High Precision ≠ High Cost: Temporal Data Fusion for Multiple Low-Precision Sensors

JINGYU ZHU, College of Computer Science, Nankai University, China
YU SUN*, College of Computer Science, Nankai University, China
SHAOXU SONG, Tsinghua University, China
XIAOJIE YUAN, College of Computer Science, Nankai University, China

High-quality data are crucial for practical applications, but obtaining them through high-precision sensors comes at a high cost. To guarantee the trade-off between cost and precision, we may use multiple low-precision sensors to obtain the nearly accurate data fusion results at an affordable cost. The commonly used techniques, such as the Kalman filter and truth discovery methods, typically compute fusion values by combining all the observations according to predictions or sensor reliability. However, low-precision sensors can often cause outliers, and such methods combining all observations are susceptible to interference. To handle this problem, we select a single observation from multiple sensor readings as the fusion result for each timestamp. The selection strategy is guided by the maximum likelihood estimation, to determine the most probable changing trends of fusion results with adjacent timestamps. Our major contributions include (1) the problem formalization and NP-hardness analysis on finding the fusion result with the maximum likelihood w.r.t. local fusion models, (2) exact algorithms based on dynamic programming for tackling the problem, (3) efficient approximation methods with performance guarantees. Experiments on various real datasets and downstream applications demonstrate the superiority and practicality of our work in low-precision sensor data fusion.

CCS Concepts: • **Information systems** → **Information integration**.

Additional Key Words and Phrases: temporal data fusion; low-precision sensors

ACM Reference Format:
Jingyu Zhu, Yu Sun, Shaoxu Song, and Xiaojie Yuan. 2024. High Precision ≠ High Cost: Temporal Data Fusion for Multiple Low-Precision Sensors. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 143 (June 2024), 32 pages. https://doi.org/10.1145/3654946

1 INTRODUCTION

High-quality data are necessary for real scenarios [44, 45, 55, 56], to support the downstream analysis and applications, such as navigation [39, 51, 61] and environmental monitoring [24, 63]. Although high-precision sensors are often used to provide guarantees for data quality [2, 27], this may cause heavy costs. In practical applications, budget constraints are commonly encountered, encompassing not only the costs associated with sensor acquisition [11, 53, 60] but also those related to power consumption [58].

*Yu Sun is the corresponding author.

Authors' addresses: Jingyu Zhu, College of Computer Science, Nankai University, China, 2013216@mail.nankai.edu.cn; Yu Sun, College of Computer Science, Nankai University, China, sunyu@nankai.edu.cn; Shaoxu Song, Tsinghua University, China, sxsong@tsinghua.edu.cn; Xiaojie Yuan, College of Computer Science, Nankai University, China, yuanxj@nankai.edu.cn.

2836-6573/2024/6-ART143 $15.00
https://doi.org/10.1145/3654946

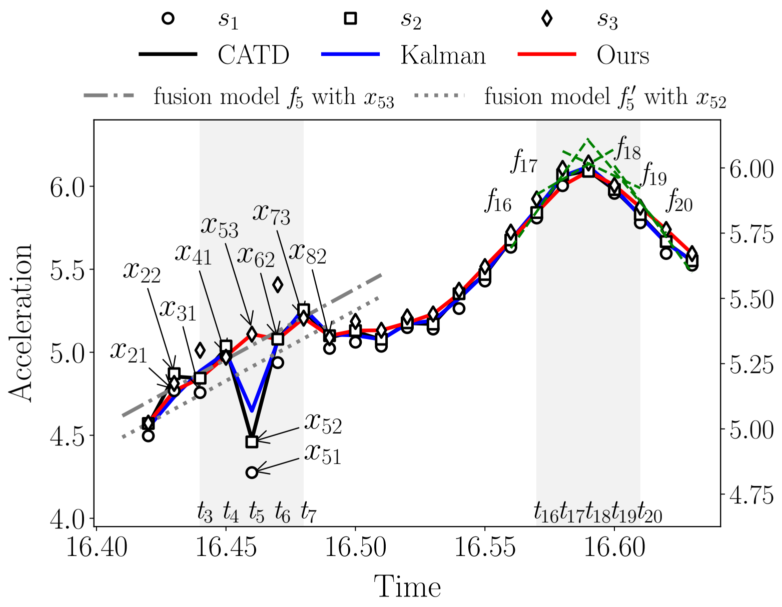

Fig. 1. Sensor readings and corresponding fusion results by different methods over the real IMU dataset, where outliers appear at the timestamp $t_5 = 16.46$.

Therefore, cost-effective low-precision sensors are commonly adopted in practical scenarios. However, low-precision sensors typically exhibit lower measurement accuracy and can often produce outliers. In this sense, we must consider the trade-off between precision and cost, trying to get nearly accurate data by fusing the observations of multiple low-precision sensors, as a cost-effective alternative to expensive sensors.

Existing methods typically use the Kalman filter [26] for sensor data fusion, which obtains the fusion result by combining prior prediction and current sensor observations. Hence, they are sensitive to outliers because the presence of outliers in observations can have a significant impact on the estimated values. While truth discovery methods [28, 31, 32, 42, 65, 68] aim to obtain the estimated truths by leveraging the sensor reliability, which is often iteratively computed based on the difference between estimated truths and observations. However, when a sensor with high weight occasionally generates an outlier, the estimated truths would be severely disrupted.

Drawing inspiration from the K-Median clustering [23], which considers the median point as the cluster center instead of mean values to resolve the impact of outliers, we choose a single observation from multiple sensor observations as the fusion result for each timestamp. The selection strategy is guided by the maximum likelihood estimation, which estimates the most probable trends across fusion results at adjacent timestamps.

Example 1. *Figure 1 presents the real IMU (Inertial Measurement Units) data, which collects real-world accelerations obtained by IMU sensors.*[1] *Directly using higher-precision IMU sensors would increase the cost from \$5 to \$1,000 even \$20,000 [35]. Therefore, fusing multiple low-precision IMU sensor readings to get nearly accurate data could be a good trade-off, especially when IMU is used for daily navigation, where the basic demand can be satisfied by relatively accurate location without the strict requirement for perfectly accurate readings.*

[1]Please see detailed explanations in Section 6.1.1.

As shown, different markers represent observations from different sensors, and solid lines with different colors represent the fusion results obtained by corresponding methods. Kalman filter [26], according to t_4 and more previous timestamps, combines all the observations at timestamp t_5 to get the inaccurate estimation 4.65. *On the other hand, the truth discovery method CATD [33] evaluates the sensor weight according to the difference between estimated values and observations, having* $9.73, 61.73, 4.53$ *for each sensor. Since most observations of the sensor s_2 are closest to the median of sensor readings, s_2 gets the largest weight. Finally, CATD gets the fusion result* 4.48 *for t_5, very close to the outlier x_{52}. In essence, this is because CATD relies on the overall performance of sensor readings to estimate the reliability, but lacks the fine-grained consideration for each individual observation.*

In contrast, to get the fusion result at t_5, our study selects the observation that best fits the temporal trend near the timestamp t_5. Specifically, we observe a consistent upward trend based on timestamps t_1-t_7. Selecting x_{51} or x_{52} would disrupt this trend, therefore, we opt for x_{53} at t_5.

In this work, (1) we design local fusion models to capture more fine-grained relationships between adjacent data, which can model distinct trends more accurately. For instance, as shown in Figure 1, the data changes between timestamps $t_3 - t_7$ and $t_{16} - t_{20}$ are very different. We thus learn different local fusion models $f_3 - f_7$ and $f_{16} - f_{20}$ for these two distinct changes respectively, where each timestamp (say t_{16}) corresponds to one local fusion model (say f_{16}) to capture data changes over its adjacent timestamps.[2] (2) To overcome the impact of outliers, rather than combining all observations to get the fusion result, we devise the selection strategy to choose only one observation that best conforms to local fusion models. Notably, it is indeed statistically explainable by the likelihood maximization. For instance, given three observations x_{51}, x_{52}, x_{53} at timestamp t_5 in Figure 1, since x_{53} leads to the maximum likelihood w.r.t. local fusion models (i.e., best conformance to the fusion model f_5 compared with x_{52} to f_5'), it is chosen as the fusion result.

1.1 Challenges

The problem of selecting the data fusion result with the maximum likelihood w.r.t. local fusion models is challenging. (1) Since there are multiple observations at each timestamp and a local fusion model captures the data changes across several adjacent timestamps, there are a lot of candidate fusion results w.r.t. each local fusion model. Therefore, it is not trivial to determine which fusion result owns the maximum likelihood w.r.t. a specific local fusion model. (2) Considering that the local fusion model captures continuous data changes with adjacent timestamps, the fusion result at each timestamp can be involved in several local fusion models. The problem thus becomes even harder, since the optimal fusion result at each timestamp should be determined by evaluating the likelihood w.r.t. various involved models.

1.2 Contributions

Our major contributions in this study are as follows.

(1) We formalize the problem of determining the optimal fusion result with the maximum likelihood w.r.t. local fusion models and analyze its hardness (Theorem 1) in Section 2. Then an overview in Section 3 outlines the end-to-end workflow of our study.

(2) We devise an exact method based on the dynamic programming in Section 4, to find the optimal data fusion result with pseudo-polynomial complexity (Proposition 2).

(3) We design an approximation algorithm by utilizing representative candidates for various observations in Section 5.1, to meet the aforesaid first challenge. Notably, it has an approximation performance guarantee by bounding the discrepancy between representative candidates and the corresponding observations (Proposition 5). To further solve the second challenge and improve

[2]Please see Section 2.1 for formal definitions and explanations of local fusion models.

Table 1. Notations

Symbol	Description
S	a set of sensors
n	the number of timestamps in the observed sequence
m	the number of sensors in a series of observed sequences
X_j	a sequence of n readings observed by the j-th sensor
X^i	the observations at i-th timestamp in X
x_{ij}	sensor readings observed by the j-th sensor at the i-th timestamp
f_i	a local fusion model at t_i
Y	a sequence of the fusion results
y_i	value in Y at the i-th timestamp
$\mathbf{y}_i$	y_i and the following κ values
κ	the length of the sequence f_i considers apart from y_i

the efficiency, we present a heuristic algorithm in Section 5.2, by reducing candidate trends across adjacent timestamps to limit the number of local fusion models.

(4) We conduct an extensive evaluation in Section 6 over real-world datasets with different numbers of sensors, different levels of cleanliness, and varying time spans. The ground truth is obtained by manually labeling or from dataset collectors. The data fusion comparison and downstream integration navigation application study demonstrate the superiority and practicality of our methods. Notably, the real-time processing experiments help bring forward potential practical uses of the approaches.

Table 1 lists the frequently used notations.

2 PROBLEM STATEMENT

In this part, we first introduce local fusion models in Section 2.1. The optimal multi-sensor data fusion problem for integrating observations from multiple low-precision sensors is then formally studied with the hardness analysis (Theorem 1) in Section 2.2.

2.1 Local Fusion Model

Consider a set of sensors $S = \{s_1, \ldots, s_m\}$ with the corresponding readings $X = \{X_1, X_2, \ldots, X_m\}$, where $X_j = \{x_{1j}, x_{2j}, \ldots, x_{nj}\}$ denotes the observed data by sensor s_j across n timestamps $\{t_1, \ldots, t_n\}$, and each $X^i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ represents the observations by all the sensors S at the timestamp t_i. Let $Y = \{y_1, \ldots, y_n\}$ represent the fusion result, based on observations X.

We denote

$$\mathbf{y}_i = \begin{pmatrix} y_i & y_{i+1} & \cdots & y_{i+\kappa} \end{pmatrix}^\top$$

as a vector of fusion values with timestamps $\{t_i, \ldots, t_{i+\kappa}\}$. To capture the changing regularity of sensor readings, we use the local fusion model f_i to predict each fusion value $y_i \in \mathbf{y}_i$ referring to its timestamp t_i,

$$f_i(t_i) \to y_i. \tag{1}$$

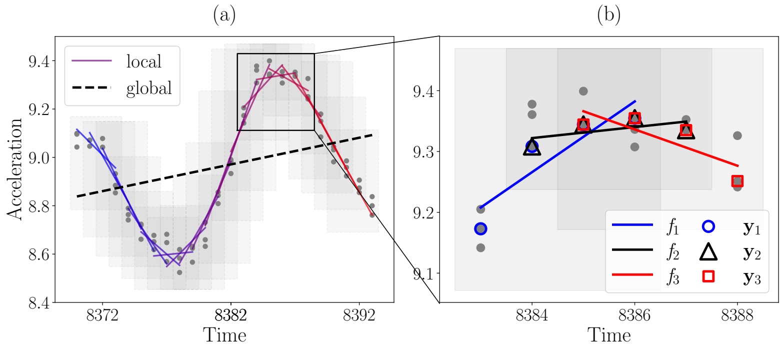

Fig. 2. (a) Fusion results obtained by global and local fusion models, and (b) detailed fusion results on sub-time series obtained by local fusion model.

Specifically, f_i can be a polynomial regression [4], logistic regression or simply linear regression [38].

For instance, given the timestamps

$$\mathbf{t}_i = \begin{pmatrix} 1 & t_i \\ 1 & t_{i+1} \\ \vdots & \vdots \\ 1 & t_{i+\kappa} \end{pmatrix},$$

we consider a linear regression[3] local fusion model f_i for predicting fusion values $\mathbf{y}_i$,

$$\mathbf{y}_i = \mathbf{t}_i \boldsymbol{\phi}_i + \boldsymbol{\epsilon}_i, \tag{2}$$

where $\boldsymbol{\phi}_i = (\phi_{i0} \quad \phi_{i1})^\top$ is the parameter of model f_i, and $\boldsymbol{\epsilon}_i = \begin{pmatrix}\epsilon_i & \epsilon_{i+1} & \dots & \epsilon_{i+\kappa}\end{pmatrix}^\top$ is the error term.

