On Aligning Tuples for Regression

Chenguang Fang
BNRist, Tsinghua University
fcg19@mails.tsinghua.edu.cn

Shaoxu Song
BNRist, Tsinghua University
sxsong@tsinghua.edu.cn

Yinan Mei
BNRist, Tsinghua University
myn18@mails.tsinghua.edu.cn

Ye Yuan
Beijing Institute of Technology
yuan-ye@bit.edu.cn

Jianmin Wang
BNRist, Tsinghua University
jimwang@tsinghua.edu.cn

ABSTRACT

Regression models are learned over multiple variables, e.g., using engine torque and speed to predict its fuel consumption. In practice, the values of these variables are often collected separately, e.g., by different sensors in a vehicle, and need to be aligned first in a tuple before learning. Unfortunately, owing to various issues like network delays, values generated at the same time could be recorded with different timestamps, making the alignment difficult. According to our study in a vehicle manufacturer, engine torque, speed and fuel consumption values are mostly not recorded with the same timestamps. Aligning tuples by simply concatenating values of variables with equal timestamps leads to limited data for learning regression model. To deal with timestamp variations, existing time series matching techniques rely on the similarity of values and timestamps, which unfortunately are very likely to be absent among the variables in regression (no similarity between engine torque and speed values). In this sense, we propose to bridge tuple alignment and regression. Rather than similar values and timestamps, we align the values of different variables in a tuple that (i) are recorded in a short period, i.e., time constraint, and more importantly (ii) coincide well with the regression model, known as model constraint. Our theoretical and technical contributions include (1) formulating the problem of tuple alignment with time and model constraints, (2) proving NP-completeness of the problem, (3) devising an approximation algorithm with performance guarantee, and (4) proposing efficient pruning strategies for the algorithm. Experiments over real world datasets, including the aforesaid engine data collected by a vehicle manufacturer, demonstrate that our proposal outperforms the existing methods on alignment accuracy and improves regression precision.

CCS CONCEPTS

• **Information systems → Data management systems**.

KEYWORDS

time series; alignment; regression model

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539373

ACM Reference Format:
Chenguang Fang, Shaoxu Song, Yinan Mei, Ye Yuan, and Jianmin Wang. 2022. On Aligning Tuples for Regression. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539373

1 INTRODUCTION

Regression models are learned over multiple variables, often collected separately. For example, engine torque and speed are collected by different sensors in a vehicle to determine its fuel consumption [26]. Likewise, for the three variables in Figure 1, voltage is predicted by monitoring active power and intensity in a household for anticipation of voltage violation [15]. To learn the regression model, a necessary preprocessing step is to align the values of multiple variables in a row, e.g., the aligned tuple of three values, active power, intensity and voltage, connected by dashed lines at time 480 in Figure 1.

Unfortunately, owing to transmission or network delays as well as hardware errors, values of different variables generated at the same time could be collected with different timestamps, making tuple alignment challenging [28]. For instance, engine torque, speed and fuel consumption values are collected simultaneously from controller area network (CAN) bus in vehicles, while logged with different timestamps in electronic control units (ECU) owing to transmission delays [25], known as timestamp variations.

Simply aligning multiple variables by equal timestamps lead to limited data for learning regression models. According to the survey in a vehicle manufacturer, only about 5% data are found with engine torque, speed and fuel consumption values having equal timestamps (see Section 4.3.1 for the empirical study).

Instead of equal timestamps, existing time series matching techniques, e.g., dynamic time warping (DTW) [4] and its variations, align variables mainly based on similar values or in a period of close timestamps. However, such proximity on values or timestamps is very likely to be absent among the variables in regression, as the example illustrated in Figure 1.

Another possible alternative is interpolation [23], i.e., interpolating missing values at each timestamp (may consider both temporal and cross-variable correlations). Obviously, the interpolation-based methods do not consider the timestamp variations, but interpolating new values that are error prone.

Example 1. Figure 1 illustrates an example of learning the regression model to predict voltage by active power and intensity. In preprocessing, we need to first align the three values, collected with different timestamps, in a row.

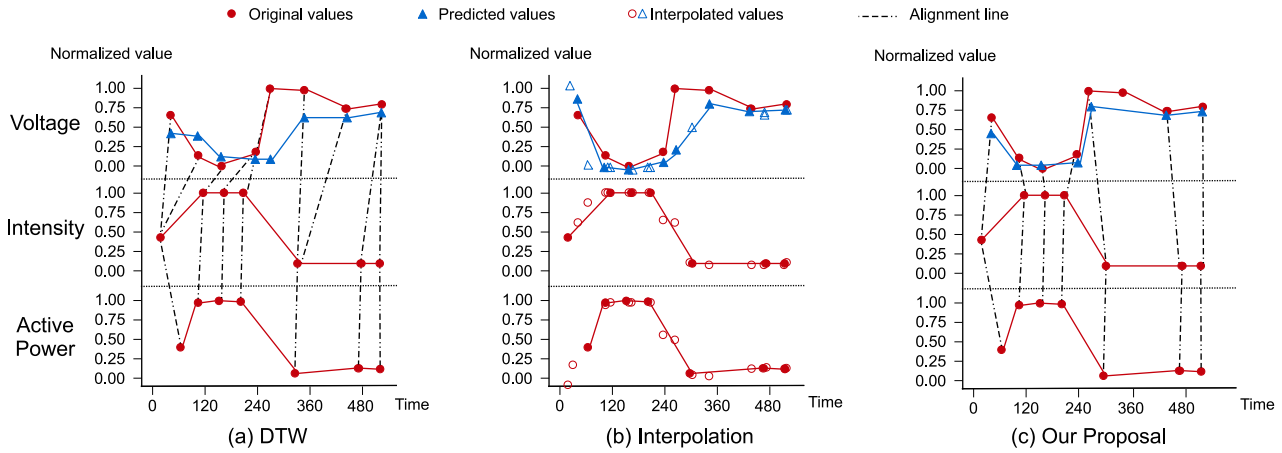


Figure 1: An example of aligning/interpolating three real-world variables by (a) DTW variation [35], (b) spline interpolation [23] and (c) our proposal. The regression model is learned to predict voltage by intensity and active power. Red dots are the original value observations. Dashed lines denote the alignment of values in a row/tuple. Blue triangles are the values predicted by the regression model learned from the aligned data. Hollow points are the interpolated values computed by interpolation algorithm on each timestamp (thus no alignment applies).

Figure 1(a) shows the result of a DTW variation [35], based on value and time similarity. While the normalized values of active power and intensity are somewhat alike and thus align well, they are not similar to voltage values. Learned over the mistakenly aligned rows of active power, intensity and voltage values, the regression model is inaccurate, with predictions (blue triangles) deviating from observations (red dots).

Figure 1(b) presents the result based on spline interpolation. The hollow points are the values interpolated by spline as time series. The interpolation-based method does not consider the alignment of timestamps, but simply interpolates (possibly imprecise) values at each timestamp with no observation. Without addressing timestamp variations but introducing new error prone values, the prediction with interpolation is inaccurate. ■

In this study, we inventively bridge tuple alignment and regression. Rather than relying on similar values as in DTW, we propose to align the values of different variables in a row, under the guide of (iteratively learned) regression models. Intuitively, the aligned row/tuple is expected to coincide well with the prediction of the regression model (learned in the previous iteration). For instance, in Figure 1(c), the aligned voltage observations (red dots) should coincide with the predictions (blue triangles) of the regression model by the corresponding intensity and active power. Indeed, the aforesaid similarity of values considered in DTW could be interpreted as a special regression model on variables with similar values, and thus subsumed by our solution. In this sense, the alignment under the guide of more general regression model would outperform DTW relying on value similarity in special case, as demonstrated in the experiments in Section 4.

Therefore, we propose to align the values of multiple variables in a tuple, which (1) are recorded in a short period, i.e., time constraint, and (2) coincide well with the (iteratively learned) regression model, known as model constraint. This strategy applies iteratively, i.e., the well aligned data improves model learning, while the improved regression model again advises the next iteration of alignment. While out-of-phase matching has been researched over last 20 years (e.g., DTW [4], GTW [35, 36]), our proposal is the first work to combine both the model constraint (dependencies) and the time constraint (close timestamps) for tuple alignment.

Our major contributions are summarized as follows.

(1) We formalize a novel problem, similarity alignment under model constraint (SAMC), which is the first to utilize both time and model constraints for aligning tuples. We analyze NP-completeness of the problem (Theorem 1).

(2) We devise an efficient approximation algorithm with theoretical performance guarantee (Proposition 2).

(3) We propose innovative strategies and structures with theoretical results to enable efficient pruning, thus significantly reduce the time cost (Propositions 3 and 7).

