are assigned by rows and the distance matrix is of size $n \times n$ with m attributes, ASPT takes $O\left(\frac{m \cdot n}{|\mathcal{P}|}\right)$ time. Next, in Line 7, Algorithm 2 calls PFVP to find violating tuple pairs by performing set operations (intersection and difference) among the tuple pair sets derived from each attribute cluster of φ. Because each cluster can contain up to $O\left(n^2\right)$ tuple pairs, the row-based partitioning strategy also applies here: each processor works on the subset of rows assigned to it, carrying out intersection and difference in parallel. As a result, executing these set operations for m attributes among $|\mathcal{P}|$ processors costs $O\left(\frac{m \cdot n^2}{|\mathcal{P}|}\right)$. Having obtained all violating tuple pairs $C_\varphi^{\mathcal{P}}$, Lines 8–16 traverse each pair (t_i, t_j) to identify erroneous cells in constant time, thus taking $O\left(n^2\right)$ time overall in that final step.

Summing up, for each constraint φ in Σ, the cost is $O\left(\frac{m \cdot n^2}{|\mathcal{P}|}\right)$. Because there are $|\Sigma|$ such constraints, the total time complexity of PED is $O\left(\frac{|\Sigma| \cdot m \cdot n^2}{|\mathcal{P}|}\right)$ as stated in proposition 1.

B. Proof of Proposition 2

Algorithm IPED begins by calling IPDM, which incrementally updates the distance matrix from size $n \times n$ to $(n + \Delta n) \times (n + \Delta n)$. Since only rows $1 \ldots (n + \Delta n)$ and columns $n \ldots (n + \Delta n)$ must be filled, each of the $|\mathcal{P}|$ processors is assigned a subset of these rows and computes the tuple pair distances within them, resulting in a total cost of $O\left(\frac{m \cdot (n+\Delta n) \cdot \Delta n}{|\mathcal{P}|}\right)$ for m attributes.

Next, the algorithm IPED invokes TAGC to build incremental clusters ΔC_j for each attribute A_j based on the new incremental distance matrix ΔD_j. TAGC partitions the rows $1 \ldots (n+\Delta n)$ among the $|\mathcal{P}|$ processors, generating the clusters for its assigned rows. Because the relevant tuple pairs now total $(n+\Delta n) \cdot \Delta n$ instead of n^2, the cost to generate clusters for m attributes is $O\left(\frac{m \cdot (n+\Delta n) \cdot \Delta n}{|\mathcal{P}|}\right)$.

Afterward, for each constraint $\varphi \in \Sigma$, the algorithm IPED detects errors incrementally. First, ASPT distributes the tuple pair sets derived from the clusters of φ among the $|\mathcal{P}|$ processors by rows, so traversing $(n + \Delta n)$ rows with m attributes costs $O\left(\frac{m \cdot (n+\Delta n)}{|\mathcal{P}|}\right)$. Then, PFVP finds violating tuple pairs by set operations (intersection and difference) over these tuple pair sets. Each cluster contains up to $(n + \Delta n) \cdot \Delta n$ tuples pairs, so carrying out these operations for m attributes in parallel takes $O\left(\frac{m \cdot (n+\Delta n) \cdot \Delta n}{|\mathcal{P}|}\right)$. Once PFVP identifies the violating tuple pairs $\Delta C_\varphi^{\mathcal{P}}$, the algorithm IPED scans each tuple pair (t_i, t_j) in constant time to detect erroneous cells, taking $O\left((n + \Delta n) \cdot \Delta n\right)$ time overall for that final step.

Hence, each constraint $\varphi \in \Sigma$ requires $O\left(\frac{m \cdot (n+\Delta n) \cdot \Delta n}{|\mathcal{P}|}\right)$ time for incremental detection, and thus the total complexity over all $|\Sigma|$ constraints is $O\left(\frac{|\Sigma| \cdot m \cdot (n+\Delta n) \cdot \Delta n}{|\mathcal{P}|}\right)$ as stated in proposition 2.

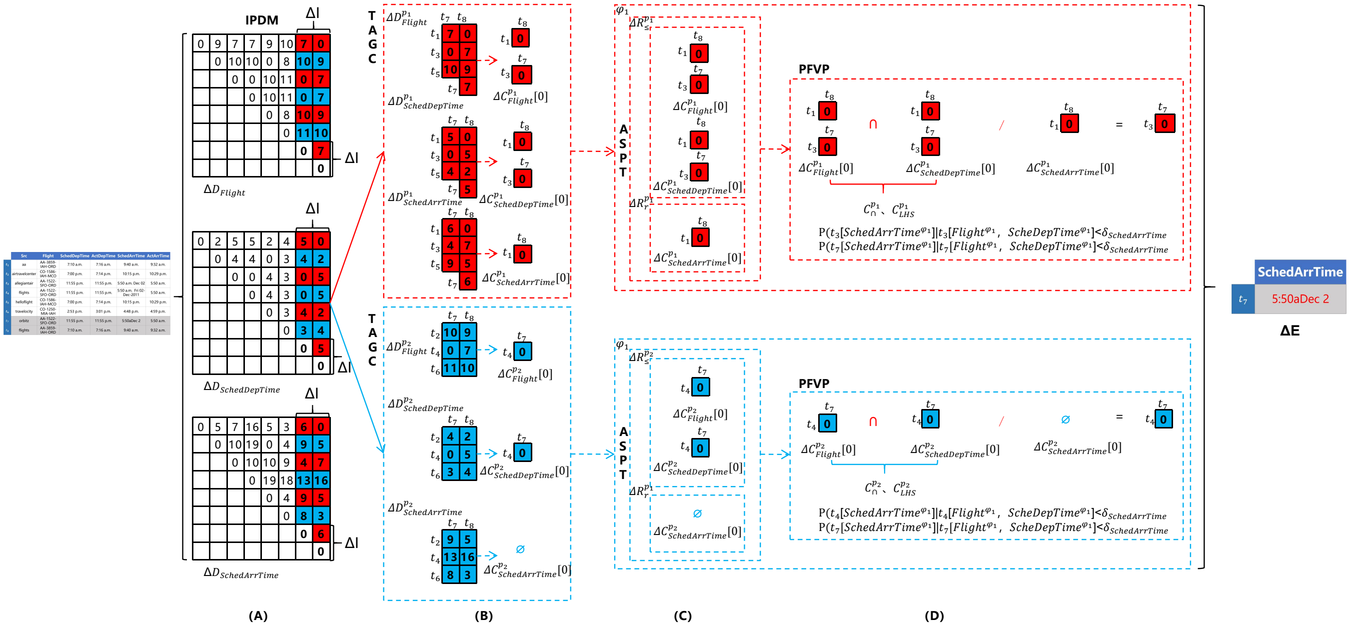

Figure 11: An Overview of Our Incremental Parallel Detection (IPED)