Such a design can achieve more accurate modeling for the sudden changes, and address the lag introduced by existing methods, e.g., Kalman filter [29, 34], due to the integration of past states during fusion. Figure 2 shows the sensor observations of IMU angular acceleration, measuring the angular velocity of a moving object. As shown in Figure 2(a), the global model over all the observations could only capture general trends of IMU angular acceleration changes. On the contrary, the local fusion model studies more fine-grained relationships between adjacent data $\{y_i, y_{i+1}, \dots, y_{i+\kappa}\}$ to get a more accurate depiction for the data changes of IMU angular acceleration. A series of rectangular shadows illustrate the evolution of local fusion models, while the solid lines reflect the continuous process where f_i is trained over $\mathbf{y}_i$. Figure 2(b) illustrates some specific local models, where each one is based on sensor observations within a certain time sequence range, e.g., the solid blue line with IMU angular acceleration observations at four timestamps.

Example 2. *Considering sensor data in Figure 1, with $\kappa = 3$. We could use f_5 to model data changes over* $\mathbf{y}_5 = \begin{pmatrix}x_{53} & x_{62} & x_{73} & x_{82}\end{pmatrix}^\top = \begin{pmatrix}5.11 & 5.08 & 5.21 & 5.10\end{pmatrix}^\top$, *having* $\boldsymbol{\phi}_5 = \begin{pmatrix}-11.35 & 1.0\end{pmatrix}^\top$. *Thus, we can obtain* $f_5(t_i) = -11.35 + 1.0 * t_i$.

[3]Experiments in Section 6 show that our methods achieve a better fusion result with the simple linear regression model, compared with existing works.

2.2 Multi-Sensor Data Fusion Problem

In real-world scenarios, low-precision sensor observations often contain outliers, which makes combining all observations susceptible to interference. Therefore, instead of combining all the observations to get a fusion result, we consider a set of fusion candidates

$$\mathrm{can}(y_i) = X^i = \{x_{i1}, x_{i2}, \dots, x_{im}\} \tag{3}$$

at the timestamp t_i, and devise a selection strategy g to obtain the most reasonable one from $\mathrm{can}(y_i)$ as the fusion result y_i, having

$$g(X^i) \rightarrow y_i. \tag{4}$$

Compared with existing methods such as Kalman filter [10, 15] or naive Bayes [18], the immediate benefit is that such a strategy could overcome the influence of outliers. As illustrated in Figure 2(b), outliers at the last timestamp may cause fusion results obtained by existing methods higher than expected. However, g will capture the reasonable trend of data changes due to the selection feature.

For each model f_i in Formula 1, we assume a normal distribution with zero mean and variance σ_i of the error term [48], i.e., $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. It is equivalent to $y_i \sim \mathcal{N}(f_i(t_i), \sigma_i^2)$, having

$$\mathcal{L}(y_i \mid X_i, f_i) = -\frac{\log(2\pi\sigma_i^2)}{2} - \frac{(y_i - f_i(t_i))^2}{2\sigma_i^2}.$$

Let $\mathbf{x}_i = \{X^i, \dots, X^{i+\kappa}\}$ denote the original sensor readings with the timestamps between t_i and $t_{i+\kappa}$. Then the likelihood of $\mathbf{y}_i$ w.r.t. $\mathbf{x}_i$ and the model f_i is

$$\mathcal{L}(\mathbf{y}_i \mid \mathbf{x}_i, f_i) = \sum_{k=0}^{\kappa} -\frac{\log(2\pi\sigma_{i+k}^2)}{2} - \frac{(y_{i+k} - f_i(t_{i+k}))^2}{2\sigma_{i+k}^2}.$$

Considering all the fusion results Y with the corresponding observations X and models $F = \{f_1, f_2, \dots f_{n-\kappa}\}$, we have the likelihood

$$\begin{aligned}\mathcal{L}(Y \mid X, F) &= \sum_{i=1}^{n-\kappa} \mathcal{L}(\mathbf{y}_i \mid \mathbf{x}_i, f_i) \\ &= \sum_{i=1}^{n-\kappa} \sum_{k=0}^{\kappa} -\frac{\log(2\pi\sigma_{i+k}^2)}{2} - \frac{(y_{i+k} - f_i(t_{i+k}))^2}{2\sigma_{i+k}^2}.\end{aligned} \tag{5}$$

As shown, given the readings X by low-precision sensors, to find the optimal fusion result Y with the maximum likelihood $\mathcal{L}(Y \mid X, F)$, it is equivalent to minimize the *fusion loss*

$$\mathscr{L}(Y) = \sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) = \sum_{i=1}^{n-\kappa} \sum_{k=0}^{\kappa} (y_{i+k} - f_i(t_{i+k}))^2. \tag{6}$$

That is, it measures how the fusion result Y conforms the corresponding local fusion models F.

Problem 1. *Given a set of multi-sensor time series data X, the* optimal multi-sensor data fusion *problem is to find a fusion result Y, such that $\mathscr{L}(Y)$ is minimized w.r.t. the local fusion models F.*

Hardness Analysis. We show that the decision version of the optimal multi-sensor data fusion problem is np-hard, even for the simple linear regression local fusion models F.

Theorem 1. *The decision version of the* optimal multi-sensor data fusion *problem is* np*-complete.*

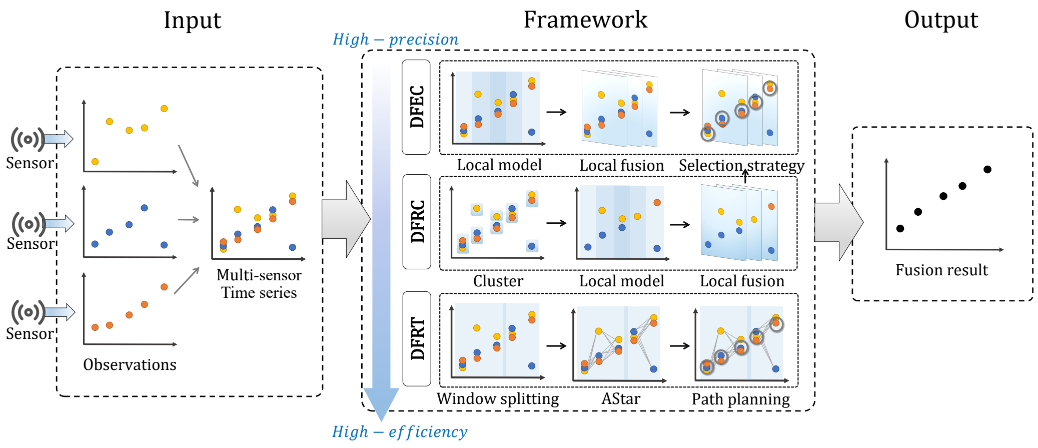

Fig. 3. An overview of our data fusion methods. Multi-sensor observations serve as inputs. Based on varying demands for precision and efficiency in different scenarios, we choose an appropriate fusion method and output the final fused time series.

Example 3 (Example 2 continued). *Consider the sensor readings X in Figure 1, with $\kappa = 3$. For the fusion result $Y = \{x_{12}, x_{21}, \ldots, x_{22,3}\}$ and local fusion models $F = \{f_1, \ldots, f_{20}\}$, let's take $\mathbf{y}_5$ as an example, $\mathbf{y}_5 = (x_{53} \quad x_{62} \quad x_{73} \quad x_{82})^\top = (5.11 \quad 5.08 \quad 5.21 \quad 5.10)^\top$. According to Formula 6, the fusion loss $\mathscr{L}(\mathbf{y}_5) = (5.11 - \boldsymbol{\phi}_5\mathbf{t}_5)^2 + (5.08 - \boldsymbol{\phi}_5\mathbf{t}_6)^2 + (5.21 - \boldsymbol{\phi}_5\mathbf{t}_7)^2 + (5.10 - \boldsymbol{\phi}_5\mathbf{t}_8)^2 = (0)^2 + (-0.04)^2 + (0.08)^2 + (-0.04)^2 = 0.0096$. Considering all the fusion values $\mathbf{y}_{20}$ w.r.t. all the local fusion models F, we obtain the fusion loss $\mathscr{L}(Y) = \sum_{i=1}^{20} \mathscr{L}(\mathbf{y}_i) = 0.076$, where $y_5 = x_{53} = 5.11$.*

3 DATA FUSION OVERVIEW

In this section, before diving into detailed explanations of our methods, we first introduce an overview to outline the end-to-end workflow of our study.

As illustrated in Figure 3, given sensor readings X collected by multiple sensors S as input, our methods compute the fusion result Y for each timestamp. Depending on the specific application scenarios, we provide three different methods, namely Data Fusion by Exact Computation (DFEC), Data Fusion with Representative Candidates (DFRC), and Data Fusion with Representative Trends (DFRT), serving varying precision and cost requirements at different levels.

DFEC: In response to the need for high precision, DFEC considers all the possible local fusion models and results across the entire time series. Subsequently, among multiple local fusion models and results, the selection strategy chooses the one leading to the minimum fusion loss in Formula 6, based on the dynamic programming.

DFRC: Rather than considering all possibilities of fusion models and results, DFRC considers representative candidates by clustering sensor observations at the same timestamp. Then the representative values for each cluster are used for the local model learning and fusion result computation procedures, thereby avoiding redundant calculations of similar observations and models. Finally, the selection strategy also chooses the representative value with the minimum fusion loss as the fusion result for each timestamp.

DFRT: For higher efficiency requirements, DFRT first divides the time series into different windows, to reduce the overlapped observations resulting from the sub-sequences considered by

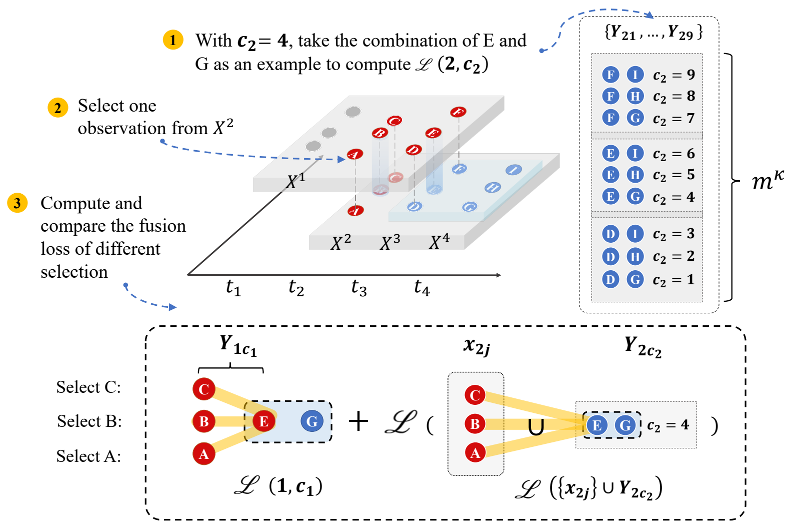

Fig. 4. The dynamic programming process for Formula 8 with $\kappa = 2$.

local fusion models. Then DFRT treats timestamps and observations within each window as a graph, where vertices correspond to sensor observations and edges correspond to consecutive selected observations on adjacent timestamps. The weight of each edge is determined by the difference between observations and the median difference of observations at respective timestamps. After that, the AStar algorithm for path planning is employed to calculate the minimum-weight path in the graph, which corresponds to the final fusion result.

4 EXACT SOLUTION

In this section, we study the exact solution of the optimal multi-sensor data fusion problem. The main steps are based on the dynamic programming, by considering the state transition of the data fusion from y_i to y_{i+1} in Section 4.1. Then the exact Algorithm 1, which ensures to return the optimal solution in pseudo-polynomial time (Proposition 2), is designed in Section 4.2.

4.1 Recursive Data Fusion

According to Formula 6, determining the fusion result of y_i requires considering κ following fusion values. Let $Y_i = \{y_{i+1}, \dots, y_{i+\kappa}\}$ denote the set of fusion results for sensor readings $\{X^{i+1}, \dots, X^{i+\kappa}\}$. As analyzed in Theorem 1, since each y_i owns m fusion candidates $\text{can}(y_i) = X^i$, we have to consider m^κ candidates $\text{can}(Y_i) = \prod_{l=1}^{\kappa} \text{can}(y_{i+l})$ to get the fusion result $Y_i \subset Y$ with the minimum loss. For instance, as shown in the top right corner of Figure 4, given $\kappa = 2$, there are $3^2 = 9$ candidates in $\text{can}(Y_1)$.

Let $Y_{ic_i} \in \text{can}(Y_i)$ denote the c-th candidate for fusing Y_i. Note that, we cannot directly determine whether Y_{ic_i} leads to the minimum loss, since Y_i is related to preceding $\{Y_{i-1}, Y_{i-2}, \dots, Y_{i-\kappa+1}\}$ and succeeding $\{Y_{i+1}, Y_{i+2}, \dots, Y_{i+\kappa-1}\}$ readings. In this sense, as illustrated in Figure 4, we maintain a current minimum loss for each Y_{ic_i} to cover m^κ candidates that may constitute Y with the minimum loss $\mathscr{L}(Y)$. Consider a possible fusion value x_{ij}, combining with the set of fusion values Y_{ic_i}, $1 \le i \le n - \kappa$, according to Formula 6, the corresponding fusion loss is

$$\mathscr{L}(\{x_{ij}\} \cup Y_{ic}) = \sum_{k=0}^{\kappa} (y_{i+k} - f_i(t_{i+k}))^2. \tag{7}$$

```
Algorithm 1: DFEC($X, \kappa$)
  Input: Multi-sensor time series $X$, local time series length $\kappa$
  Output: The loss of the optimal fusion result
  /* Recursion to get minimum loss                                    */
1 for $i \leftarrow 1$ to $n - \kappa$ do
2     for $c_i \leftarrow 1$ to $|X^i|^\kappa$ do
3         $\mathscr{L}(i, c_i) \leftarrow \infty$ ;
4         for $j \leftarrow 1$ to $|X^i|$ do
5             $Y_{i-1,c_{i-1}} \leftarrow \{x_{ij}\} \cup Y_{ic_i} \setminus \{y_{i+\kappa}\}$ ;
6             if $\mathscr{L}(i, c_i) < \mathscr{L}(i-1, c_{i-1}) + \mathscr{L}(\{x_{ij}\} \cup Y_{ic_i})$ then
7                 $\mathscr{L}(i, c_i) \leftarrow \mathscr{L}(i-1, c_{i-1}) + \mathscr{L}(\{x_{ij}\} \cup Y_{ic_i})$
8 return $min_{1 \le c_{n-\kappa} \le |X^{n-\kappa}|^\kappa} \mathscr{L}(n-\kappa, c_{n-\kappa})$
```

This gives rise to the recursive formula, where we abbreviate the minimum loss up to the i-th timestamp as

$$\mathscr{L}(i, c_i) = \min_{j \in [1,m]} \{\mathscr{L}(i-1, c_{i-1}) + \mathscr{L}(\{x_{ij}\} \cup Y_{ic_i})\}, \tag{8}$$

where c_{i-1} is the index of $Y_{i-1,c_{i-1}} = \{x_{ij}\} \cup Y_{ic_i} \setminus \{y_{i+\kappa}\}$, and $Y_{i-1,c_{i-1}} \in \text{can}(Y_{i-1})$ is obtained by moving backward one timestamp based on Y_{ic_i}. We set $\mathscr{L}(0, c_0) = 0$ for the boundary condition.