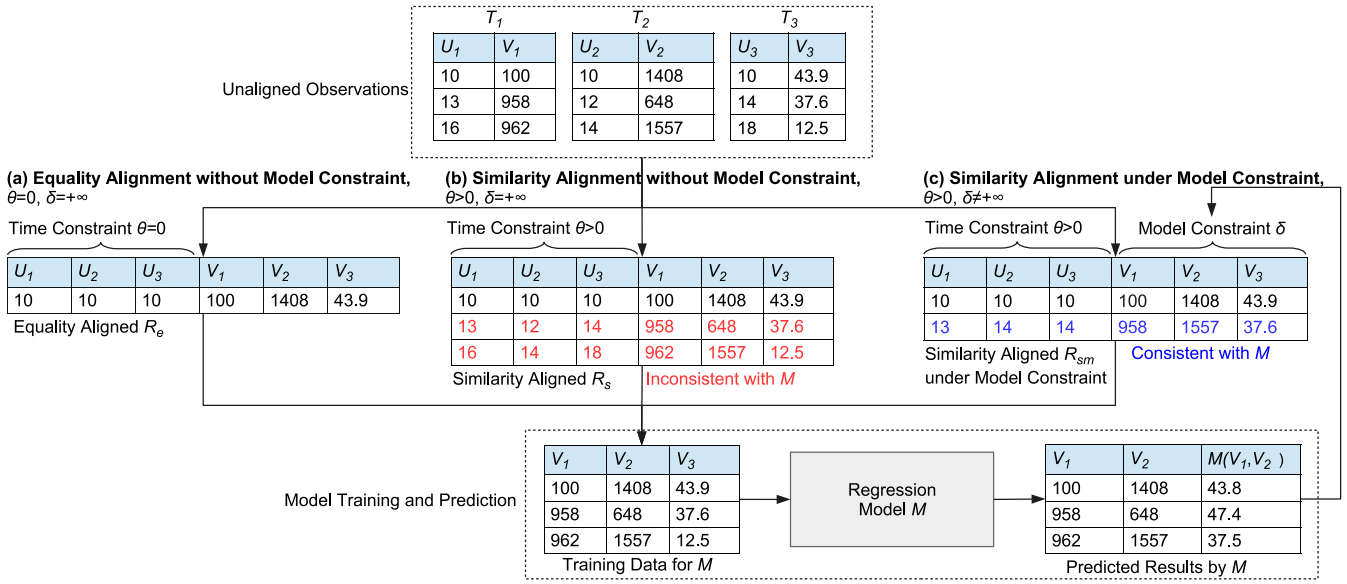
Finally, we conduct a comprehensive experiment over real-world datasets, including the engine data collected by a vehicle manufacturer. It demonstrates that our proposal outperforms the state-of-the-art methods on both alignment and regression performance.

The proposal has been deployed in Apache IoTDB [1], an open-source time series database. The code is in the Github repository of Apache IoTDB [2]. All the Proofs [1] of the main theoretical results and experimental code are available in [3].

2 TUPLE ALIGNMENT PROBLEM

Figure 2 provides an overview of aligning multiple variables for learning regression models. Consider m variables $T_1, T_2, ..., T_m$ with schema $\mathcal{T}_i(U_i, V_i)$, where U_i denotes the timestamp identifying when the variable is observed, and V_i is the corresponding value. Methods are introduced below to obtain an aligned instance R with schema $\mathcal{R}(U_1, U_2, \ldots, U_m, V_1, V_2, \ldots, V_m)$. As illustrated in Figure 1, we do not assume the sampling rate of variables, i.e., supporting irregular time intervals. By a projection on the attributes $V_1, V_2, \ldots, V_m$ over R, we obtain a set of training data. The regression model M could be trained over the aligned attributes.

[1] Due to the limited space, all the proofs are made online in [3].



T_1		T_2		T_3	
U_1	V_1	U_2	V_2	U_3	V_3
10	100	10	1408	10	43.9
13	958	12	648	14	37.6
16	962	14	1557	18	12.5

U_1	U_2	U_3	V_1	V_2	V_3
10	10	10	100	1408	43.9

U_1	U_2	U_3	V_1	V_2	V_3
10	10	10	100	1408	43.9
13	12	14	958	648	37.6
16	14	18	962	1557	12.5

U_1	U_2	U_3	V_1	V_2	V_3
10	10	10	100	1408	43.9
13	14	14	958	1557	37.6

V_1	V_2	V_3
100	1408	43.9
958	648	37.6
962	1557	12.5

V_1	V_2	$M(V_1,V_2)$
100	1408	43.8
958	648	47.4
962	1557	37.5

Figure 2: An overall of various tuple alignment methods with $m = 3$ as an example, including (a) equality alignment (b) similarity alignment and (c) similarity alignment under model constraint. Equality alignment is a special case of similarity alignment with time constraint $\theta = 0$, i.e., each aligned tuple $r \in R_e$ has equal timestamps U_1, U_2, U_3 of T_1, T_2, T_3. Similarity alignment ensures that timestamps U_1, U_2, U_3 in an aligned tuple $r \in R_s$ have distances $\leq \theta$ with each other. The preliminary aligned tuples (by equality or similarity) serve as the training data to learn an initial model M. This model M is then employed as the model constraint on V_1, V_2, V_3 in alignment, i.e., the distance between the prediction $M(r[V_1], r[V_2])$ and $r[V_3]$ of aligned tuple $r \in R_{sm}$ should be no greater than a threshold δ, denoted by $\Delta(r, M) \leq \delta$. The similarity alignment with model constraint rules out alignments inconsistent with M (red tuples in R_s in (b)) and finds more consistent alignments (blue one in R_{sm} in (c)). Multiple iterations could be applied, where the new learned M serves as the model constraint for another round of alignment.

Intuitively, to align variables $T_1, T_2, \ldots, T_m$, a natural idea is based on equal timestamps, namely equality alignment. The corresponding results however are very limited, owing to issues such as transmission delays. The alignment based on similar timestamps, namely similarity alignment, largely enriches the training data but with many aligned tuples inconsistent with the regression model. To rule out inconsistent results, in Section 2.1, we propose to employ the regression model M as a constraint.

To obtain more aligned tuples for training, the aligned instance R is expected to be maximized in size. We show that obtaining an optimal similarity alignment under model constraint is NP-hard (Theorem 1), and devise an approximation (Proposition 2).

2.1 Similarity Alignment and Model Constraint

In this section, we first propose the time constraint, which restricts the timestamps in a given threshold. The model constraint is then devised based on regression models. Combining them, the similarity alignment under model constraint problem is finally proposed.

Definition 1. *The alignment cost of an aligned tuple $r \in R$ w.r.t. timestamp similarity is defined on the time attributes $U_1, U_2, \ldots, U_m$.*

$$\Theta(r) = \max_{A,B \in \{U_1, U_2, \ldots, U_m\}} |r[A] - r[B]|. \tag{1}$$

Intuitively, the values in an aligned tuple $r \in R$ are expected to be recorded in a short period. That is, their timestamps should have distances no greater than a threshold θ, known as *time constraint*,

$$|r[U_i] - r[U_j]| \leq \theta, 0 \leq i, j \leq m. \tag{2}$$

Definition 2. *We say that an aligned instance R satisfies the* time constraint *θ, if $\forall r \in R, \Theta(r) \leq \theta$.*

The aligned tuple based on similar timestamps might not be accurate. In addition to the constraint on timestamps $U_1, U_2, \ldots, U_m$, we further introduce the constraint on aligned values $V_1, V_2, \ldots, V_m$. Intuitively, the regression model M (learned in previous steps) tell the relationships among the values and could guide the subsequent alignment as illustrated in Figure 2(c).

Definition 3. *For a regression model M trained on $V_1, V_2, \ldots, V_{m-1}$ to predict V_m, the* alignment cost *of a tuple $r \in R$ is defined as*

$$\Delta(r, M) = |M(r[V_1], r[V_2], \ldots, r[V_{m-1}]) - r[V_m]|. \tag{3}$$

The larger the value alignment cost $\Delta(r, M)$ is, the more the values in tuple r are inconsistent with the learned regression model M.

Definition 4. *We say that an aligned instance R satisfies the* model constraint *δ, if $\forall r \in R$, it has $\Delta(r, M) \leq \delta$.*

A desired alignment is to obtain aligned tuples that satisfy the model constraint, i.e., consistent with M. Together with the aforesaid time constraint, we state the alignment problem as follows.

Problem 1 (Similarity Alignment under Model Constraint)**.** *Given variables $T_1, T_2, \ldots, T_m$, time constraint θ and model constraint δ, the problem of* similarity alignment under model constraint *is to*

obtain an aligned instance R, also denoted by R_{sm}, such that (1) each time attribute $U_1, U_2, \ldots, U_m$ is a candidate key of R, i.e., each tuple in $T_1, T_2, \ldots, T_m$ can only be aligned once, (2) each tuple $r \in R$ satisfies the time constraint $\Theta(r) \leq \theta$ and the model constraint $\Delta(r, M) \leq \delta$, and (3) the number of aligned tuples $|R|$ is maximized.

In this sense, to initialize M in the whole process, as illustrated in Figure 2, we use equality/similarity alignment to obtain some preliminary R for learning a regression model M to guide the subsequent alignment. Similar to the idea of EM algorithm, both the aligned instance R and the model M could be interactively improved in the iteration. (See Figure 5 in Section 4.2.1 for an empirical study.)

The definitions of equality and similarity alignment are given in Appendix A.2, where we also show that they are both special cases of the proposed similarity alignment under model constraint.

2.2 Hardness Analysis

The similarity alignment problem under model constraint is generally hard when $m \geq 3$.

Theorem 1. *Given variables $T_1, T_2, \ldots, T_m$, $m \geq 3$, time constraint θ, model constraint δ and constant κ, the problem is NP-complete to determine whether there is an aligned instance R having (1) candidate keys $U_1, U_2, \ldots, U_m$, (2) time constraint $\Theta(r) \leq \theta$ and model constraint $\Delta(r, M) \leq \delta$ satisfied for each $r \in R$, (3) $|R| \geq \kappa$.*

Proof sketch. To show NP-hardness of the problem, we build a reduction from the *maximum 3-dimensional matching* problem [17, 19, 20]. Please see [3] for proofs in detail. ∎

3 ALIGNMENT ALGORITHM

Motivated by the reduction from the maximum k-dimensional matching problem [17, 19] in the hardness analysis in Theorem 1, we devise an approximate alignment approach. A set R_c of candidate aligned tuples is first obtained, in Section 3.1, referring to time and model constraints, but with overlapping timestamps, i.e., does not meet the requirement (1) in Problem 1. In Section 3.2, a local search algorithm is then employed to eliminate the aligned tuples with overlapping timestamps. Remarkably, we show in Proposition 2 the bound of the approximation. Finally, in Section 3.3, we propose efficient pruning strategies for the algorithm. Proofs of all the theoretical results are available in [3].

3.1 Candidate Generation

Given variables $T_1, T_2, \ldots, T_m$, we generate a set R_c of candidate aligned tuples that meet the requirements of time constraint θ and model constraint δ. Rather than investigating the huge space of $T_1 \times T_2 \times \cdots \times T_m$, efficient candidate generation is possible given the ordered variables and timestamp distance $\leq \theta$.

Algorithm 1 in Appendix A.3 provides an overview of the Candidate Generation algorithm. $T_1, T_2, \ldots, T_m$, read from databases, are naturally sorted on timestamps. Referring to time constraint in Definition 2, alignment happens only among tuples with timestamp distance $\leq \theta$. Thereby, it is sufficient to consider candidate aligned tuples within a window of size θ. Motivated by the idea of merge join [37], we slide the window over $T_1, T_2, \ldots, T_m$ to generate all the possibly aligned tuples w.r.t. time constraint θ.

For each tuple $t_k \in T_k$, the algorithm specifies all the tuples $t_j \in T_j$ in the window of t_k with size θ. Combining all the t_j found in the former steps, it generates all the possible aligned tuples in the window. Likewise, we perform such a strategy for each variable T_k respectively, and thus generate all possible candidates satisfying the time constraint. The completeness and correctness of this strategy are clear. For each possible candidate r, suppose that the minimal timestamp in r appears in $U_{k_{min}}$. Then, r must be found when traversing $T_{k_{min}}$. After generating the candidates following time constraint, model constraint δ (Definition 4) is adopted to filter the candidates conflicting with the model M.

In the worst case, there will be $|T_1| \cdot |T_2| \ldots |T_m|$ aligned tuples generated. Let the prediction cost of regression model M be $c(M)$, the time complexity of Algorithm 1 is $O(|T_1| \cdot |T_2| \cdots |T_m| \cdot c(M))$.

3.2 Alignment Search

A tuple in $T_1, T_2, \ldots, T_m$ may appear in multiple candidate aligned tuples in R_c. Referring to requirement (1) in Problem 1, we need to find a subset $R_{sm} \subseteq R_c$ where each tuple in $T_1, T_2, \ldots, T_m$ is aligned at most once. Moreover, the requirement (3) in Problem 1 seeks such a subset with size $|R_{sm}|$ as large as possible.

Alignment Search algorithm employs a ρ-optimal local search strategy [7, 18], which is a heuristic algorithm adopted by many matching problems (e.g., k-set packing problem). The idea is to first initialize R_{sm} by a subset of R_c without overlapping timestamp, and then gradually extend the set by swapping more tuples in. Figure 3 presents a running example.

First, for any candidate aligned tuples $r_1 = (t_{11}, t_{21}, \ldots, t_{m1})$, $r_2 = (t_{12}, t_{22}, \ldots, t_{m2})$ in R_c, we say that r_1, r_2 overlap on some variables, denoted by $r_1 \asymp r_2$, if $\exists k, 1 \leq k \leq m, t_{k1} = t_{k2}$, i.e., a tuple is aligned more than once in r_1, r_2. In our Alignment Search algorithm, we first initialize R_{sm} with a greedy strategy, i.e., we continuously add $r_c \in R_c$ that does not overlap with any existing $r_{sm} \in R_{sm}$.

Next, given a threshold ρ of subset sizes to search, and unchosen candidates R_{unc}, i.e., $R_{unc} = R_c \setminus R_{sm}$, the algorithm searches for a tuple collection $R_{out} \subseteq R_{unc}$ with size $p = 2, 3, \ldots, \rho$ outside R_{sm} which satisfies (1) $|R_{out}| = p$, (2) any two tuples in R_{out} do not overlap with each other, i.e., $\forall r_1, r_2 \in R_{out}, r_1 \not\asymp r_2$, and (3) tuples in R_{out} overlap at most $p - 1$ tuples (denoted by R_{in}) in R_{sm}, i.e., $|R_{in}| < |R_{out}|, R_{in} = \{r_{in} | r_{in} \in R_{sm}, \exists r_{out} \in R_{out}, r_{in} \asymp r_{out}\}$. If the three conditions are satisfied, since R_{out} does not introduce further conflict, we could safely swap it into R_{sm} by removing R_{in}. R_{sm} is then expanded with an increase of $(|R_{out}| - |R_{in}|)$ on the size. When no further tuple pair could be swapped in R_{sm}, it reaches local optima and returns R_{sm}.

For each R_{out} with $|R_{out}| = p$, we conduct hash table to check the overlapping as well as find overlapping sets with $O(\rho + |R_{sm}|)$ time cost. Recall that $|R_{sm}| \leq \tau$ since a tuple in $T_1, T_2, \ldots, T_m$ can only be aligned once. That is, the total time cost of checking whether the tuple overlapping with each other in R_{out} and finding R_{in} is $O(\rho + \tau)$. For a fixed p, at most $\binom{|R_c|}{p}$ R_{out} will be checked. When the size of R_c is large, we can assume that $\rho < |R_c|/2$, thus we have $\binom{|R_c|}{p} \leq \binom{|R_c|}{\rho}$, $p \leq \rho$. Besides, the iteration (Line 8) will run ρ times, thus the total time cost will be $O(\rho\binom{|R_c|}{\rho}(\rho + \tau))$. The searching will stop when reaches local optima. Again, since the iteration either increases the size of R_{sm} or stops, and the size of R_{sm}

is bounded by τ, the iteration performs (Line 6) at most τ times. To conclude, with a given moderately large ρ, the algorithm thus runs in $O(\tau\rho\binom{|R_c|}{\rho}(\rho+\tau))$ time. Algorithm 2 gives the pseudocode.

Proposition 2. *Given m variables $T_1, T_2, \ldots, T_m$, $m \geq 3$, and threshold ρ of subset size, the approximation ratio ξ of similarity alignment under model constraint by Algorithms 1 and 2 is bounded by*

$$\xi = \frac{m(m-1)^{\frac{\rho+1}{2}} - m}{2(m-1)^{\frac{\rho+1}{2}} - m}, \qquad \text{if } \rho = 3, 5, 7, \ldots \tag{4}$$

$$\xi = \frac{m(m-1)^{\frac{\rho}{2}} - 2}{2(m-1)^{\frac{\rho}{2}} - 2}, \qquad \text{if } \rho = 2, 4, 6, \ldots \tag{5}$$

Recall that the parameter ρ is the largest size of the sets we consider for swapping. Intuitively, a larger ρ contributes to more accurate alignment results. In practice, a moderately large ρ (e.g., 2 or 3) is sufficient. Theoretically, according to Proposition 2, given $m = 3$, $\rho = 2$, we have $\xi = 2$, i.e., factor-2 approximation. Empirically, the alignment accuracy is already high (F1-score > 0.9) when given $\rho = 2$ in the experiments in Figure 6 in Section 4.2.2. In such a scenario with $\rho = 2$, Algorithm 2 runs in $O(\tau^2|R_c|^2)$ time.