As illustrated in Figure 4, for $\mathscr{L}(1, c_1)$, we initially calculate the optimal selection on X^2 and X^3 for each of 9 fusion candidates, such that the fusion loss of $t_1 - t_3$ is minimized. Following this, we consider $\mathscr{L}(2, c_2)$ next. On X^3 and X^4, we similarly have 9 fusion candidates, as shown in the upper right corner of the figure. Let's take EG, i.e., $c_2 = 4$, as an example, and we select the optimal points on X^2 for this candidate. When making selections, in addition to considering the fusion loss with EG, we also need to take into account the corresponding $\mathscr{L}(1, c_1)$. For instance, if the candidate EG selects A, besides the fusion loss of AEG, we also need to consider $\mathscr{L}(1, c_1)$ for the candidate AE.

Example 4. *Let's take* $Y_{11} = \{x_{21}, x_{31}, x_{41}\} = \{4.77, 4.76, 4.98\}$ *in Figure 1 as an example. According to Formula 8, we first compute* $\mathscr{L}(1, c_1) = \min_{j \in [1,m]} \{\mathscr{L}(0, c_0) + \mathscr{L}(\{x_{1j}\} \cup Y_{1c_1})\}$ *for* Y_{11}. *Consider the selection* x_{11} *in* X^1, *then we have* $\mathscr{L}(1, c_1) = \mathscr{L}(\{x_{11}\} \cup Y_{1c_1}) = \mathscr{L}(\{4.5, 4.77, 4.76, 4.98\}) = \sum_{k=0}^{3}(y_{1+k} - \mathbf{t}_i \begin{pmatrix}-230.27 & 14.3\end{pmatrix}^\top)^2 = (4.5 - (-230.27 + 16.42 * 14.3))^2 + (4.77 - (-230.27 + 16.43 * 14.3))^2 + (4.76 - (-230.27 + 16.44 * 14.3))^2 + (4.98 - (-230.27 + 16.45 * 14.3))^2 = 0.014$. *Next, we replace* x_{11} *with* x_{12} *and* x_{13} *respectively, and calculate the fusion loss for each case. Finally,* x_{12} *with the minimum fusion loss* 0.0044 *is chosen as the fusion result at timestamp* t_1.

4.2 Exact Algorithm

Algorithm 1, DFEC(X, κ), Data Fusion by Exact Computation, presents the data fusion process for sensor readings X to get the optimal result Y^*. According to Formula 8, to compute the optimal fusion result, Lines 1-2 consider all the fusion values Y_i with all the candidates can(Y_i). Line 3 initializes $\mathscr{L}(i, c_i)$ with ∞, which is larger than any legal value of $\mathscr{L}(i, c_i)$. For $\mathscr{L}(i, c_i)$, Lines 4-7 choose the optimal fusion value from can(y_i) for each y_i. Specifically, in Line 5, we compute the the previous combination $Y_{i-1,c_{i-1}}$ formed by x_{ij} and the other points $\{y_{i+1}, y_{i+2}, \ldots, y_{i+\kappa-1}\}$ in Y_{ic_i}. If the condition in Line 6 is true, it implies that the choice of x_{ij} at X^i would incur a lower $\mathscr{L}(i, c_i)$. Therefore, we update $\mathscr{L}(i, c_i)$ in Line 7. Let $\mathscr{L}(n-\kappa, c_{n-\kappa})$ be the minimum loss for a specific

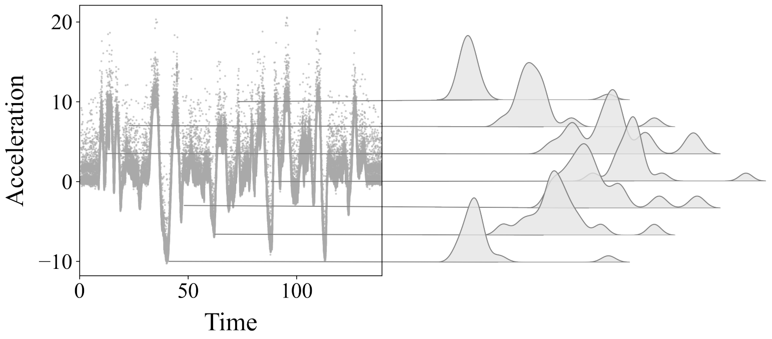

Fig. 5. IMU data distribution at different timestamps through the kernel density estimation.

$Y_{n-\kappa,c}$, then we can get the minimum loss among $Y_{n-\kappa}$ in Line 8, By retracing all $\mathscr{L}(i, c_i)$ leading to the loss, the optimal fusion result Y^* is obtained.

Proposition 2. *Algorithm 1 returns the optimal fusion result Y^* in $O(nm^\kappa\kappa)$ time with $O(nm^\kappa)$ space, where $n = |X_j|$, $m = |S|$.*

Example 5 (Example 4 continued). *Consider the sensor readings X in Figure 1, with $\kappa = 3$ and the same $\mathscr{L}(1, c_1)$ obtained in Example 4. We proceed to the next $Y_{21} = \{x_{31}, x_{41}, x_{51}\} = \{4.76, 4.98, 4.28\}$ as an example, having $Y_{1c_1} = \{x_{2j}\} \cup \{4.76, 4.98, 4.28\} \setminus \{4.28\}$ for each j in Line 5. Thus, we have $\mathscr{L}(2, c_2) = \min\{0.0044 + \mathscr{L}(\{4.77, 4.76, 4.98, 4.28\}), 0.029 + \mathscr{L}(\{4.87, 4.76, 4.98, 4.28\}), 0.016 + \mathscr{L}(\{4.81, 4.76, 4.98, 4.28\})\} = \min\{0.1944, 0.199, 0.196\} = 0.1944$. Still, we should consider $\mathscr{L}(3, c_3)$ for all c_3 and corresponding Y_{3c_3} in Line 2. Then, in Lines 4-7, we compare and update the selection with the minimum fusion loss for every candidate. After repeating the aforesaid process, we obtain the final result $\min \mathscr{L}(20, c_{20}) = 0.076$, with $y_5 = x_{53} = 5.11$.*

5 APPROXIMATION ALGORITHM

According to Proposition 2, exhaustively considering all candidates $\text{can}(y_i)$ to obtain the exact fusion result is proven to be computationally expensive (in $O(nm^\kappa\kappa)$ time). Therefore, it is intuitive to further design the approximation strategy to reduce the search space for fusing Y_i. Consequently, an approximation algorithm is formulated based on representative candidates, ensuring the polynomial time complexity (Proposition 3), and accompanied by performance guarantees (Proposition 5) as detailed in Section 5.1. Furthermore, to enhance efficiency and constrain the exponent of the term m to a constant value within the complexity, an additional approximation algorithm is introduced in Section 5.2, targeting the reduction of complexity in local fusion models f_i. The main idea is to accelerate the data fusion process by reducing similar data changes and training models based on representative trends.

5.1 Data Fusion with Representative Candidates

As explained in Section 1, outliers occasionally occur in low-precision sensors, and most observations X^i in the same time are usually close to each other. For instance, as shown in Figure 5, the sensor readings at the same timestamp can naturally be clustered into several different clusters based on the distribution of their values. Inspired by this, rather than considering all the candidates $\text{can}(y_i)$ for y_i, we use the clustered data to reduce the scale of candidates. Specifically, we perform clustering on each X^i and obtain the cluster set $U_i = \{U_{i1}, U_{i2}, \ldots, U_{i\lambda_i}\}$. For instance, can be K-Means [36], K-Median [23] or DBSCAN[16]. Then we select the observation closest to the cluster center C_{ik} as the representative candidate x'_{ik} of each cluster $U_{ik} \in U_i$, having

Algorithm 2: DFRC(X, κ)

```
  Input: Multi-sensor time series $X$, local time series length $\kappa$, the number of clusters $\lambda$
  Output: The loss of the approximate fusion result $Y$
1 for each $X^i \in X$ do
2     $U_i \leftarrow$ Cluster($X^i$) ;
3     for each $U_{ik} \in U_i$ do
4         $x'_{ik} = \arg\min_{x_{ij} \in U_{ik}} |x_{ij} - C_{ik}|$ ;
5     $X' \leftarrow X' \cup \{x'_{ik}\}$;
6 return DFEC($X', \kappa$)
```

$$x'_{ik} = \arg\min_{x_{ij} \in U_{ik}} |x_{ij} - C_{ik}|. \tag{9}$$

Algorithm 2, DFRC(X, κ), Data Fusion with Representative Candidates, is then designed to get fusion results with representative candidates over clustered data. As discussed above, Lines 2-4 compute the representation candidates x'_{ik} for observations X^i at each timestamp t_i, to construct X' in Line 5. Finally, the data fusion result Y can be obtained by calling DFEC (X', κ).

Proposition 3. *Algorithm 2 returns an approximate fusion result in $O(n\lambda^\kappa \kappa)$ time with $O(n\lambda^\kappa)$ space, where $n = |X_j|$, $m = |S|$, λ is the maximum number of clusters.*

Next, we study the performance guarantee for Algorithm 2. First, we explore the bound for the optimal estimation of the local fusion model f_i under the equidistant timestamp condition. Let's denote the absolute value of the residual between the optimal solution $y_i^* \in \mathbf{y}_i^*$ and the estimation $f_i(t_i)$ as $d = \max_{y_i^* \in \mathbf{y}_i^*} |y_i^* - f_i(t_i)|$.

Proposition 4. *For any sequence $\{X^i, \dots, X^{i+\kappa}\} \subset X$, we have*

$$d \leq \frac{7}{4} D_i,$$

where $D_i = \max_{\substack{i \leq i_1, i_2 \leq i+\kappa \\ 1 \leq j_1, j_2 \leq m}} |x_{i_1 j_1} - x_{i_2 j_2}|$, $i \in [1, n-\kappa]$ is the maximum distance between any two observations within $\{X^i, \dots, X^{i+\kappa}\}$.

Let L^* and L denote the fusion loss of the optimal solution Y^* by Algorithm 1 and the approximation result Y by Algorithm 2 respectively. We realize that the gap between Y^* and Y mainly arises from the clustering procedure. Combining with the error bound introduced by the fusion estimation of f_i in Proposition 4, we obtain the approximation guarantee of DFRC Algorithm 2.

Proposition 5. *Algorithm 2 returns an approximate fusion result Y having*

$$L - L^* \leq (\kappa + 1)(n - \kappa)\left(\frac{7}{2} D\eta + \eta^2\right),$$

where $\eta = \max_{\substack{1 \leq i \leq n \\ 1 \leq k \leq \lambda_i}} \max_{x_{ik_1}, x_{ik_2} \in U_{ik}} |x_{ik_1} - x_{ik_2}|$ is the maximum distance between any two observations within each cluster, and $D = \max_{1 \leq i \leq n-\kappa} D_i$ is the maximum D_i among all the sequences $\{X^i, \dots, X^{i+\kappa}\} \subset X$.

Example 6 (Example 5 continued). *Consider the sensor data in Figure 1 with $\kappa = 3$. As an example, for timestamp t_1, we cluster $X^1 = \{x_{11}, x_{12}, x_{13}\} = \{4.5, 4.57, 4.57\}$ and obtain $U_1 = \{U_{11}, U_{12}\}$ in*

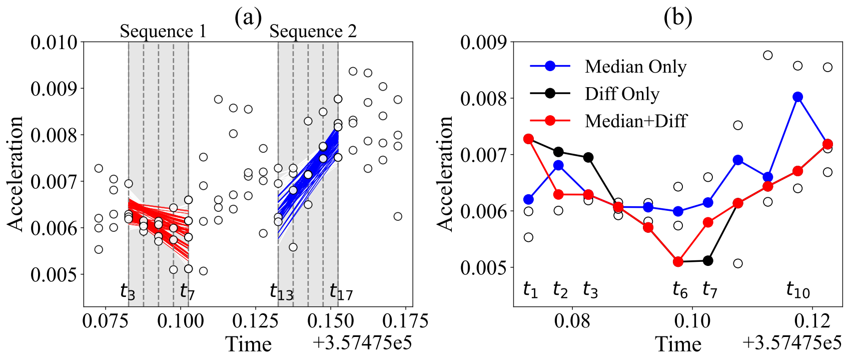

Fig. 6. (a) Different data changing trends with corresponding local fusion models f_i within different sequences, and (b) fusion results with different edge weight settings over GINS.

Line 2, where $U_{11} = \{4.5\}$ with $C_{11} = 4.5$ and $U_{12} = \{4.57, 4.57\}$ with $C_{12} = 4.57$. Then, according to Formula 9, we choose $x_{11} \in X^1$ as the representative candidate x'_{11}, and $x'_{12} = 4.57$ from U_{12}. Similarly, we could obtain X' by considering each $X^i \in X$ in Line 1, and perform the process based on X' as described in Example 5, leading to the final result $\mathscr{L}(20, c_{20}) = 0.076$ with $y_5 = 4.98$.

5.2 Data Fusion with Representative Trends

Considering that spending $O(\lambda^\kappa)$ time using representative candidates of clusters may be costly for a large λ or κ, we study further approximation to increase the efficiency. Based on the observation that some trends in sequence $\{X^i, \ldots, X^{i+\kappa}\} \subset X$ are similar, we design the heuristics to reduce the consideration of similar data changes and models by training f_i with representative trends.