3.3 Optimization and Pruning

As shown in Section 3.2, the complexity of our proposed Alignment Search algorithm is $O(\tau\rho\binom{|R_c|}{\rho}(\rho+\tau))$, which runs in polynomial w.r.t $|R_c|$ and τ. However, in practice, the amount of candidates is significantly larger than the length of the multiple variables, i.e., $|R_c| \gg \tau$. Even given a small threshold ρ, the size of $\binom{|R_c|}{\rho}$ is still out of control. When $\rho = 2$, Algorithm 2 still runs in $O(\tau^2|R_c|^2)$ time. Since the unchosen candidates have $R_{unc} = R_c \setminus R_{sm}$, and $R_{sm} \leq \tau$, it also indicates $|R_{unc}| \gg |R_{sm}|$.

In this section, to further optimize and prune the Alignment Search algorithm, we start from two concerns. (1) According to Section 3.2, in each iteration, Alignment Search algorithm traverses all the subsets of unchosen candidates R_{unc} with size p, leading to $\binom{|R_c|}{p}$ time cost. Since $|R_c| \gg \tau$ and $|R_{sm}| \leq \tau$, is it possible to traverse the subsets of R_{sm} instead? (2) Rather than traversing all the subsets of R_{sm}, is there any strategy to prune some subsets before finding swaps for them in traversing?

3.3.1 Traversing R_{sm} Instead of R_{unc}. For the first concern, given currently chosen candidates R_{sm}, unchosen candidates $R_{unc} = R_c \setminus R_{sm}$ and optimal parameter ρ, we thus traverse the subsets R_{in} of R_{sm} with size $p = 1, 2, \ldots, \rho - 1$ instead. The algorithm then searches for a set $R_{out} \subseteq R_{unc}$ that satisfies (1) $|R_{out}| \geq p + 1$, and conditions (2) and (3) in Section 3.2. An example is illustrated in Figure 3. However, the problem is how to find the possible R_{out} without traversing all possible subsets again in R_{unc}.

To address the aforesaid problem, we discover that, if R_{out} could be swapped with R_{in}, besides conditions (1), (2) and (3), each tuple in R_{out} overlaps with at least one tuple in R_{in}, and each tuple in R_{in} overlaps with at least one tuple in R_{out} (will be proved below in Proposition 3). This observation could be further leveraged to prevent the time-consuming traversal.

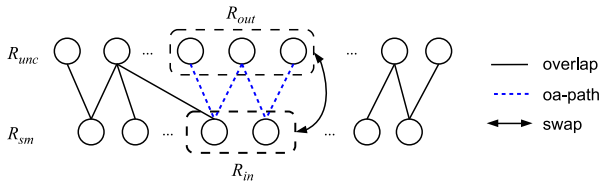


Figure 3: An example of Alignment Search algorithm. R_{sm} and R_{unc} are the currently chosen and unchosen candidates, respectively. R_{in} is a subset of R_{sm} with size $p = 2$ and R_{out} is a subset of R_{unc} with size $> p$. Edges denote the overlapping relationships between two tuples (vertices). The blue dotted lines form an oa-path, where tuples in R_{in} and R_{out} alternatively occur. The line with arrows is an attempt of swapping.

Proposition 3. *Given R_{unc}, R_{sm} and p, if $R_{out} \subseteq R_{unc}$ could be swapped with $R_{in} \subseteq R_{sm}$ having $|R_{in}| = p$, it should also satisfy following conditions:*

(4) $\forall r_{out} \in R_{out}, \exists r_{in} \in R_{in}, r_{out} \asymp r_{in}$.
(5) $\forall r_{in} \in R_{in}, \exists r_{out} \in R_{out}, r_{out} \asymp r_{in}$.

Algorithm 3 in Appendix A.3 outlines the improved algorithm. With conditions (4) and (5) in Proposition 3, we could search for r_{out} that overlaps with tuples in R_{in}, without traversing all the subsets of R_{unc}. The searching of the overlapping tuples could be prebuilt by hash map. Since $|R_{sm}| \leq \tau$, the time complexity of searching for possible swapping could be reduced from $O(\rho\binom{|R_c|}{\rho}(\rho+\tau))$ Lines 8-15 of Algorithm 2, to $O(\rho\binom{\tau}{\rho-1}|R_c|\tau)$, in Lines 8-14 of Algorithm 3. For a small ρ, we have $O(\rho\binom{|R_c|}{\rho}(\rho+\tau)) \approx O(\rho|R_c|^\rho(\tau))$ and $O(\rho\binom{\tau}{\rho-1}|R_c|\tau) \approx O(\rho\tau^\rho|R_c|)$. Since $|R_c| \gg \tau$, Algorithm 3 significantly reduces the time complexity compared to Algorithm 2.

3.3.2 Pruning with OA-Path and Extended Time Constraint. For the second concern, motivated by the alternating path approach for bipartite graph matching, we find analogous alternating path in our problem, namely overlapping alternating path (oa-path in Definition 5). That is, $r_{in} \in R_{in}$ and $r_{out} \in R_{out}$ alternatively occur in the path, as illustrated in Figure 3. By introducing oa-path, we (1) extend the time constraint in Definition 2 to the extended time constraint over oa-path (i.e., a set of tuples) in Lemma 4, and (2) prove that for any two tuples $r_1, r_2 \in R_{in}$, there exists an oa-path with length $\leq 2p - 1$ between them in Lemmas 5 and 6.

Combining both findings, Proposition 7 proves that the time distance of any two tuples in R_{in} is within $(2p-1)\theta$. We thus employ Proposition 7 for pruning R_{in} in Algorithm 3, which significantly reduces the time cost of the Alignment Search algorithm.

Definition 5. *(Overlapping alternating path, oa-path). Given R_{in} and R_{out}, an overlapping alternating path (oa-path) of length l is defined as $(r_1, r_2, \ldots, r_l)$, having (1) $r_i \asymp r_{i+1}$, and (2) r_i and r_{i+1} belong to R_{in} and R_{out} (or R_{out} and R_{in}), respectively, $\forall i \in \{1, 2, \ldots, l-1\}$.*

In short, an oa-path is a list of tuples where consecutive tuples overlap with each other, and tuples from R_{in} and R_{out} alternatively occur. Figure 3 shows an example, where vertices are the tuples $r \in R_{in} \cup R_{out}$. There is an edge between them, if $r_i \asymp r_j$. The blue edges form an oa-path. While the time constraint is originally defined on tuple pairs in Definition 2, we extend it to restrict the time differences of the start and end vertices over oa-path for pruning.

Lemma 4. *(Extended time constraint over oa-path). For an oa-path $(r_1, r_2, \ldots, r_l)$, the differences between any timestamps of r_1 and r_l are less than $l\theta$, i.e., $\forall k \in \{1, 2, \ldots, m\}, |t_{k1}[U_k] - t_{kl}[U_k]| \leq l\theta$.*

To apply the extended time constraint in Lemma 4, we identify the oa-path in the graph w.r.t. tuple overlaps $\asymp$.

Lemma 5. *For any two tuples $r_a, r_b \in R_{in} \cup R_{out}$, if r_a and r_b are connected by a path, the path connecting them must be an oa-path.*

Lemma 5 states that any path connecting tuples in R_{in} and R_{out} must be an oa-path, which ensures the correctness of utilizing oa-path for pruning. We further investigate its length.

Lemma 6. *Given R_{in}, R_{out} and $|R_{in}| = p$, assuming that the algorithm has found all possible swaps with $|R'_{in}| < p$, for $|R_{in}| = p$, if R_{in} could be swapped by R_{out}, $\forall r_a, r_b \in R_{in}$, there must exist an oa-path $P_{oa} = (r_a, \ldots, r_b)$ with length $\leq 2p - 1$.*

Lemma 6 shows that any two tuples $r_a, r_b \in R_{in}$ are connected by an oa-path, with a bounded length. Combining with Lemma 4 about the extended time constraint over oa-path, we finally apply the extended time constraint to tuple pairs of R_{in} in Proposition 7.

Proposition 7. *Given R_{in} and $|R_{in}| = p$, if R_{in} is possible to be swapped by $R_{out} \subseteq R_{unc}$, we have $|t_{ka}[U_k] - t_{kb}[U_k]| \leq (2p - 1)\theta$, $\forall r_a, r_b \in R_{in}$, $k \in \{1, 2, \ldots, m\}$, i.e., the timestamp differences timestamps any r_a and r_b are less than $(2p - 1)\theta$.*

Algorithm 3 presents the optimized Alignment Search with pruning. According to Proposition 7, we prune unnecessary R_{in} by simply checking the time differences of every tuple pair in R_{in} (Line 10). The time cost of searching possible swapping is further reduced from $O(\rho\binom{\tau}{\rho-1}|R_c|\tau)$ in Section 3.3.1 to $O(\rho\binom{\tau}{\rho-1}\rho m)$. Again, as discussed by the end of Section 3.2, a moderately large $\rho = 2$ is sufficient, leading to the time complexity $O(\tau m)$ of Algorithm 3. As illustrated in Section 4.2.2, time performance is significantly improved by the proposed strategies in practice.

4 EXPERIMENTS

In this section, we evaluate our proposal by comparing with the state-of-the-art approaches over real-world datasets. The experiment settings are in Appendix A.4.

4.1 Comparison with Existing Methods

For each dataset, we compare our SAMC with the existing alignment methods listed in Appendix A.4.3 over different regression models introduced in Appendix A.4.2. Table 1 reports the alignment F1-score and model RMSE. To demonstrate that the differences in algorithms are significant, the experiments are conducted 5 times. The best performances based on Student's Paired t-test [29] at 95% significance level (i.e., $p < 0.05$) are bold in Table 1.

For alignment accuracy, the results of baselines do not change with models, since their alignment leverages only similarities without considering the regression models. As shown, SAMC outperforms all the baselines over five datasets on alignment F1-score. The result is not surprising, since as introduced in Section 1 the similarity of variables can be captured by a special regression model

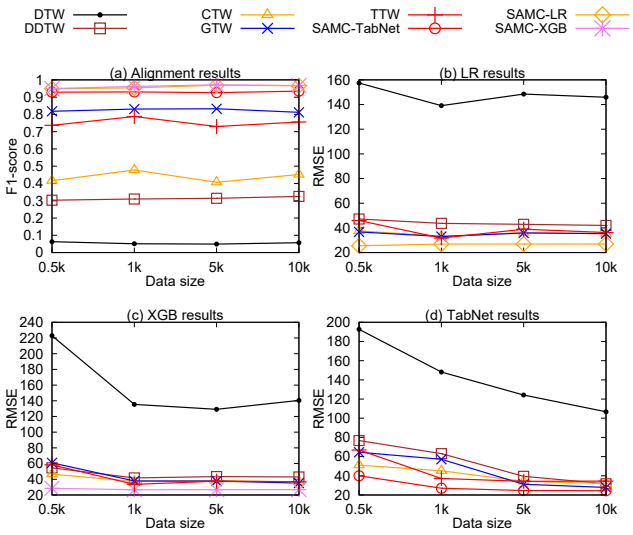


Figure 4: Comparing our SAMC with the existing alignment methods over Fuel dataset on (a) F1-score of the alignment results, (b) RMSE loss of the LR model, (c) RMSE loss of the XGB model and (d) RMSE loss of the TabNet model

on variables with similar values. In contrast, the more general regression on distinct variables is not considered in the existing alignment methods. It verifies the intuition of our study, regression model indeed guides the alignment of multiple variables.

Table 1 also reports the model RMSE over test data. In general, more accurately aligned data lead to better model learning performance. Therefore, our proposal achieves the lowest model RMSE. Compared to the models trained on interpolated time series in Table 1, our proposal also shows competitive performance. The result is not surprising since the interpolation does not consider the timestamp variations, but interpolating new values that are error prone, as the example illustrated in Figure 1(b) in the Introduction.

Figure 4 reports the alignment and model accuracies by varying data sizes (i.e., sequence length). The results are stable over various data sizes and generally similar to Table 1.

4.2 Evaluation of Proposed Techniques

This experiment evaluates the effectiveness of applying iteration, optimization and pruning strategies. Due to the limited space, we report similar results on all datasets in [3].

4.2.1 Varying the Number of Iterations. As illustrated in Figure 2 and discussed at the end of Section 2.1, the aligned instances by equality R_e or similarity R_s can serve in initialization to obtain a preliminary regression model M. It is then used in the similarity alignment under model constraint, known as M initialized by R_e and M initialized by R_s, respectively. The new predicted model M' can further be used in the model constraint in another iteration of similarity alignment under model constraint M'.

Figure 5 presents the results under various number of iterations. It is not surprising that using similarity alignment R_s to initialize M has better performance.More iterations lead to better performance.

Table 1: Comparison to existing methods over five datasets. The best performances based on Student's Paired t-test at 95% significance level (i.e., $p < 0.05$) are bold.

Dataset	Model	Alignment F1-score						Regression RMSE on Test Data						
		DTW	DDTW	CTW	GTW	TTW	SAMC	DTW	DDTW	CTW	GTW	TTW	SAMC	Interpolation
House	LR	0.222	0.024	0.007	0.675	0.049	**0.929**	17.937	5.223	3.825	**3.359**	**3.390**	**3.222**	3.690
	XGB	0.222	0.024	0.007	0.675	0.049	**0.932**	29.367	8.594	5.464	5.015	4.884	**4.337**	4.659
	TabNet	0.222	0.024	0.007	0.675	0.049	**0.928**	27.465	7.778	5.232	**4.519**	5.190	**4.152**	4.714
Telemetry	LR	0.547	0.301	0.580	0.148	0.502	**0.829**	0.255	0.253	0.257	0.259	0.256	**0.221**	0.283
	XGB	0.547	0.301	0.580	0.148	0.502	**0.889**	0.169	0.171	0.170	0.171	**0.165**	**0.157**	0.240
	TabNet	0.547	0.301	0.580	0.148	0.502	**0.855**	0.240	0.231	0.248	**0.224**	0.267	**0.212**	0.263
Water	LR	0.140	0.629	0.092	0.061	0.002	**0.948**	0.239	0.210	0.204	0.195	1.081	**0.057**	0.279
	XGB	0.140	0.629	0.092	0.061	0.002	**0.964**	0.037	0.037	0.036	0.036	0.173	**0.031**	0.051
	TabNet	0.140	0.629	0.092	0.061	0.002	**0.939**	0.106	0.072	0.085	0.112	1.400	**0.038**	0.059
Air Quality	LR	0.190	0.056	0.048	0.037	0.001	**0.693**	50.571	47.215	43.133	50.146	49.072	**35.670**	38.056
	XGB	0.190	0.056	0.048	0.037	0.001	**0.731**	44.748	43.907	40.608	39.231	54.105	**37.281**	38.409
	TabNet	0.190	0.056	0.048	0.037	0.001	**0.727**	55.836	78.012	66.054	67.821	83.836	**50.045**	54.742
Fuel	LR	0.057	0.325	0.452	0.812	0.788	**0.966**	139.077	37.881	31.075	30.926	31.801	**25.132**	71.647
	XGB	0.057	0.325	0.452	0.812	0.788	**0.967**	151.752	40.515	33.410	32.260	33.558	**24.636**	181.011
	TabNet	0.057	0.325	0.452	0.812	0.788	**0.935**	154.779	38.814	33.137	31.673	37.215	**24.585**	54.154

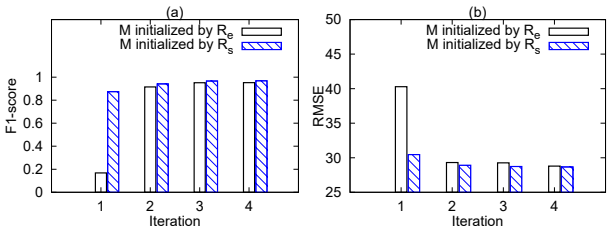


Figure 5: Alignment accuracy and model prediction performance by multiple iterations of similarity alignment under time constraint $\theta = 120$ and model constraint $\delta = 35$ on Fuel

The results do not further improve by conducting even more iterations, i.e., converge. That is to say, most of the aligned tuples conform to the regression and with similar timestamps, even though some of them are not in ground truth.

4.2.2 Evaluating Optimization and Pruning Strategy. Table 2 reports the running time of the Alignment Search algorithm with different optimization and pruning strategies. We conduct experiments over dataset House with different lengths τ of variables. AS is the original Alignment Search algorithm. AS+Opt represents the Alignment Search with optimization strategy introduced in Section 3.3.1. AS+Opt+Pruning represents the Alignment Search with both optimization and pruning strategies in Section 3.3.2.

Table 2 shows that the optimization strategy reduces the time cost of the original algorithm. The pruning strategy does not perform when $\rho = 2$, i.e., there is only one tuple in R_{in}. Nevertheless, for $\rho > 2$, the pruning strategy significantly improves the efficiency (reducing 85% time when $|R_c| \approx 10^4$).

Figure 6 illustrates the alignment performance of our proposal by varying ρ, the largest size of the sets we will consider for swapping. The results verify that a moderately large ρ is sufficient as discussed at the end of Section 3.2, since the performance of the proposal with $\rho > 2$ only shows slight improvement while suffering from a loss of efficiency as also shown in Table 2.