To handle various trends of data changes within X, we devise local fusion models f_i to capture different trends within different sequences $\{X^i, \ldots, X^{i+\kappa}\} \subset X$. Moreover, Figure 6 shows some observations from multiple sensors over the GINS dataset, where each point denotes an observation. Each solid line (marked in red or blue) in Figure 6(a) depicts a local fusion model based on distinct observations at timestamps t_3-t_7 and t_{13}-t_{17}. As illustrated, the changing trend is similar within the same sequence 1 (t_3-t_7) or sequence 2 (t_{13}-t_{17}), but may be different in different sequences, e.g., the downward trend among t_3-t_7 and the upward trend among t_{13}-t_{17} in Figure 6(a). Therefore, due to the similar issue about the local setting as mentioned in Section 2.1 and Figure 2(a), we further partition X into different windows of size $\kappa + 1$, thus creating a collection of windows $\mathbf{W} = \{W_1, W_2, \ldots, W_{\lceil \frac{n}{\kappa+1} \rceil}\}$, where $W_i = \{X^{\kappa i-\kappa+1}, \ldots, X^{\kappa i}\}$ denotes the i-th window in X.

Now, with the consideration of representative trends, let's regard the sensor readings at the first timestamp $t_{\kappa i}$ and the last $t_{\kappa i+\kappa}$ as the start and end points. Then the orientation of the end point relative to the start point just indicates the trend of $\{X^{\kappa i-\kappa+1}, \ldots, X^{\kappa i}\}$. Thus, the problem becomes that, given W_i, how to reduce similar trends within it and find $\{y_{\kappa i-\kappa+1}, \ldots, y_{\kappa i}\}$ with the minimum sum of distances from the start point to the end point based on the remaining representative trends. Intuitively, we consider relating W_i to a $(\kappa + 1)$-partite sub-graph $\mathbf{G}_i = (\mathbf{P}_i, \mathbf{E}_i)$ that consists of κ partitions of a point set $\mathbf{P}_i = \{P_{\kappa i-\kappa+1}, \ldots, P_{\kappa i}\}$ and an edge set $\mathbf{E}_i$. Sensor readings X^i at each timestamp t_i correspond to $P_i = \{p_{i1}, \ldots, p_{im}\}$ and each observation $x_{ij} \in X^i$ corresponds to $p_{ij} \in P_i$. Furthermore, the sequential process from $g(X^i)$ to $g(X^{i+1})$ corresponds to an edge $(p_{i,j}, p_{i+1,j'})$, $1 \le j, j' \le m$. By expanding this process to all timestamps t_i, we construct

$$\mathbf{E}_i = \{(P_i, P_{i+1}) \mid P_i, P_{i+1} \in \mathbf{P}_i\}. \tag{10}$$

```
Algorithm 3: DFRT($X, \kappa$)
   Input: Multi-sensor time series $X$, local series length $\kappa$
   Output: The approximate fusion result $Y$
 1 for each $W_i \in \mathbf{W}$ do
       /* Create the graph $\mathbf{G}_i$ based on $W_i$                    */
 2     $\mathbf{G}_i.\mathbf{P}_i \leftarrow \{P_{\kappa i-\kappa+1}, \dots, P_{\kappa i+i}\}$;
 3     $\mathbf{G}_i.\mathbf{E}_i \leftarrow \{(P_i, P_{i+1}) \mid P_i, P_{i+1} \in \mathbf{P}_i\}$ ;
 4     forall $(p_{i,j}, p_{i+1,j'}) \in \mathbf{E}_i$ do $\omega(p_{i,j}, p_{i+1,j'}) \leftarrow |x_{i,j} - x_{i+1,j'}| + |x_{i+1,j'} - M_{\kappa i+1}|$;
       /* Testing different start and end points pairs                    */
 5     for $j_1 \leftarrow 1$ to $m$ do
 6         for $j_2 \leftarrow 1$ to $m$ do
 7             $\mathbf{G}_i.p_0 \leftarrow p_{\kappa i-\kappa+1,j_1}$ ;
 8             $\mathbf{G}_i.p_{\kappa+2} \leftarrow p_{\kappa i,j_2}$ ;
 9             $\tau_{(j_2-1)m+j_1} \leftarrow$ AStar($\mathbf{G}_i$) ;
10             if $H(\tau_j) < H(T)$ then
11                 $T \leftarrow \tau_{(j_2-1)m+j_1}$ ;
12     forall $p_{kj} \in T$ do $y_{\kappa i-\kappa+k} \leftarrow x_{kj}$;
13 return $Y$;
```

According to $\mathbf{G}_i$, we realize that we can transform the task of finding $\{y_{\kappa i-\kappa+1}, \dots, y_{\kappa i}\}$ with the minimum sum of distances between sensor readings at adjacent timestamps into finding a path $T = \{p_{\kappa i-\kappa+1,j}, \dots, p_{\kappa i,j_{i+\kappa}}\}$ with the minimum edge weight. Based on the target of this problem, we employ AStar algorithm [21] to obtain T on $\mathbf{G}_i$. Specifically, we provide each edge $(p_{i,j}, p_{i+1,j'}) \in \mathbf{E}_i$ with a weight consisting of the distance between x_{i,j_1} and x_{i+1,j_2} as well as the distance between p_{i+1,j_2} and the median of X^{i+1},

$$\omega(p_{i,j}, p_{i+1,j'}) = |x_{i,j} - x_{i+1,j'}| + |x_{i+1,j'} - M_{i+1}|, \tag{11}$$

where M_{i+1} is the median of X^{i+1}. As shown in Figure 6(b), the fusion result represented by the black line is obtained when Formula 11 only includes the first distance term $|x_{i,j} - x_{i+1,j'}|$ and the blue line is derived when Formula 11 only includes the second median term $|x_{i+1,j'} - M_{i+1}|$, as well as the red line incorporating both of two terms. We can observe that the absence of the median in Formula 11 may lead to an overly smooth trend during fusion, such as the fusion results at timestamps $t_1 - t_3$ and $t_6 - t_7$. While the absence of difference may result in a degradation to the median, making it more susceptible to the influence of outliers, as seen in the fusion result at t_{10}.

Example 7. *Consider again the data in Figure 1 with $\kappa = 3$. First, we divide the observations X into $\lceil \frac{n}{\kappa+1} \rceil = \lceil \frac{20}{4} \rceil = 5$ windows, i.e., $\mathbf{W} = \{W_1, W_2, W_3, W_4, W_5\}$. Taking $W_1 = \{X^1, X^2, X^3, X^4\}$ as an example, we create the graph $\mathbf{G}_1 = (\mathbf{P}_1, \mathbf{E}_1)$ based on W_1, where $\mathbf{P}_1 = \{P_1, \dots, P_4\}$ corresponds to $\{X^1, \dots, X^4\}$ and $\mathbf{E}_1 = \{(P_1, P_2)\} \cup \{(P_2, P_3)\} \cup \{(P_3, P_4)\}$ according to Formula 10. According to Formula 11, we take (p_{11}, p_{21}) as an example for the edge weight, which has $|4.5 - 4.77| + |4.77 - 4.81| = 0.31$. Then, we calculate and set weights for all edges in E_1.*

Algorithm 3, DFRT(X, κ), Data Fusion with Representative Trends, presents the computation based on the heuristics of training f_i with representative trends. Line 1 repeats the process as follows for all $W_i \in \mathbf{W}$. In Lines 2-4, we generate the corresponding sub-graph $\mathbf{G}_i = (\mathbf{P}_i, \mathbf{E}_i)$ for W_i, which requires $O(\kappa m^2)$ and $O(\kappa m^2)$ space to store edge weights and different paths respectively. Due to the influence of start and end points pairs on the trend of the shortest path, Lines 5-12 compute with different m pairs. By considering the loops in Lines 1 and 5, it results in $O(\lceil \frac{n}{\kappa+1} \rceil) \times O(m^2)$ complexity. Lines 7-8 introduce two dummy vertices p_0 and $p_{\kappa+2}$ to represent the start and end points respectively of $\mathbf{G}_i$. The points are considered to be placed at the far left and far right edges of W_i, i.e., $p_0 \in P_0$ and $p_{\kappa+2} \in P_{\kappa+2}$. When calculating the edge weights, they are sequentially regarded as $p_{\kappa i-\kappa+1,j_1} \in P_{\kappa i-\kappa+1}$ and $p_{\kappa i,j_2} \in P_{\kappa i}$, $1 \leq j_1, j_2 \leq m$. Line 9 calls AStar($\mathbf{G}_i$) to obtain the shortest path T on $\mathbf{G}_i$, each time leading to $O((m\kappa + 1)(1 + m^2)log((\kappa + 1)m))$ complexity and $O(wm)$ space. Let τ_j denote the shortest path obtained based on the j-th pair, $H(\tau_j)$ as the total path edge weight of τ_j. Then Lines 10-11 select τ_j with the overall minimum cost as the best shortest path T of W_i. All the points in T just represent the fusion results within W_i. Finally, in Line 12, we get the fusion results $\{y_{\kappa i-\kappa+i}, \ldots, y_{\kappa i+i}\}$ based on T from timestamp $t_{\kappa i-\kappa+1}$ to $t_{\kappa i}$ one by one. Therefore, Algorithm 3 runs in time $O(\lceil \frac{n}{\kappa+1} \rceil m^2(\kappa m + \kappa m^2)log(\kappa m))$, i.e., $O(nm^4 log(\kappa m))$, with $O(\kappa m^2)$ space.

Example 8 (Example 7 continued). *Consider observations X in Figure 1 with the same $\mathbf{W}$ in Example 7. Let's take into account τ_1 with start and end points $p_0 = (16.41, 4.5), p_5 = (16.46, 4.28)$ in Lines 7-8, and we have $\tau_1 = \{p_0, p_{13}, p_{22}, p_{33}, p_{41}, p_5\}$ with the path $H(\tau_1) = 0 + 0.31 + 0.03 + 0.13 + 0.87 = 1.34$. Similarly, we can obtain $\tau_2 = \{p_0, p_{13}, p_{22}, p_{33}, p_{41}, p_5\}$ with the path $H(\tau_2) = 0 + 0.24 + 0.03 + 0.13 + 0.87 = 1.27$. For each W_i, we maintain τ_j with the smallest $H(\tau_j)$ in Line 11. Then, the points $p_{kj} \in T$ are the fusion results for readings X^1, X^2, X^3, X^4. For example, following Line 12, if $T = \tau_2$, we will obtain $y_1 = 4.57, y_2 = 4.87, y_3 = 5.01, y_4 = 4.98$.*

6 EXPERIMENT

In this section, we evaluate our methods with the following objectives:

1) We extensively measure our methods on a variety of real datasets with different sizes, sensor numbers, noise levels, inconsistency degree, uneven distributions, missing rates and the online setting. In all these scenarios, our methods consistently outperform other existing methods.

2) We evaluate the fusion performance of our methods, including the ablation study, approximation ratio, theoretical and experimental parameter determination, and clustering methods.

3) We validate the fusion results of our methods in the integration navigation application.

Our experimental highlights are: (i) our methods achieve superiority in real datasets with various characteristics; (ii) our methods perform best in the downstream application, demonstrating their practical values. The source code and data are available online.[4]

6.1 Experimental Settings

All programs are implemented in Python and experiments are performed on a machine with AMD Ryzen Mobile 4800H CPU, 16 GB Memory.

6.1.1 Datasets. We employ three real datasets and one synthetic dataset in evaluation.

IMU [40] consists of the observations from an array of 16 IMU (Inertial Measurement Units). Ground truths are provided by higher-precision gyroscopes. The observations may contain different types of outliers, including noise, sensor faults, and so on, owing to various factors such as sensor precision limitations, calibration issues, and environmental conditions. Each sequence of IMU observations contains at least 138,775 timestamps.

GPS is manually collected by carrying 4 GPS devices deployed in mobile phones and walking around the campus to collect the positional information. Since we know exactly the trajectory, the truths of the data fusion result are manually labeled. For each sensor, there are 1,264 observations.

[4]https://github.com/erssss/TDFMLPS

GINS [50] provides the incremental angle and velocity of an object in motion from 4 different levels of sensors, with more than 331,077 observations in each sequence. Since it has no ground truth, we use it to test the performance of data fusion algorithms in downstream applications in Section 6.4. In the GINS dataset, there are integration drift, angle drift, noise, and sensor faults, and more, owing to object vibrations, environmental disturbances, and hardware precision limitations.

WEATHER comprises daily temperature forecasts for both high and low temperatures in Los Angeles from September 1, 2022, to July 6, 2023. We gather this data from 3 different weather forecasting platforms AerisWeather[5], WorldWeatherOnline[6], and WeatherUnderground[7]. We also collect true temperature to assess the quality of the fusion process. Each sensor contains more than 310 timestamps, leading to 1,856 temperature values in total. Notably, different from the aforesaid datasets, WEATHER contains generally clean data, with no obvious outlier. We use this dataset to validate whether our methods can perform well in noise-free scenarios.

MOVE is generated following the same line of existing studies [43, 64], by a random acceleration process through physical models to simulate real-world motion scenarios. Specifically, the random walk model holds $a_i = a_{i-1} + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$, $x_{(i)} = x_{(i-1)} + a_{i-1}\Delta t^2$, where a_i is the acceleration, $x_{(i)}$ is the generated truth of the displacement at t_i, and Δt is the time interval. The observation error distribution of different sources is modeled by Gaussian noise with different variances [33, 62]. It contains no fewer than 2,500 timestamps and 4 sensors.

6.1.2 Criteria. To evaluate the data fusion accuracy, we compare the fusion result Y to the ground truth Y^* using the MSE (mean square error), $MSE = \frac{\sum_{i=1}^{n}(y_i - y_i^*)^2}{n}$. In the downstream integration navigation task, we fuse the observations from four sensors in the GINS dataset, and then navigate based on the fusion results by an integrated navigation system [50]. Finally, we compare the navigation result by MSE with the positioning information observed by a high-precision device.

6.1.3 Competing Methods. We compare our methods against various existing approaches. Please see Section 7 for explanations on categorizing these nine competing methods.

(1) **Kalman Filter** [22, 59] takes both predictions and observations into consideration to estimate the fusion value, which consists of two steps usually, i.e., prediction and update.

(2) **Sums**[8] [28] treats sensors as web pages and analyzes the mutual influence among sensors and their observations.