Table 2: Running time (s) of Alignment Search algorithm using different optimization and pruning strategies over House dataset with $\theta = 80$ and $\delta = 8$

ρ	τ	$\lvert R_c \rvert$	AS	AS+Opt	AS+Opt+Pruning
2	100	986	1.56	**0.03**	**0.03**
3	100	986	525.93	0.17	**0.06**
2	1000	9686	112.02	**0.32**	**0.32**
3	1000	9686	> 1 hour	15.61	**2.17**

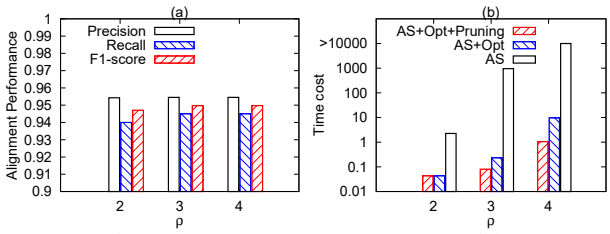


Figure 6: (a) Alignment performance and (b) time cost of SAMC by varying ρ over House dataset with $\theta = 80$ and $\delta = 8$

4.3 Automatic Constraint Determination

In this section, we first illustrate how the time constraint θ and model constraint δ affect the alignment and model learning performance. Then, we introduce the automatic determination of proper θ and δ in practice without the ground truth.

4.3.1 Varying Time Constraint θ. (For each fixed δ,) it is not surprising that increasing θ leads to more aligned tuples in Figure 7(b). More tuple pairs with larger timestamp distances can meet

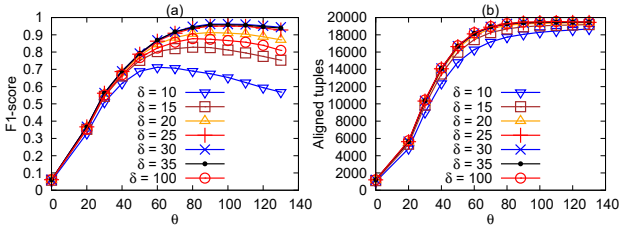


Figure 7: (a) Alignment accuracy and (b) aligned tuple amount under various time constraints θ and model constraints δ on Fuel dataset

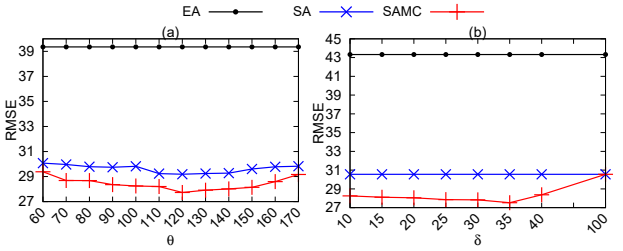


Figure 8: Model accuracy under (a) various time constraints θ with a fixed $\delta = 35$, and (b) various model constraints δ with a fixed $\theta = 100$ over Fuel data using XGB

the requirement of a time constraint with a larger distance threshold θ. Figure 7(a) illustrates that by increasing θ, the F1-score first increases and then drops. For a small θ (the extreme case is equality alignment with $\theta = 0$), a limited number of tuples are aligned, leading to low alignment recall. On the other hand, for a large θ, too distant tuples not occurring in the same period are mistakenly aligned, with low accuracy. Thereby, a moderately large time constraint θ is expected (see Section 4.3 on determining proper θ).

Figure 8(a) presents the results of model accuracy by SA and SAMC under various time constraints θ. The result of EA with a fixed $\theta = 0$ is also reported as the baseline. The model performance is generally analogous to the alignment accuracy in Figure 7(a). EA with $\theta = 0$ has low alignment F1-score and high model prediction RMSE. Time constraints θ with better alignment F1-score, e.g., in the range of $[90, 130]$ for $\delta = 35$ in Figure 7(a), show lower model prediction RMSE as well in Figure 8(a).

4.3.2 Varying Model Constraint δ. For a small δ, i.e., strict model constraint, fewer tuples can meet the requirement as shown in Figure 7(b). Again, a small δ leading to a limited number of aligned tuples has low recall. For a large δ (the extreme case is similarity alignment without model constraint $\delta = +\infty$), aligned tuples do not conform to the model M will be considered, leading to low alignment precision. As shown in Figure 7(a), on Fuel dataset, δ in a range of $[25, 35]$ shows the best alignment accuracy (also see the determination of δ in Section 4.3).

Likewise, Figure 8(b) reports equality and similarity alignment with $\delta = +\infty$ as the baselines. Again, SAMC under a model constraint $\delta = 35$, showing better alignment accuracy in Figure 7(a), leads to lower model RMSE in Figure 8(b). We observe that a small δ leads to an inaccurate model. It is not surprising since a small δ ignores too many alignments and thus is insufficient for learning.

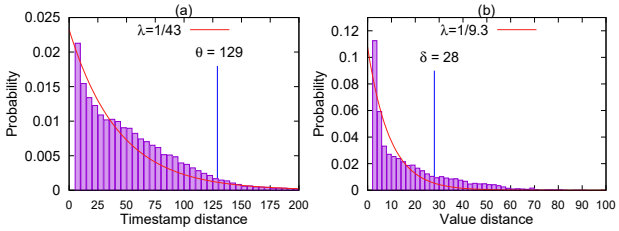


Figure 9: Distribution of (a) timestamp distances on consecutive tuples in dataset Fuel, which automatically determine time constraint $\theta = 129$, and (b) value distances between aligned candidates and their model predictions, which automatically determine model constraint $\delta = 28$

4.3.3 Determining Time Constraint θ. To determine a proper threshold θ of timestamp distances between aligned tuples, we observe the distribution on timestamp distances of consecutive tuples in the input dataset. Interpreting consecutive tuples as the nearest neighbors on timestamps, we can use a Poisson process to approximate the appearance of the nearest neighbors [27]. That is, the timestamp distances of consecutive tuples are modeled with an exponential distribution, denoted by $d_U \sim Exp(\lambda)$, having probability density function (PDF) $f(d_U \mid \lambda) = \lambda \exp(-\lambda d_U)$. The parameter λ is estimated by $\lambda \approx \frac{1}{MTI}$, where MTI is the median of all timestamp distances. Figure 9(a) shows the distribution over the dataset used in the experiment with estimated parameter $\lambda = \frac{1}{43}$.

Once timestamp distances of consecutive tuples are modeled by an exponential distribution, we determine the threshold θ of timestamp distances that can be aligned under confidence level 0.95 [33], i.e., find a θ having cumulative distribution function (CDF) $F(\theta \mid \lambda) = 0.95$. For the distribution in Figure 9(a), we determine $\theta = 129$. As in Figures 7(a) and 8(a), such a time constraint leads to better alignment accuracy and model prediction performance.

4.3.4 Determining Model Constraint δ. Similarly, to find a proper threshold of model constraint δ, we observe the value distances between aligned candidate tuples r and the predictions of M, i.e., $\Delta(r, M)$ in Formula 3. Again, we can model the distances on nearest neighbors by an exponential distribution [27], $d_V \sim Exp(\lambda)$, having probability density function (PDF) $f(d_V \mid \lambda) = \lambda \exp(-\lambda d_V)$. Figure 9(b) presents the distribution observed in dataset Fuel with estimated parameter $\lambda = \frac{1}{9.3}$. With confidence level 0.95 [33], we determine δ=28. As shown in Figures 7(a) and 8(b), the corresponding alignment accuracy is high, while the model RMSE is low. Moreover, if the dependent variable could not be predicted, i.e., the regression relationship does not hold, the problem of similarity alignment under model constraint (in Section 2.1) is equivalent to the problem of similarity alignment, i.e., exactly the mentioned simple approach of only comparing the timestamp. In such scenarios, we can set δ to $+\infty$ and the proposal still works. Since the parameters θ and δ are determined by observing the data distributions, the proposal could tolerate noise to some extent.

5 RELATED WORK

In addition to the typical methods compared in the experiments in Section 4.1 (and introduced in Appendix A.4.3), similar approaches could also be used for aligning time series. ERP [9] and EDR [10]

utilize edit distance with dynamic programming to evaluate the similarities of time series. LCSS [32] computes the similarity of sequences in a bounded time window, and considers the similarity of the values. Although most of these methods do not require a bounded matching window, they highly rely on the similarities among time series. Unfortunately, such proximities of similar values and timestamps are very likely to be absent among the variables in regression, e.g., in Figure 1. For the same reason, the approaches [12, 16] also emphasizing similarities among a cluster of time series do not apply.

Apart from warping, Dignös et al. [13] study temporal alignment, and propose a relational algebra for providing sequenced semantics in DBMS. However, it cannot be applied to our situation, since (1) the method deals with time durations, e.g., the period of a stay in the hotel, rather than discrete time, e.g., the instant data collected from sensors, and (2) it focuses on the query operations between two temporal relations rather than multiple time series.

Unlike time lag discovery problems [31] that indicates hidden temporal dependencies, we study unaligned variables that are caused by unexpected timestamp variations such as transmission delays. Such delays cannot be modeled as temporal dependencies. To learn a high-quality regression model for downstream tasks, it is possible and necessary for the alignment over variables.

The regression among multiple aligned variables further guides the alignment in the following alignment iteration, analogous to the temporal constraints for time series data cleaning [30]. To handle the inconsistency of the aligned instance R to the model, we employ the model M as the reference data. The idea is similar to data repairing with master data [14]. While Fan et al. [14] propose to modify the data values to eliminate violations, our study finds alignment variables that conforms to M.

6 CONCLUSION

In this paper, to align tuples for learning regression models on the corresponding variables, we propose a novel similarity alignment under model constraint (SAMC). While the time constraints ensure that the values in an aligned tuple are temporally close with similar timestamps in variables, the model constraints further rule out the alignment inconsistent with the (iteratively learned) regression model. We prove NP-completeness of the alignment problem and propose an approximation algorithm with theoretical performance guarantee. Novel pruning strategies are proposed to improve efficiency. Experiments over real datasets demonstrate that by our proposal, aligning tuples and learning regression model improve with each other in iteration. The improvement verifies the necessity of aligning tuples for learning regression model.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (62072265, 62021002), the National Key Research and Development Plan (2021YFB3300500, 2019YFB1705301, 2019YFB1707001), Beijing National Research Center for Information Science and Technology (BNR2022RC01011), and Alibaba Group through Alibaba Innovative Research (AIR) Program. Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

REFERENCES

[1] http://iotdb.apache.org.
[2] https://github.com/apache/iotdb/tree/research/alignment-model.
[3] https://github.com/fangfcg/SAMC.
[4] Dynamic time warping (DTW). In S. Z. Li and A. K. Jain, editors, *Encyclopedia of Biometrics*, page 231. Springer US, 2009.
[5] Scikit-learn, https://scikit-learn.org/stable/.
[6] S. O. Arık and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, pages 6679–6687. AAAI Press, 2021.
[7] E. M. Arkin and R. Hassin. On local search for weighted k-set packing. *Math. Oper. Res.*, 23(3):640–648, 1998.
[8] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro. Messing up with BART: error generation for evaluating data-cleaning algorithms. *Proc. VLDB Endow.*, 9(2):36–47, 2015.
[9] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803. Morgan Kaufmann, 2004.
[10] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD Conference*, pages 491–502. ACM, 2005.
[11] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.
[12] D. Chudova, S. Gaffney, E. Mjolsness, and P. Smyth. Translation-invariant mixture models for curve clustering. In *KDD*, pages 79–88. ACM, 2003.
[13] A. Dignös, M. H. Böhlen, and J. Gamper. Temporal alignment. In *SIGMOD Conference*, pages 433–444. ACM, 2012.
[14] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *Proc. VLDB Endow.*, 3(1):173–184, 2010.
[15] A. Furlani Bastos, S. Santoso, V. K. Krishnan, and Y. Zhang. Machine learning-based prediction of distribution network voltage and sensors allocation. Technical report, National Renewable Energy Lab. (NREL), 2020.
[16] S. Gaffney and P. Smyth. Joint probabilistic curve clustering and alignment. In *NIPS*, pages 473–480, 2004.
[17] P. Haxell and L. Narins. A stability theorem for matchings in tripartite 3-graphs. *Combinatorics, Probability & Computing*, 27(5):774–793, 2018.
[18] C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discret. Math.*, 2(1):68–72, 1989.
[19] V. Kann. Maximum bounded 3-dimensional matching is MAX snp-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.
[20] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.
[21] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *SDM*, pages 1–11. SIAM, 2001.
[22] S. Khorram, M. G. McInnis, and E. M. Provost. Trainable time warping: Aligning time-series in the continuous-time domain. In *ICASSP*, pages 3502–3506, 2019.
[23] M. Lepot, J.-B. Aubin, and F. H. Clemens. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, 9(10):796, 2017.
[24] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.
[25] S. E. Marx, J. D. Luck, S. K. Pitla, and R. M. Hoy. Comparing various hardware/software solutions and conversion methods for controller area network (can) bus data collection. *Computers and Electronics in Agriculture*, 128:141–148, 2016.
[26] Y. Min, Y. Xiaogang, and J. Shengjie. Energy optimization by parameter matching for a truck-mounted concrete pump. *Energy Procedia*, 88:574–580, 2016.
[27] Y. Noh, F. C. Park, and D. D. Lee. Diffusion decision making for adaptive k-nearest neighbor classification. In *NIPS*, pages 1934–1942, 2012.
[28] P. Philipp and S. Altmannshofer. Experimental validation of a new moving horizon estimator approach for networked control systems with unsynchronized clocks. In *ACC*, pages 4939–4944. IEEE, 2012.
[29] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, pages 623–632, 2007.
[30] S. Song, A. Zhang, J. Wang, and P. S. Yu. SCREEN: stream data cleaning under speed constraints. In *SIGMOD Conference*, pages 827–841. ACM, 2015.
[31] L. Tang, T. Li, and L. Shwartz. Discovering lag intervals for temporal dependencies. In *KDD*, pages 633–641. ACM, 2012.
[32] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684. IEEE Computer Society, 2002.
[33] J. H. Zar. *Biostatistical analysis*. Pearson Education India, 1999.
[34] F. Zhou and F. D. la Torre. Canonical time warping for alignment of human behavior. In *NIPS*, pages 2286–2294. Curran Associates, Inc., 2009.
[35] F. Zhou and F. D. la Torre. Generalized time warping for multi-modal alignment of human motion. In *CVPR*, pages 1282–1289. IEEE Computer Society, 2012.
[36] F. Zhou and F. D. la Torre. Generalized canonical time warping. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):279–294, 2016.
[37] J. Zhou. Sort-merge join. In *Encyclopedia of Database Systems (2nd ed.)*. Springer, 2018.

A SUPPLEMENTARY MATERIAL

A.1 Notations

Table 3 summarizes some frequently used notations in this paper.

Table 3: Notations

Symbol	Description
$T_1, T_2, \ldots, T_m$	m variables
τ	the minimum number of values of each variable
θ, δ	thresholds of time and model constraints
R_e, R_s, R_{sm}	tuples aligned by EA, SA and SAMC
R_c, R_{unc}	all candidates and unchosen candidates
R_{in}, R_{out}	subsets of R_{sm} and R_{unc} to be swapped
ρ	threshold of searched subset size

A.2 Special Cases

Problem 1 defines the similarity alignment under model constraint (SAMC). Besides the time constraint in Definition 2, which restricts the similar timestamps, the model constraint in Definition 4 ensures $\Delta(r, M) \leq \delta$ for each $r \in R$.

As we introduced in Section 2, equality alignment, a natural idea of aligning tuples, is based on equal timestamps. Similarity alignment employs time constraint, to further enrich the training data. Indeed, we will prove that they are both special cases of the proposed problem. According to the similarity alignment under model constraint (SAMC), we will give the definitions of the equality alignment (EA) problem and the similarity alignment (SA) problem. An example of their relationships and the pipeline of employing the methods is illustrated in Figure 2.

The equality alignment (EA) problem enforces the same timestamps in aligned tuples, which is an intuitive alignment strategy. The definition of equality alignment is provided as follows.

Problem 2 (Equality Alignment)**.** *Given variables $T_1, T_2, \ldots, T_m$, the* equality alignment *problem is to obtain an aligned instance R, namely R_e, such that (1) each time attribute U_i is a candidate key of R, i.e., each tuple in $T_1, T_2, \ldots, T_m$ can only be aligned once, (2) each tuple $r \in R$ has equal timestamp $r[U_1] = r[U_2] = \cdots = r[U_m]$, and (3) the number of aligned tuples $|R|$ is maximized.*

Solving the equality alignment problem is trivial. By performing join operations $R = T_1 \bowtie T_2 \bowtie \ldots \bowtie T_m$ on $U_1 = U_2 = \cdots = U_m$, it naturally maximizes the number of aligned tuples $|R|$.

The Similarity Alignment (SA) problem employs time constraint alone to filter the aligned tuples.

Problem 3 (Similarity Alignment)**.** *Given variables $T_1, T_2, \ldots, T_m$ and time constraint θ, the* similarity alignment *problem is to obtain an aligned instance R, namely R_s, such that (1) each time attribute $U_1, U_2, \ldots, U_m$ is a candidate key of R, i.e., each tuple in $T_1, T_2, \ldots, T_m$ can be aligned once, (2) each tuple $r \in R$ satisfies the time constraint $\Theta(r) \leq \theta$, and (3) the number of aligned tuples $|R|$ is maximized.*

Rather than equal timestamps in Equality Alignment, the time constraint in Similarity Alignment, defined as time constraint $\Theta(r) \leq \theta$ in Definition 2, considers similar timestamps. It ensures that in each aligned tuple $r \in R$, the distances between any two timestamps are no greater than θ. To solve the similarity alignment problem, our proposed Candidate Generation (Algorithm 1) and Alignment Search (Algorithm 2) are also applied, by ignoring the model constraint part, as well as the pruning strategies in Algorithm 3.