(3) **TruthFinder**[8] [65] treats sensor observations as claims and estimates the existence probability of each observation, then evaluate the reliability of different sensors by Bayesian analysis.

(4) **Investment**[8] [42] leverages the concept of investment, where each sensor invests its reliability in its own observations, and updates the credibility of sensors based on the estimations in turn.

(5) **GTM** [68] is a Bayesian probabilistic method tailored for continuous data conflict resolution, i.e., addressing the issue of inconsistent sensor observations.

(6) **CRH**[9] [32] estimates the reliability of sensors and confidence of their observations by minimizing the overall weighted deviation between observations and the fusion values to obtain fusion values.

(7) **CATD**[9] [31] also establishes a relationship between sensor reliability and confidence in their observations, to handle the long-tail phenomenon.

(8) **Dyna** [33] is an incremental truth discovery framework based on streaming data, capable of updating source weights and providing estimations upon the arrival of new data.

[5]https://www.aerisweather.com
[6]https://www.worldweatheronline.com
[7]https://www.wunderground.com
[8]https://github.com/joesingo/truthdiscovery
[9]https://sites.google.com/iastate.edu/qili

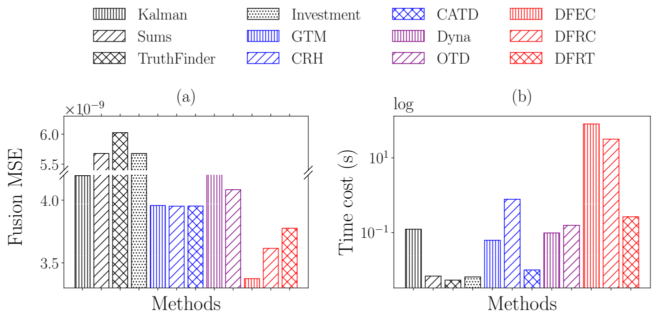

Fig. 7. Data fusion performance of various methods over GPS.

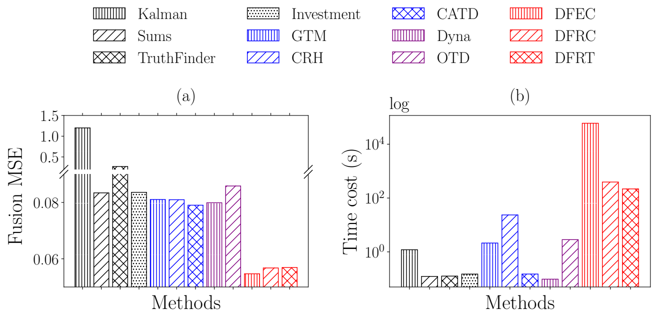

Fig. 8. Data fusion performance of various methods over IMU.

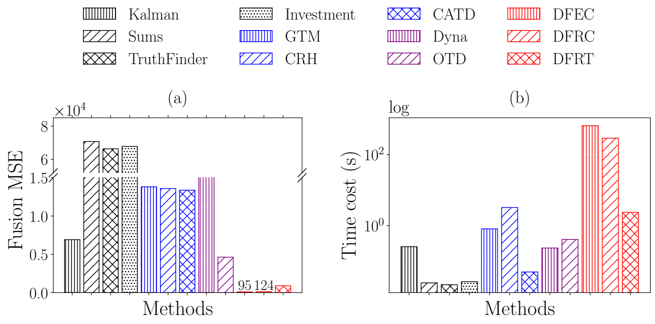

Fig. 9. Data fusion performance of various methods over MOVE.

(9) **OTD** [62] is also a truth discovery framework primarily designed for the online setting, by incorporating prediction values into the truth estimation.

6.2 Comparison with Existing Techniques

Figures 7-10 report the performance of fusing multiple sensor readings over the datasets with known truths as introduced in Section 6.1.1. Since Kalman filter obtains fusion values by combining sensor observations and model predictions, the estimation can be affected when there are outliers. It is thus not surprising that Kalman filter performs poorly over IMU and MOVE datasets as shown in Figures 8, 9. Besides, Kalman filter assumes that data change continuously over time, and exhibits the lag in the estimation value when dealing with rapidly changing observations. This is another reason for its suboptimal performance over IMU, which collects high-frequency changes.

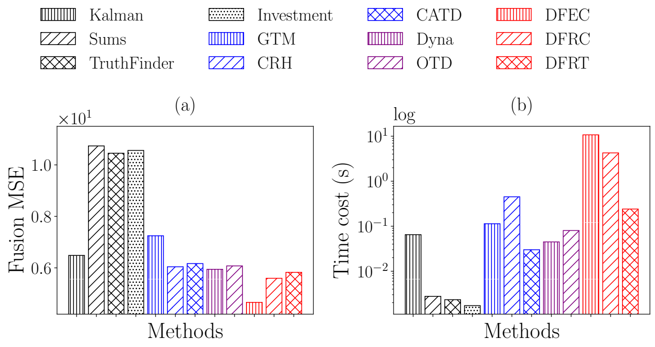

Fig. 10. Data fusion performance of various methods over WEATHER.

The truth discovery based Sums, TruthFinder, and Investment perform poorly across all the datasets in Figures 7-10, since they are primarily designed for discrete data. Moreover, the other truth discovery approaches GTM, CRH, and CATD are designed to handle continuous (or heterogeneous) data and can obtain intermediate fusion values through a weighted approach. Therefore, they achieve generally better results compared with the aforesaid methods. However, the limited number of sensors restricts the estimation of sensor reliability, making they cannot achieve comparable performance with our methods even when dealing with relatively low levels of noise as shown in Figure 10. Although CATD takes into account the long-tail phenomenon, lacking the consideration of local features within varying time series like our local fusion model f_i makes it fail to capture precise temporal trends, e.g., the sudden changes. As for the truth discovery frameworks designed for the online setting, Dyna is based on the smoothing assumption, while OTD calculates the global sensor weights, thus neither of them can effectively handle outliers in sensor readings.

In contrast, our methods DFEC, DFRC, and DFRT achieve respectively an average improvement in fusion accuracy of 53.7%, 48.9%, and 46.1%, compared to baselines across all datasets (with outliers or not) in Figures 7-10. The results demonstrate the contribution of using the selection strategy to maximize the likelihood during fusing multiple low-precision sensor readings w.r.t. local fusion models. Among our methods, DFRT is more efficient than DFEC and DFRC, but DFEC and DFRC return better fusion results than DFRT in most cases, which verifies the rationale of solving Problem 1 exactly by DFEC and the approximation guarantee in Proposition 5 by DFRC. Although DFEC and DFRC achieve superiority over baselines with higher time costs, DFRT shows a comparable time cost to most baselines, with better fusion results. Therefore, we suggest DFEC and DFRC for small datasets (e.g., GPS, MOVE, and WEATHER) to ensure high accuracy, while DFRT is preferred for large datasets (e.g., IMU and GINS) to balance effectiveness and efficiency.

Note that, although unobserved truth indeed puts forward a strict challenge to our selection strategy based on observations, we could also provide any unobserved values as candidates, e.g., predicted values by existing studies or local fusion models, for our methods. Actually, all the datasets (GPS, IMU, WEATHER, MOVE) used for evaluating the data fusion performance in Figures 7-10 contain the unobserved true value, where our methods still achieve the best fusion performance only based on available observations. This is attributed to the ability to mitigate the influence of outliers, thereby yielding results that are closer to the true values through the effective fusion between observations.

In addition, we also study the experimental evaluation for different methods under various settings over MOVE. Figure 11 reports the results with various numbers of timestamps. As shown, the fusion accuracy is generally stable, with consistent data distribution. Observing Figure 12, we notice that as the number of sensors increases, all methods exhibit varying degrees of performance

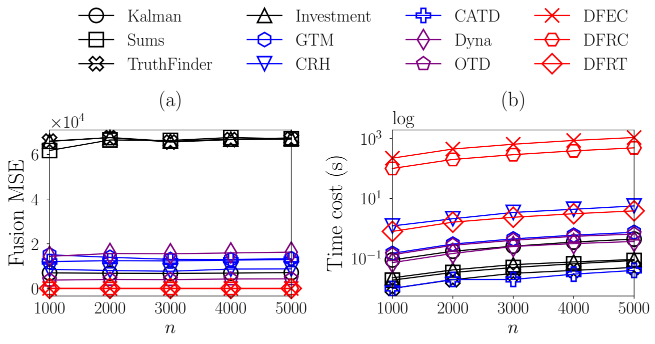

Fig. 11. Data fusion performance for varying number of timestamps n over MOVE, with 20% outlier rate, $m = 5$.

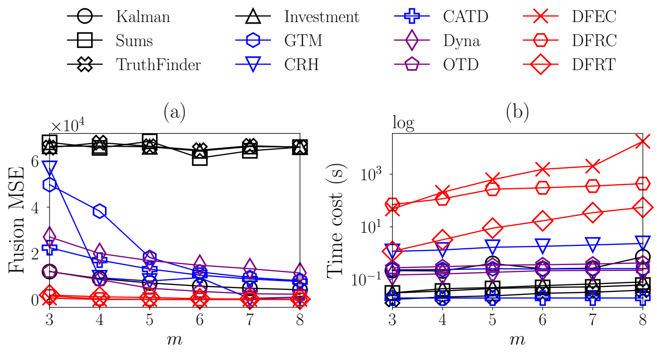

Fig. 12. Data fusion performance for varying the number of sensors m over MOVE, with 20% outlier rate, $n = 2500$.

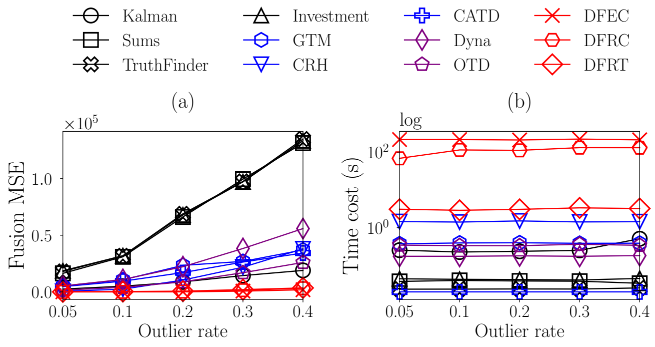

Fig. 13. Data fusion performance for varying the outlier rate over MOVE, with $m = 4$, $n = 2500$.

improvement. This indicates that increasing the number of low-precision sensors used in practical applications could lead to better accuracy and robustness. Furthermore, it is not surprising that most truth discovery methods exhibit subpar performance in light of the limited number of sensors, rendering them more susceptible to outliers from highly weighted sensors. Figure 13 shows that the fusion accuracy of most competing methods drops markedly with the increase of outliers. However, our methods show robustness and exhibit significantly better tolerance to outliers, thanks to the selection strategy with likelihood maximization.

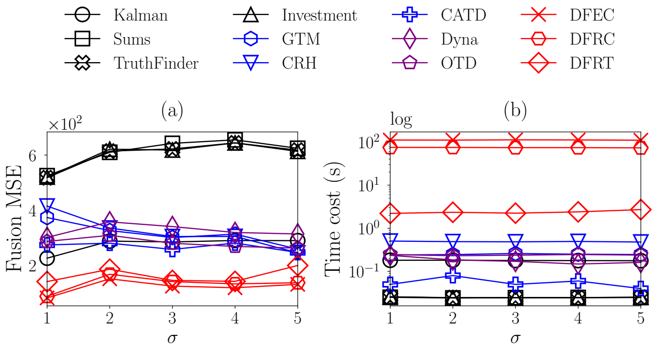

Fig. 14. Data fusion performance for varying the standard deviation σ over MOVE, with 20% outlier rate, $m = 4$, $n = 2500$.

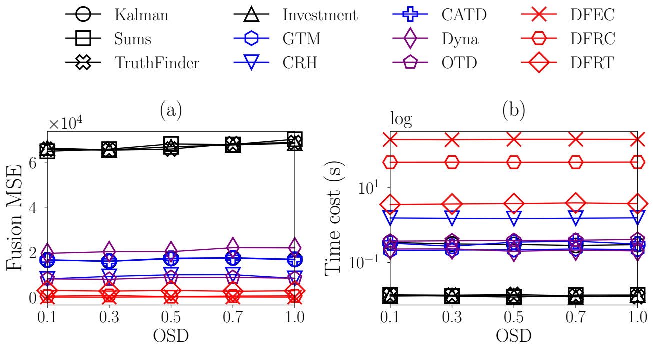

Fig. 15. Data Fusion performance for varying offset standard deviation (OSD) over MOVE, with 20% outlier rate, $m = 4$, $n = 2500$.

Although inconsistent timestamps bring difficulties for data fusion, our methods can be complementary with alignment methods to handle them. To experimentally explore such a scenario, we generate the observations with inconsistent timestamps based on the MOVE dataset, where observation intervals follow a normal distribution with a given observation frequency as the mean value and varying standard deviation σ. A higher σ indicates a more pronounced misalignment/asynchronous phenomenon among observations. Figure 14 reports the experiment results, where we first employ the alignment method SAMC [17] to align inconsistent timestamps, and the aligned results are then applied to data fusion by various methods. As σ increases, we find that the fusion performance will not be significantly affected, indicating that our methods do not necessarily rely on the assumption of timestamp consistency.

Our methods can inherently handle unevenly distributed timestamps since local fusion models allow computing irregular timestamps in Formula 1. To experimentally investigate the performance of our algorithms with unevenly distributed timestamps, we generate unevenly spaced timestamps with varying offset standard deviations (OSD) [30, 52] into the MOVE dataset. Specifically, $OSD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(t_{i+1}-t_i)^2}$ describes the stability level of sensor observation frequency. As shown in Figure 15, our methods are not sensitive to unevenly distributed timestamps and consistently achieve the best fusion result.

What's more, our methods can naturally accommodate sensors with missing values, which corresponds to reducing the fusion candidates in Formula 3. To assess the performance of our

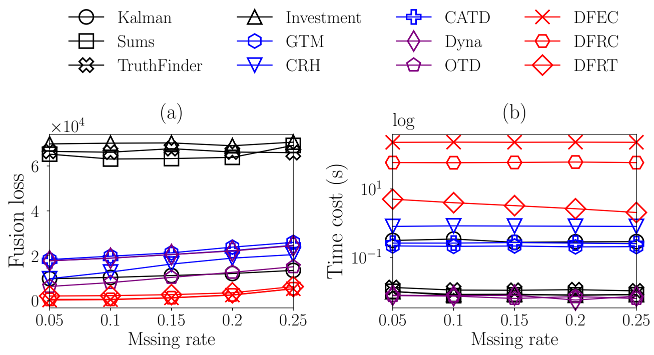

Fig. 16. Data Fusion loss for varying missing rate over MOVE, with 20% outlier rate, $m = 4$, $n = 2500$.

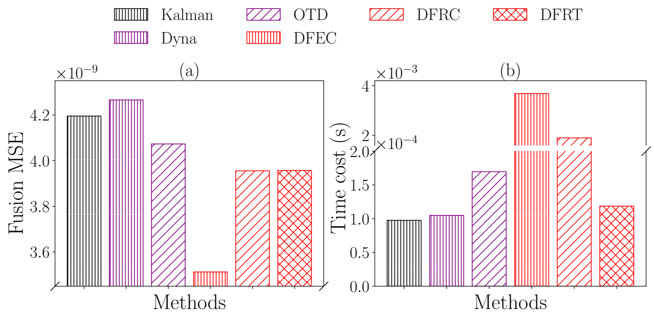

Fig. 17. Data fusion performance in the online setting of various methods over GPS.

methods in such scenarios, we inject missing values into the MOVE dataset with different missing rates and conduct the experiments in Figure 16. As shown, our methods still achieve the best result with missing values, against various competing methods. Moreover, as the missing rate increases, we observe varying degrees of performance degradation in different approaches, while our methods consistently maintain superior performance with relative stability.

Actually, our methods are applicable to real-time processing scenarios. (1) Specifically, for DFEC, leveraging the characteristics of the dynamic programming, we can naturally implement the state transitions based on the observations X^i at the current timestamp t_i in Line 1 in Algorithm 1. For the update at the i-th timestamp, according to Formula 8, we only need to calculate $\mathscr{L}(i, c_i)$ based on the $\mathscr{L}(i-1, c_{i-1})$ obtained from the previous timestamp in Lines 6-7. (2) Since the main distinction between DFRC and DFEC is that DFRC involves an extra clustering step (Line 2 in Algorithm 2), DFRC can naturally execute state transitions based on the latest observations X^i (Line 1 in Algorithm 1). In contrast to the original Algorithm 2, where clustering is conducted for all timestamps before data fusion, the online version sequentially performs clustering for each timestamp. (3) As for DFRT, to promptly provide fusion results for the latest timestamp, we no longer partition the observations into separate windows as Algorithm 3. Instead, we establish a window on the observations $X^{i-\kappa+1}$ to X^i, which differs from Lines 1-3 in Algorithm 3. Then, we perform AStar algorithm on this new window and take the last vertex on the optimal path obtained by AStar as the fusion result for the current timestamp in Line 9.

To further test the practicality of our methods, we conduct experiments under the online streaming setting in Figure 17 over GPS (collected by carrying mobile devices), with the online version of our methods (whose explanations are provided above). As shown, our methods also perform better than competing methods in the online setting in Figure 17(a). Figure 17(b) shows the average time

Table 2. Ablation study (fusion MSE).

Datasets	DFEC	DFRC	DFRT	Prediction	Global
GPS	3.3741e-09	3.6155e-09	3.7762-09	3.3943e-09	8.6394e-07
IMU	0.0547	0.0568	0.0569	0.0651	14.817
WEATHER	4.6570	5.5955	5.8252	11.324	34.313
MOVE	95.033	123.64	894.16	132.31	7.7834e+08

Table 3. Data fusion loss over varying datasets.

Datasets	DFEC	DFRC	Difference	Ratio
GPS	3.58E-08	8.76E-08	5.18E-08	2.45
IMU	19.98	29.32	9.34	1.47
WEATHER	1207	1725.3	518.3	1.43
MOVE	5319.1	8421.3	3102.1	1.58

cost of various methods to fuse data for each timestamp. The results indicate that our DFRT processes each fusion result quickly, at a millisecond level (comparable with specified online methods Dyna and OTD), with higher data fusion accuracy. Such results make it suitable for applications that require millisecond-level responses, such as mobile devices [13], real-time navigation [37], VR [6], and similar domains.

6.3 Performance of Our Methods

To elucidate the effectiveness of the components in our methods, i.e., the selection strategy and local fusion models, we conduct the ablation study over all datasets with available truths in Table 2, where we measure the fusion MSE over varying datasets with different variants. The prediction variant takes prediction values by local fusion models as the fusion result, instead of selecting one of the observations. The global variant builds a global fusion model based on the entire temporal sequence, instead of establishing a series of local fusion models. As shown, the prediction variant exhibits performance inferior to DFEC because of its weak noise resistance. Due to the presence of noise in sensor data, using predictions as fusion results directly might introduce additional uncertainty based on noise. On the contrary, the selection strategy chooses the most reasonable value from sensor observations, avoiding potential situations where predictions could deviate significantly from observations. Moreover, the global variant performs poorly due to its inability to capture fine-grained data changes, yielding only coarse fusion results. Such results verify the effectiveness of our designs about the selection strategy and local fusion models.

To assess the significance of the approximation result theorem, we conduct experiments to quantify the actual approximation ratio, as presented in Table 3. Our experiments involve measuring the fusion loss (as defined in Formula 6) of both the exact algorithm DFEC and the approximation algorithm DFRC, alongside comparing their differences and approximation ratios across all datasets used for evaluating data fusion performance in Section 6.2. Notably, we observe that the approximation ratio of DFRC typically remains tight (<2.5), indicating its ability to yield solutions closely akin to those of DFEC. This observation demonstrates the practical implications and significance of the approximation guarantee provided in Proposition 5.

Additionally, we conduct experiments to evaluate the performance of different methods in this study with various parameter settings as shown in Figure 18(a), which presents the results over

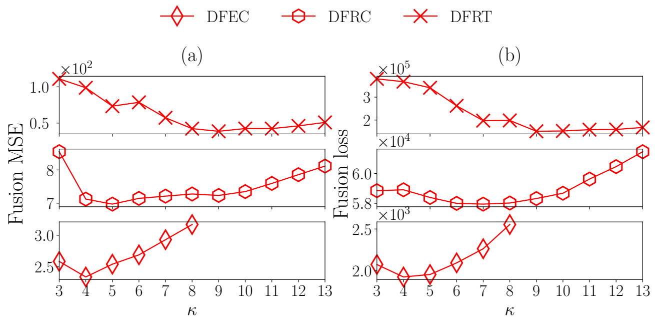

Fig. 18. Data fusion performance for varying the length of local fusion model κ over MOVE, with 20% outlier rate, $m = 4, n = 1000$.

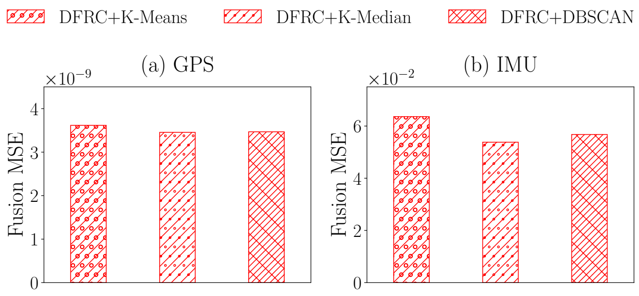

Fig. 19. Data fusion performance for varying the clustering method performed in DFRC over IMU.

various κ. As Figure 18(a) shown, the fusion MSE generally exhibits a trend of decreasing firstly and then increasing as κ increases. When κ is too small, few adjacent readings are considered for training local fusion models f_i, which increases the vulnerability to becoming trapped in minor trends induced by observation outliers. Conversely, when κ is excessively large, it may hinder f_i from discerning subtle data variations and then result in learning only the general data trend direction, thus losing the advantage of the local model. Therefore, we can set $\kappa = 5$ by default for efficiency or accurately determine it as follows.

To practically determine an appropriate value for κ in real-world applications, we could run algorithms with varying κ and choose the one leading to the minimum cumulative fusion loss, i.e., the fusion loss in Formula 6 under different κ settings of the fusion results obtained by certain κ. Theoretically, a low fusion loss implies good suitability for the fusion values w.r.t. local fusion models, which is thus believed to suggest an accurate fusion result. We thus recommend determining the κ value corresponding to the fusion result with the minimum fusion loss. Figure 18 illustrates the data fusion MSE and loss of our methods with varying κ. As shown, the κ values leading to the minimum fusion loss in Figure 18(b) generally return the best fusion result in Figure 18(a), which demonstrates the effectiveness of the determination strategy empirically.

Moreover, taking into account the potential impact of clustering in DFRC, we further conduct experiments. Figure 19 reports the results over GPS and IMU datasets where various clustering techniques, i.e., K-Means, K-Median, and DBSCAN, are used in our DFRC. As shown, DFRC performs overall similarly with different clustering methods, which demonstrates the robustness of DFRC to clustering methods and results.

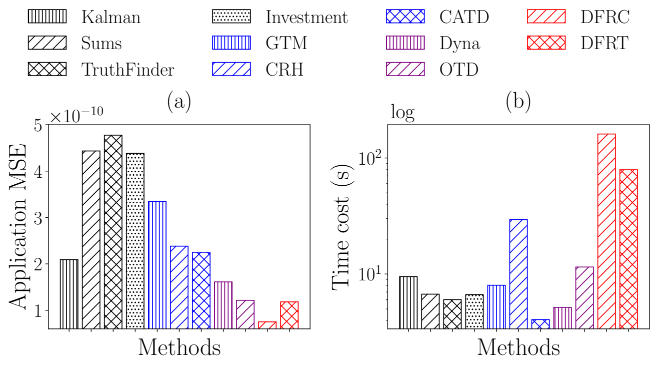

Fig. 20. Downstream integration navigation performance of various methods over GINS.

6.4 Application Study

To validate the effectiveness of conducting data fusion in real-world applications, we consider the application study of integration navigation over GINS dataset. The fusion results returned by various approaches are fed into an inertial navigation system [50] to derive the position values, which are then compared with the high-precision position observations to evaluate the MSE. As shown in Figure 20, those approaches perform well in the data fusion task in Figures 7-9 generally have better integration navigation performance as well. Moreover, our DFRC achieves the best accuracy, followed by DFRT. The results verify again the superiority of our approaches on both the data fusion result and the improvement of real downstream applications.

7 RELATED WORK

In this section, we discuss related works in the fields of temporal data fusion, temporal data repairing, and truth discovery, as well as how they relate to our work.

7.1 Temporal Data Fusion

Although there are many methods used for data fusion, few of them are specifically designed for fusing temporal data, especially for multiple sensor data. Bayesian estimators [1, 3, 8] are utilized to model data reliability by probability distributions, which makes them susceptible to interference from outliers. Kalman filter [26] and its non-Gaussian noise [12] or nonlinear variants [25, 54] are widely used data fusion techniques, which estimate system states and reduce estimation uncertainty by recursively fusing prior and current measurement information. Besides, distributed Kalman filter [41] primarily focuses on estimating the dynamic system state through multiple distributed sensors of different types or located in different positions. On the contrary, our methods specifically address scenarios involving multiple sensor readings for the same observation. However, Kalman filter and its various variants are not designed for fusing multiple same-type sensor data and are more suited for complementary or cooperative data [7]. Although deep Learning techniques like CNN and LSTM are also employed for multi-modal data fusion, lacking the consideration of temporal information from multiple sensors, they are not suitable for our problem either. Furthermore, unlike Kalman filter requiring accurate prior knowledge and deep neural network models asking for sufficient labeled data, our methods do not need additional prior information or domain-specific knowledge, making it more versatile and extensible.

7.2 Temporal Data Repairing

Data repairing is an essential way to handle the outliers in temporal data. Constraints are widely used in temporal data repairing. Relying on various constraints, it is able to rectify the erroneous data. Sequential dependencies [20] consider the range of value changes between two consecutive observations, and speed and acceleration constraints [46, 47] are further devised to repair outliers by modeling value changes with adjacent timestamps. Correcting data with the statistical features is also an important direction of temporal data repairing. The smoothing techniques [5, 9] identify and repair dirty values to make the time series data more smooth. Various likelihoods associated with data change rates are also studied for a statistical-based repairing [66], and IMR [67] further constructs ARX models with the help of a few labeled data. Besides, misplaced data repairing is also studied [49], based on the likelihood estimation between nearest neighbors. Notably, all the aforesaid temporal data repairing methods are designed for cleaning the outliers in a single time series, which are not directly applicable for fusing the multiple time series observations generated by same-type sensors.

7.3 Truth Discovery

Early truth discovery methods are primarily designed for handling discrete data [14, 19, 42, 65, 69]. Among them, Accusim [14] considers the situation where the source reliability is not independent. Besides, subsequent methods [31, 68] consider continuous data while lacking the consideration of temporal relationships between data. Recent studies [33, 57, 62] take into account the temporal information for the truth discovery. The existing study [57] considers varying truths but focuses on the batch operation on categorical data, which is thus not applicable for fusing multiple sensor observations. In contrast, Dyna [33] considers continuous data, relying on the smoothing assumption that truth values are similar at adjacent timestamps, limiting its adaptability to potential abrupt changes in data. To handle this problem, OTD [62] obtains the fusion values by predicting values through the predicting model and weighted observations, then updates source reliability based on the estimation. However, the estimation of truth and assessment of source reliability lag behind continuously updated data, which introduces inaccuracy. Furthermore, most truth discovery methods basically rely on assessing source reliability to compute fusion values, and so typically require a substantial number of sources for data comparison and integration. However, in scenarios such as multi-sensor applications where the available number of sources (sensors) is limited, these methods become less effective.

8 CONCLUSION

In this paper, we devise the data fusion strategy based on the maximum likelihood estimation to determine the most probable changing trends of fusion results with adjacent timestamps. Such strategies can fuse the observations by multiple low-precision sensors, to obtain nearly accurate fusion results at an affordable cost, instead of simply using the expensive high-precision sensors. To find the data fusion result having the maximum likelihood w.r.t. local fusion models, we (1) formalize the problem and analyze the NP-hardness of the problem in Theorem 1; (2) devise an exact algorithm based on the dynamic programming that runs in pseudo-polynomial time in Proposition 2; (3) present an efficient approximation algorithm with performance guarantees in Proposition 5; (4) design a heuristic algorithm for the trade-off between effectiveness and efficiency. Experiments on real datasets with various characteristics show the superiority of our work in low-precision sensor data fusion. Moreover, the fast response for fusing online streaming sensor observations with better fusion accuracy demonstrates the practicality and scalability of our study.

Acknowledgements

We thank the reviewers for their constructive and professional comments. We would also like to thank Dr. Yinan Mei for insightful discussions and valuable suggestions. This work is supported in part by the National Natural Science Foundation of China (62302241, 62072265), Natural Science Foundation of Tianjin (22JCQNJC01520), the Fundamental Research Funds for the Central Universities, Nankai University (63231147).

REFERENCES

[1] W. A. Abdulhafiz and A. Khamis. Handling data uncertainty and inconsistency using multisensor data fusion. *Advances in Artificial Intelligence*, 2013:11–11, 2013.

[2] D. H. Bailey. High-precision floating-point arithmetic in scientific computation. *Computing in science & engineering*, 7(3):54–61, 2005.

[3] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[4] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[5] D. R. Brillinger. *Time series: data analysis and theory.* SIAM, 2001.

[6] P. Caserman, A. Garcia-Agundez, R. Konrad, S. Göbel, and R. Steinmetz. Real-time body tracking in virtual reality using a vive tracker. *Virtual Reality*, 23:155–168, 2019.

[7] F. Castanedo et al. A review of data fusion techniques. *The scientific world journal*, 2013, 2013.

[8] C.-Y. Chong. Distributed multitarget multisensor tracking. *Multitarget-multisensor tracking: Advanced applications*, pages 247–296, 1990.

[9] M. K. Chung. Gaussian kernel smoothing. *arXiv preprint arXiv:2007.09539*, 2020.

[10] T. Cipra and R. Romera. Kalman filter with outliers and missing observations. *Test*, 6:379–395, 1997.

[11] E. Clark, S. L. Brunton, and J. N. Kutz. Multi-fidelity sensor selection: Greedy algorithms to place cheap and expensive sensors with cost constraints. *IEEE Sensors Journal*, 21(1):600–611, 2021.

[12] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on signal processing*, 50(3):736–746, 2002.

[13] Y. Deng. Deep learning on mobile devices: a review. In *Mobile Multimedia/Image Processing, Security, and Applications 2019*, volume 10993, pages 52–66. SPIE, 2019.

[14] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.

[15] Y. Dong and Z. Hongyue. Correction of kalman filter in the presence of outlier. In *Proceedings of 1994 IEEE International Conference on Industrial Technology-ICIT'94*, pages 29–33. IEEE, 1994.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[17] C. Fang, S. Song, Y. Mei, Y. Yuan, and J. Wang. On aligning tuples for regression. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 336–346, 2022.

[18] E. Frank, L. Trigg, G. Holmes, and I. H. Witten. Naive bayes for regression. *Machine Learning*, 41:5–25, 2000.

[19] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140, 2010.

[20] L. Golab, H. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *Proceedings of the VLDB Endowment*, 2(1):574–585, 2009.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[22] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 11–22, 2004.

[23] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[24] N. Javed and T. Wolf. Automated sensor verification using outlier detection in the internet of things. In *2012 32nd International Conference on Distributed Computing Systems Workshops*, pages 291–296. IEEE, 2012.

[25] S. J. Julier and J. K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. Spie, 1997.

[26] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[27] A. Klein. Incorporating quality aspects in sensor data streams. In *Proceedings of the ACM first Ph. D. workshop in CIKM*, pages 77–84, 2007.

[28] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[29] T. D. Larsen, K. L. Hansen, N. A. Andersen, and O. Ravn. Design of kalman filters for mobile robots; evaluation of the kinematic and odometric approach. In *Proceedings of the 1999 IEEE international conference on control applications (Cat. No. 99CH36328)*, volume 2, pages 1021–1026. IEEE, 1999.

[30] H. Li, G. Gong, and Q. Du. Prototype of white rabbit network in lhaaso. In *Proc. 15th ICALEPCS*, pages 1–4, 2015.

[31] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014.

[32] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198, 2014.

[33] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han. On the discovery of evolving truth. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 675–684, 2015.

[34] C. Liu, P. Shui, G. Wei, and S. Li. Modified unscented kalman filter using modified filter gain and variance scale factor for highly maneuvering target tracking. *Journal of Systems Engineering and Electronics*, 25(3):380–385, 2014.

[35] D. Liu, H. Wang, Q. Xia, and C. Jiang. A low-cost method of improving the gnss/sins integrated navigation system using multiple receivers. *Electronics*, 9(7):1079, 2020.

[36] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[37] K. Mahmood, S. Niki, Y. Nakahara, X. Lu, I. Luque, and K. Mori. Autonomous real-time navigation for service level agreement in distributed information service system. In *Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)*, pages 231–238. IEEE, 2007.

[38] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[39] D. V. Nam and K. Gon-Woo. Robust stereo visual inertial navigation system based on multi-stage outlier removal in dynamic environments. *Sensors*, 20(10):2922, 2020.

[40] D. Nemec, J. Andel, V. Simak, and J. Hrbcek. Homogeneous sensor fusion optimization for low-cost inertial sensors. *Sensors*, 23(14):6431, 2023.

[41] R. Olfati-Saber. Distributed kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498. IEEE, 2007.

[42] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 877–885, 2010.

[43] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: Fast similarity search under the time warping distance. In *Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, page 326–337, New York, NY, USA, 2005. Association for Computing Machinery.

[44] M. Scannapieco and T. Catarci. Data quality under a computer science perspective. *Archivi & Computer*, 2:1–15, 2002.

[45] G. Shankaranarayanan and Y. Cai. Supporting data quality management in decision-making. *Decision support systems*, 42(1):302–317, 2006.

[46] S. Song, F. Gao, A. Zhang, J. Wang, and P. S. Yu. Stream data cleaning under speed and acceleration constraints. *ACM Transactions on Database Systems (TODS)*, 46(3):1–44, 2021.

[47] S. Song, A. Zhang, J. Wang, and P. S. Yu. Screen: stream data cleaning under speed constraints. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 827–841, 2015.

[48] R. G. D. Steel, J. H. Torrie, et al. Principles and procedures of statistics. *Principles and procedures of statistics.*, 1960.

[49] Y. Sun, S. Song, C. Wang, and J. Wang. Swapping repair for misplaced attribute values. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 721–732. IEEE, 2020.

[50] H. Tang, X. Niu, T. Zhang, J. Fan, and J. Liu. Exploring the accuracy potential of imu preintegration in factor graph optimization. *arXiv preprint arXiv:2109.03010*, 2021.

[51] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. *Computer*, 39(7):55–62, 2006.

[52] D. Veitch, S. Babu, and A. Pasztor. Robust synchronization of software clocks across the internet. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 219–232, 2004.

[53] K. Vinther, K. F. Jensen, J. A. Larsen, and R. Wisniewski. Inexpensive cubesat attitude estimation using quaternions and unscented kalman filtering. *Automatic Control in Aerospace*, 4(1), 2011.

[54] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[55] R. Y. Wang. A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65, 1998.

[56] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[57] S. Wang, D. Wang, L. Su, L. Kaplan, and T. F. Abdelzaher. Towards cyber-physical systems in social spaces: The data reliability challenge. In *2014 IEEE Real-Time Systems Symposium*, pages 74–85, 2014.

[58] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64, 2004.

[59] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.

[60] Y. Xiang, R. Piedrahita, R. P. Dick, M. Hannigan, Q. Lv, and L. Shang. A hybrid sensor system for indoor air quality monitoring. In *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 96–104. IEEE, 2013.

[61] E. Xu, Z. Ding, and S. Dasgupta. Target tracking and mobile sensor navigation in wireless sensor networks. *IEEE Transactions on mobile computing*, 12(1):177–186, 2011.

[62] L. Yao, L. Su, Q. Li, Y. Li, F. Ma, J. Gao, and A. Zhang. Online truth discovery on time series data. In *Proceedings of the 2018 siam international conference on data mining*, pages 162–170. SIAM, 2018.

[63] C. Yawut and S. Kilaso. A wireless sensor network for weather and disaster alarm systems. In *International conference on information and electronics engineering, IPCSIT*, volume 6, pages 155–159. Citeseer, 2011.

[64] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. 2000.

[65] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1048–1052, 2007.

[66] A. Zhang, S. Song, and J. Wang. Sequential data cleaning: A statistical approach. In *Proceedings of the 2016 International Conference on Management of Data*, pages 909–924, 2016.

[67] A. Zhang, S. Song, J. Wang, and P. S. Yu. Time series data cleaning: From anomaly detection to anomaly repairing. *Proceedings of the VLDB Endowment*, 10(10):1046–1057, 2017.

[68] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. *Proc. of QDB*, 1817, 2012.

[69] B. Zhao, B. I. Rubinstein, J. Gemmell, and J. Han. A bayesian approach to discovering truth from conflicting sources for data integration. *arXiv preprint arXiv:1203.0058*, 2012.

A PROOFS

A.1 Proof of Theorem 1

The problem is clearly in NP. Given a selection Y, it can be verified in polynomial time whether the loss of the fusion result $\mathscr{L}(Y) \leq \ell$.

To prove the NP-hardness, we show a reduction from the Path Traveling Salesman Problem (Path-TSP) problem.

Consider a graph $G = (V, E)$, where V is the set of vertices and E is the edge set. Let ω_{ij} denote the weight of the edge $e_{ij} = (v_i, v_j), v_i, v_j \in V$. Then we consider constructing the corresponding input of multi-source sensor data X. At each timestamp, there exists an observation x_{ij} in X^i if a vertex $v_j \in V$ is present. The fusion loss for the selection of two observations x_{ij_1} and x_{i+1,j_2} at consecutive timestamps is denoted by e_{ij}, when an edge connects v_{j_1} and v_{j_2}. Otherwise, the fusion loss is deemed infinite. Specifically, $\mathscr{L}(Y) = \mathscr{L}(\{y_1, y_2, \ldots, y_n\}) = \sum_{i=1}^{n-1} \omega_{y_i y_{i+1}}$. Moreover, if there are duplicate vertices in the selection, we will set the fusion loss to ∞.

Then, we will show that there exists a path $\mathscr{P}$ with distance $L(\mathscr{P}) \leq \ell$, if and only if there is a selection Y with loss $\mathscr{L}(Y) \leq \ell$.

First, suppose that G has a path $\mathscr{P} = \{P_1, P_2, \ldots, P_n\}$ with path distance $L(\mathscr{P}) = \ell = \sum_{i=1}^{n-1} d(P_i, P_{i+1})$. For each vertex $P_i \in \mathscr{P}$, $P_i \in \mathbb{V}$, and $P_i \neq P_j$ if $i \neq j$, where $i, j \in [1, n]$. We consider a set of multi-sensor time series data X with timestamp number n and sensor number m, $m = n$, the fusion result Y with the sub-time series length $\kappa(= n)$. For every observation $x_{i,j}$, we assign a corresponding identifier $z_{i,j}$ to indicate whether it is selected at timestamp X^i. Therefore, for every $z_{i,\_}$ at X^i, we set $z_{i,j}$ to 1 if $P_i = v_j$, otherwise to 0. Thus, $\mathscr{L}(Y) = \sum_{i=1}^{n-1} z_{i,j} z_{i+1,k} \omega_{x_{i,j} x_{i+1,k}}, j, k \in [1, n]$. Obviously, $\omega_{x_{i,j} x_{i+1,k}}$ can only be counted when both $z_{i,j}$ and $z_{i+1,k}$ are equal to 1, which occurs when $P_i = v_j$ and $P_{i+1} = v_k$, at this point, $d(v_j, v_k)$ is also counted into path distance ℓ. Then we have $\mathscr{L}(Y) = \ell$.

Next, suppose that Y is a selection with loss ℓ. We consider a path $\mathscr{P} = \{P_1, P_2, \ldots, P_n\}$, $P_i = v_j$ if $z_{i,j} = 1$, which also means that the fusion result y_i at timestamp X^i equals to x_{ij}. Then we have $L(\mathscr{P}) = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) = \ell$, where we can obtain the whole path distance' but not the individual distances between v_i and v_{i+1} themselves.

Moreover, it is notable that the m^κ complexity component cannot be avoided whenever we want to compute the exact result. The reason is that there are always m^κ possible candidates considered for each local fusion model. Specifically, we consider a subsequence with length κ each time, i.e., $\mathscr{P} = \{P_1, P_2, \ldots, P_\kappa\}$. For any specific $\mathscr{P}$, it has m candidates for each timestamp X^i, thus we have m^κ candidates for $\mathscr{P}$ within a sub-time series with length κ. In this sense, to obtain the exact solution of the optimal multi-sensor data fusion model, any algorithm cannot avoid the m^κ complexity component.

A.2 Proof of Proposition 2

First, we'll show the correctness of the dynamic programming recurrence equation in Formula 8. That is, $\mathscr{L}(n - \kappa, c_{n-\kappa})$ always calculates the minimum loss.

Suppose that there is a $\mathscr{L}(n - \kappa, c^*_{n-\kappa})$, $\mathscr{L}(n - \kappa, c^*_{n-\kappa}) < \mathscr{L}(n - \kappa, c_{n-\kappa}), c^*_{n-\kappa} \neq c_{n-\kappa}$. It contradicts with Line 8, where we choose the $Y_{n-\kappa, c^*}$ with minimum loss.

Suppose that there is another $\mathscr{L}^*(n - \kappa, c^*_{n-\kappa})$, $\mathscr{L}^*(n - \kappa, c^*_{n-\kappa}) < \mathscr{L}(n - \kappa, c_{n-\kappa}), c^*_{n-\kappa} = c_{n-\kappa}$. Then there would also be another backtracking process. Let's backtrack to get c^*_i and c_i iteratively from $\mathscr{L}^*(n-\kappa, c^*_{n-\kappa})$ and $\mathscr{L}(n-\kappa, c_{n-\kappa})$ respectively. During the iterative process, when $\mathscr{L}(i, c^*_i) < \mathscr{L}(i, c_i)$, once $c^*_{i-1} = c_{i-1}$, it implies that the equation will derive $\mathscr{L}^*(i, c^*_i) = \mathscr{L}(i, c_i)$ in the current iteration of backtracking. It leads to a contradiction as well. Therefore, if there is no contradiction all the way until $i = 1, c_1 \neq c^*_1$, we could obtain $\{\mathscr{L}^*(1, c^*_1), \ldots, \mathscr{L}^*(n-\kappa, c^*_{n-\kappa})\} \cap \{\mathscr{L}(1, c_1), \ldots, \mathscr{L}(n-\kappa, c_{n-\kappa})\} = \emptyset$ during the iterative process, which means $c^*_{n-\kappa} \neq c_{n-\kappa}$ and contradicts the condition

$c^*_{n-\kappa} = c_{n-\kappa}$. Finally, if the condition $\mathscr{L}^*(i, c^*_i) < \mathscr{L}(i, c_i)$ never occurs during the iterative process, then there should be $\mathscr{L}^*(n-\kappa, c^*_{n-\kappa}) = \mathscr{L}(n-\kappa, c_{n-\kappa})$, which also contradicts the assumption at the start.

Thus, the correctness of the recurrence equation is proved.

Next, we will provide a proof of the complexity. As shown in Formula 8, we need $O(nm^\kappa)$ space to maintain $\mathscr{L}(i, c_i)$. By considering m^κ combinations for $\mathscr{L}(\{x_{ij}\} \cup Y_{ic_i})$ in each calculation of $\mathscr{L}(i, c_i)$, the loops in Lines 1, 2, 4 lead to time complexity $O((n-\kappa) \times m^\kappa \times m)$, i.e., $O((n-\kappa)m^{\kappa+1})$. Furthermore, even considering that the simple linear regression is employed in the computation of $\mathscr{L}(\{x_{ij}\} \cup Y_{ic})$ in Line 6, the dynamic programming still runs in $O((n-\kappa)m^{\kappa+1}\kappa)$, i.e., $O(nm^\kappa\kappa)$.

A.3 Proof of Proposition 3

Line 2 obtains the cluster set through clustering algorithms for each X^i and Lines 3 to 4 choose represent points for each U_{ik} at X^i, which leads to the time complexity $O(m\lambda(q+1))$ for K-means, $O(m + m\lambda)$ for DBSCAN, $O(m^2 + m\lambda)$ for Hierarchical clustering, where q is the clustering iterations. Repeating the above process from X^1 to X^n, we will have $O(nm\lambda(q+1))$, $O(nm)$, $O(nm^2)$ respectively. Furthermore, Line 6 performs DFEC(X', κ), which results in time complexity $O(n\lambda^\kappa\kappa + n\lambda(q+1))$ considering g_i as the widely adopted K-Means, in $O(n\lambda^\kappa\kappa + nm(m+\lambda))$ time as DBSCAN, in $O(n\lambda^\kappa\kappa + nm(m^2+\lambda))$ time as Hierarchical clustering. Actually, different clustering algorithms do not affect the final time complexity $O(n\lambda^\kappa\kappa)$ of Algorithm 2. Similarly, we need $O(n\lambda^\kappa + n\lambda)$ space to generate X' and run DFEC(X', κ) in Line 6, additional space requirements $O(m)$ for DBSCAN, $O(m^2)$ for Hierarchical clustering, $O(m+\lambda)$ for K-means. Therefore, all of them do not impact the final space complexity, i.e., $O(n\lambda^\kappa)$.

A.4 Proof of Proposition 4

Considering that the data $\{X^i, \dots, X^{i+\kappa}\}$ are evenly spaced on the x-axis (equidistant timestamps), let's perform an offset on the observations. When κ is even, we move sensor observations at the center timestamp (say $t_{\frac{\kappa}{2}+1}$) to the y-axis. Moreover, if κ is odd, we will set the y-axis as the mid-line between two central timestamps (say $t_{\frac{\kappa+1}{2}}$ and $t_{\frac{\kappa+1}{2}+1}$). In this scenario, we can obtain the mean value of timestamps $\bar{t} = \frac{t_i+t_{i+1}+\cdots+t_{i+\kappa}}{\kappa+1} = 0$, $i \in [1, n-\kappa]$. Therefore, it follows that: $\begin{cases} a = \frac{\sum_{i=1}^{\kappa+1} t_i(y_i-\bar{y})}{\sum_{i=1}^{\kappa+1} t_i^2} \\ b = \bar{y} \end{cases}$, where a denotes the weight of f_i, and b is the bias, with the mean value $\bar{y} = \frac{\sum_{i=j}^{j+\kappa} y_i}{\kappa+1}$ of sensor readings selected by the local fusion model. Subsequently, the corresponding local fusion model performs the fusion process based on the selected sensor readings. For the absolute value of the residual between fusion result y_j and the prediction value $\hat{y}_j$ obtained by the local fusion model at t_j, $1 \le j \le n$. Considering the differences between prediction values and observations at the j-th timestamp t_j, we have

$$d = \left|\hat{y}_j - y_j\right|. \tag{12}$$

When substituting into the prediction equation yields, it becomes

$$\left|\hat{y}_j - y_j\right| = \left|at + b - y_j\right|. \tag{13}$$

We further consider the formulas for the model's slope and bias, having

$$\left|at + b - y_j\right| = \left|\frac{\sum_{i=1}^{\kappa+1}(t_i y_i) - \kappa\bar{t}\bar{y}}{\sum_{i=1}^{\kappa+1} t_i^2} t + \bar{y} - a\bar{t} - y_j\right|. \tag{14}$$

Since $\bar{t} = 0$, it leads to

$$\left|\frac{\sum_{i=1}^{\kappa+1}(t_i y_i) - \kappa \bar{t}\bar{y}}{\sum_{i=1}^{\kappa+1} t_i^2} t + \bar{y} - a\bar{t} - y_j\right| = \left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t + \bar{y} - y_j\right|. \tag{15}$$

Combining all the aforesaid Formulas 12-15, we obtain the equation

$$d = \left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t + \bar{y} - y_j\right|. \tag{16}$$

Considering the case when κ is an even number. If (t, y_j) is the point on the mid-line, due to $t = 0$, then $d = \left|\bar{y}_j - y_j\right| < D$.

If (t, y_j) is not the point on the mid-line, there are $\frac{\kappa}{2}$ observations to the left and the right of y-axis. Let r represent the spacing between observations along the x-axis. Consider that after the offset, the values are now symmetrical based on the y-axis. We can merge the two segments on the y-axis at timestamps equidistant from the y-axis. Additionally, since the timestamps are evenly spaced, we know that $\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right|$ equals to

$$\left|\frac{r^2\left[(y_{\kappa+1} - y_1)\frac{\kappa}{2} + (y_\kappa - y_2)\frac{\kappa-2}{2} + \cdots + \left(y_{\kappa+1-\frac{\kappa}{2}} - y_{\frac{\kappa}{2}}\right)\right]}{2r^2\left(1^2 + 2^2 + \cdots + \left(\frac{\kappa}{2}\right)^2\right)} \frac{t}{r}\right|. \tag{17}$$

According to the definition of D_i, we have $D_i \leq y_{\kappa+1-i} - y_{1+i}$. Combining with Formula 17, we can obtain

$$\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right| \leq \left|\frac{D_i\left(\frac{\kappa}{2} + \frac{\kappa-2}{2} + \cdots + 1\right)}{2\left(1^2 + 2^2 + \cdots + \left(\frac{\kappa}{2}\right)^2\right)} \frac{t}{r}\right|. \tag{18}$$

Furthermore, combining the arithmetic sequence formula and the sum of squares formula can lead to

$$\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right| \leq \left|\frac{3D_i}{2(\kappa+1)} \frac{t}{r}\right|. \tag{19}$$

Taking the characteristics of timestamps into account, i.e., $-\frac{\kappa}{2} \leq \frac{t}{r} \leq \frac{\kappa}{2}$, we can derive

$$d \leq \left|\frac{3D_i}{2(\kappa+1)} \frac{t}{r}\right| + D_i \leq \frac{7\kappa + 4}{4(\kappa+1)} D_i. \tag{20}$$

Thus, we conclude that

$$d \leq \frac{7}{4} D_i. \tag{21}$$

Considering the case when κ is an odd number. There are $\frac{\kappa+1}{2}$ observations to the left and the right of y-axis. Similarly, based on the symmetry, we know that $\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right|$ equals to

$$\left|\frac{r^2\left[(y_{\kappa+1} - y_1)\frac{\kappa}{2} + (y_\kappa - y_2)\frac{\kappa-2}{2} + \cdots + (y_{\kappa+1-\frac{\kappa+1}{2}} - y_{\frac{\kappa+1}{2}})\frac{1}{2}\right]}{2r^2\left((\frac{1}{2})^2 + (\frac{3}{2})^2 + \cdots + (\frac{\kappa}{2})^2\right)} \frac{t}{r}\right|. \tag{22}$$

Moreover, since we know $D_i \leq y_{\kappa+1-i} - y_{1+i}$, we can also get

$$\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right| \leq \left|\frac{D_i\left(\frac{\kappa}{2}+\frac{\kappa-2}{2}+\cdots+\frac{1}{2}\right)}{2r^2\left((\frac{1}{2})^2+(\frac{3}{2})^2+\cdots+(\frac{\kappa}{2})^2\right)}\frac{t}{r}\right|. \tag{23}$$

Then, combine the arithmetic sequence formula and the sum of squares formula,

$$\left|\frac{\sum_{i=1}^{\kappa+1} t_i y_i}{\sum_{i=1}^{\kappa+1} t_i^2} t\right| \leq \left|\frac{D_i(\kappa+1)}{\frac{8((\kappa+1)^2-1)}{3}}\frac{t}{r}\right|. \tag{24}$$

In conjunction with $-\frac{\kappa}{2} \leq \frac{t}{r} \leq \frac{\kappa}{2}$, we can obtain

$$d \leq \left|\frac{D_i(\kappa+1)}{\frac{8((\kappa+1)^2-1)}{3}}\frac{t}{r}\right| + D_i \leq \frac{7\kappa+11}{4(\kappa+2)}D_i. \tag{25}$$

Therefore, we can also conclude that

$$d \leq \frac{7}{4}D_i. \tag{26}$$

A.5 Proof of Proposition 5

In this proof, we'll start with the topic of optimal estimation and then derive the bound of the error introduced by the clustering procedure between the approximation solution L and the exact solution L^*.

As for the fusion loss $\mathscr{L}(\mathbf{y}_i)$ defined in Formula 6, we assume that it considers the loss between $\mathbf{y}_i$ and its optimal estimation of f_i. Here, we expand the definition to $\mathscr{L}(\mathbf{y}_i \mid l)$, signifying the loss between $\mathbf{y}_i$ and an estimation l. Given $\mathbf{y}_i$ of the approximation solution Y with $f_i(\mathbf{y}_i)$'s optimal estimation $l'_{\mathbf{y}_i}$ and its any other estimation $l_{\mathbf{y}_i}$, we can obtain $\mathscr{L}(\mathbf{y}_i \mid l'_{\mathbf{y}_i}) \leq \mathscr{L}(\mathbf{y}_i \mid l_{\mathbf{y}_i})$, because the optimal estimation is the coefficient values that minimize the sum of squared errors (in the least squares method).

If $\forall y_i \in Y$, there is $y_i \in U_{ik}$, $\{y_i^*\} \cap U_{ik} \neq \emptyset$, which means that every $y_i \in Y$ is in the same cluster with the observation in the optimal solution $y_i^* \in y^*$. Considering $\mathbf{y}_i^*$ in Y^*, we can thus obtain

$$\sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) - \mathscr{L}(\mathbf{y}_i^*) \leq \sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i \mid l'_{\mathbf{y}_i^*}) - \mathscr{L}(\mathbf{y}_i^* \mid l'_{\mathbf{y}_i^*}). \tag{27}$$

Then, consider the maximum difference that may exist between the optimal solution and the approximate solution at each timestamp for each local fusion model. We can get the following inequality

$$\sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) - \mathscr{L}(\mathbf{y}_i^*) \leq \sum_{i=1}^{n-\kappa}\sum_{k=1}^{\kappa+1} \left[(d_k+\eta)^2 - d_k^2\right], \tag{28}$$

where d_κ is the absolute value of the residual between the optimal solution $y_\kappa^* \in \mathbf{y}_\kappa^*$ and the estimation obtained by the corresponding local fusion model.

Furthermore, combining with Proposition 4, which specifies the upper bound of d, we have

$$\sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) - \mathscr{L}(\mathbf{y}_i^*) \leq \sum_{i=1}^{n-\kappa} (\kappa+1)\left(\frac{7}{2}D_i\eta + \eta^2\right). \tag{29}$$

If $\exists y_i \in Y$, there is $y_i \in U_{ik}$, $\{y_i^*\} \cap U_{ik} = \emptyset$, which means that at least one y_i is not in the same cluster with the observation in the optimal solution $y_i^* \in Y^*$. Since Algorithm 1 is applied on X', such a situation implies that selecting points from different $U_{ik'} \in U_i$ results in lower loss compared to that from the same cluster U_{ik} as Y^*. It means that

$$\sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) \le \sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}'_i), \tag{30}$$

where $\mathbf{y}'_i \subset Y'$, $\forall y'_i \in Y'$ if $y'_i \in U_{ik}$, then $\{y^*_i\} \cap U_{ik} \neq \emptyset$. That is, Y' is the selection whose every observation is in the same clusters with Y^*.

Conjunction with Formula 29, whether or not y_i belongs to the same cluster as the optimal selected observation y^*_i, it can be deduced that

$$\sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) - \mathscr{L}(\mathbf{y}^*_i) \le \sum_{i=1}^{n-\kappa} (\kappa + 1) \left(\frac{7}{2} D_i \eta + \eta^2 \right) \tag{31}$$

Thus, combining Formula 29 and Formula 31, we can conclude the approximation guarantee that

$$L - L^* = \sum_{i=1}^{n-\kappa} \mathscr{L}(\mathbf{y}_i) - \mathscr{L}(\mathbf{y}^*_i) \le (\kappa + 1)(n - \kappa) \left(\frac{7}{2} D\eta + \eta^2 \right). \tag{32}$$

Received October 2023; revised January 2024; accepted February 2024