Proposition 8. *Given $\delta = +\infty$, i.e., no model constraint, Problem 1 is equivalent to the similarity alignment. Together with $\delta = 0$, it is indeed the equality alignment.*

A.3 Algorithms

Algorithm 1 provides an overview of the candidate generation. Let $T_1, T_2, \ldots, T_m$ be naturally sorted on timestamps. The algorithm first specifies all the tuples $t_j \in T_j$ in the window of t_k with size θ (Line 3). Model constraint δ is adopted to filter the candidates conflicting with the model M (Line 6). As aforementioned, in the whole process, equality/similarity alignment are used to initialized preliminary R for learning a regression model M to guide the subsequent alignment (illustrated in Figure 2).

Algorithm 2 presents the Alignment Search algorithm. Line 3 initializes R_{sm} with a greedy strategy, i.e., we continuously add $r_c \in R_c$ that does not overlap with any existing $r_{sm} \in R_{sm}$. Lines 10-12 filter possible swapping by the conditions and generate all the possible aligned tuples in the time window that conform to the model M. Line 14 finally conducts the swapping. It follows the framework of ρ-optimal local search algorithm for k-SP problem [18, 24], which has a bounded approximation ratio (Proposition 2).

Algorithm 3 outlines the Alignment Search algorithm with pruning strategies. Line 9 is proved by Proposition 3, to traverse R_{sm} instead of R_{unc}. Line 10 employs oa-path with extended time constraint to further filter the candidates, proved in Proposition 7.

Algorithm 1: Candidate Generation ($T\_List, \theta, M, \delta$)

Input: variables list $T\_List = (T_1, T_2, \ldots, T_m)$, time constraint θ, model constraint (M, δ)
Output: candidates of aligned tuples R_c

1 $R_c \leftarrow \emptyset$;
2 **for** *each* $T_k \in T\_list$ **do**
3 $\quad R_k \leftarrow \{(t_1, t_2, \ldots, t_m) | \forall 1 \leq i \leq m, t_i \in T_i, t_k[U_k] \leq t_i[U_i] \leq t_k[U_k] + \theta\}$;
4 $\quad R_c \leftarrow R_c \cup R_k$;
5 **for** *each* $r \in R_c$ **do**
6 $\quad$ **if** $\Delta(r, M) > \delta$ **then**
7 $\quad\quad R_c \leftarrow R_c \setminus \{r\}$
8 **return** R_c;

A.4 Experiment Settings

A.4.1 Datasets. (1) House [2] is about electric power consumption in one household. The regression model is built on active power, reactive power and intensity attributes to predict voltage. (2) Telemetry [3] is from environmental sensor telemetry data. The regression model is built on carbon monoxide, humidity and smoke attributes

[2] https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption
[3] https://www.kaggle.com/garystafford/environmental-sensor-data-132k

Algorithm 2: Alignment Search (R_c, ρ)

Input: aligned candidates R_c, threshold ρ
Output: aligned tuples R_{sm} without overlapping timestamp

$R_{sm} \leftarrow \emptyset$;
for $r_c \in R_c$ **do**
  if $\forall r_{sm} \in R_{sm}, r_c \not\asymp r_{sm}$ **then**
    $R_{sm} \leftarrow R_{sm} \cup \{r_c\}$;
local optimal $\leftarrow$ False;
while *not local optimal* **do**
  local optimal $\leftarrow$ True;
  for $2 \le p \le \rho$ **do**
    $R_{unc} \leftarrow R_c \setminus R_{sm}$;
    for $R_{out} \subseteq R_{unc}, |R_{out}| = p$ **do**
      if $\forall r_a, r_b \in R_{out}, r_a \not\asymp r_b$ **then**
        $R_{in} \leftarrow \{r_{in} \in R_{sm} | \exists r_{out} \in R_{out}, r_{out} \asymp r_{in}\}$;
        if $|R_{in}| \le p - 1$ **then**
          $R_{sm} \leftarrow R_{sm} \setminus R_{in} \cup R_{out}$;
          local optimal $\leftarrow$ False;
return R_{sm};

Algorithm 3: Alignment Search-Pruning (R_c, ρ, θ)

Input: aligned candidates R_c, threshold ρ, time constraint θ
Output: aligned tuples R_{sm} without overlapping timestamp

$R_{sm} \leftarrow \emptyset$;
for $r_c \in R_c$ **do**
  if $\forall r_{sm} \in R_{sm}, r_c \not\asymp r_{sm}$ **then**
    $R_{sm} \leftarrow R_{sm} \cup \{r_c\}$;
local optimal $\leftarrow$ False;
while *not local optimal* **do**
  local optimal $\leftarrow$ True;
  for $1 \le p \le \rho - 1$ **do**
    for $R_{in} \subseteq R_{sm}, |R_{in}| = p$ **do**
      if $\forall r_a \in R_{in}, \forall r_b \in R_{in}, \forall i \in \{1, 2, \ldots, m\}, |t_{ia} - t_{ib}| \le (2p-1)\theta$ **then**
        $R_{out} \leftarrow \{r_{out} \in R_{unc} | \exists r_{in} \in R_{in}, r_{out} \asymp r_{in}\} \cap \{r_{out} \in R_{unc} | \forall r \in R_{sm} \setminus R_{in}, r \not\asymp r_{out}\}$;
        if $|R_{out}| \ge p + 1$ **then**
          $R_{sm} \leftarrow R_{sm} \setminus R_{in} \cup R_{out}$;
          local optimal $\leftarrow$ False;
return R_{sm};

to predict temperature. (3) Water [4] is collected by sensors from water quality monitoring stations. The regression model is built on electric conductivity, water temperature and turbidity to predict water level. (4) Air quality [5] is air pollutants data from air-quality monitoring sites in Beijing. It is a high-dimensional dataset with 11 attributes, the regression model is to predict PM10 by other attributes. (5) Fuel is real-world fuel consumption data of vehicles, collected by a vehicle manufacturer. The regression model is built on engine torque and speed to predict fuel consumption.

[4] https://www.kaggle.com/ivivan/real-time-water-quality-data
[5] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data

For House, Telemetry and Air Quality, their timestamps are all naturally aligned to serve as ground truth. We simulate the unaligned variables by introducing disturbance on timestamps [8]. For Fuel, due to the aforementioned issues like transmission delays, only about 5% data are naturally aligned. Analogously, for Water dataset, among 29k tuples, only around 13k tuples are naturally aligned. We thus consider the part of naturally aligned tuples as the ground truth of evaluation, denoted as R_{truth}.

A.4.2 Regression Models. We employ regression models including Linear Regression (LR) and XGBoost (XGB [11]), implemented by scikit-learn[5]. TabNet [6] is a more recent canonical deep tabular data learning architecture for multiple regression.

A.4.3 Alignment Methods. We compare our proposed **S**imilarity **A**lignment under **M**odel **C**onstraint (SAMC) with existing methods: (1) dynamic time warping (DTW) [4], a dynamic programming-based algorithm; (2) derivative dynamic time warping (DDTW) [21], an extension of DTW considering the local derivatives of the data; (3) canonical time warping (CTW) [34], an extension of canonical correlation analysis (CCA) for spatio-temporal alignment of human motion; (4) generalized time warping (GTW) [35, 36], which extends DTW for temporally aligning multi-modal sequences from multiple subjects; (5) trainable time warping (TTW) [22] utilizes a sinc convolutional kernel and a gradient-based optimization technique for multiple alignment. Since our scenario focuses on aligning multiple variables, for DTW, DDTW and CTW proposed only on pairs of time series, we extend them with the framework of Procrustes analysis [35] for handling multiple time series alignment.

A.4.4 Evaluation Metrics. We propose to evaluate how accurate the methods align the tuples (alignment accuracy) and how accurate the models are learned from the aligned data (model accuracy).

For alignment accuracy, we compare the instance R, aligning $T_1, T_2, \ldots, T_m$ by various algorithms, to the truth R_{truth} introduced in Section A.4.1. The alignment accuracy is given by

$$\text{F1-score} = 2 * \frac{precision * recall}{precision + recall},$$

where $recall = \frac{R_{truth} \cap R}{R_{truth}}$, and $precision = \frac{R_{truth} \cap R}{R}$.

For model accuracy, we reserve 20% naturally aligned tuples of each dataset as test data R_{test} for evaluating the learned models. For a learned regression model M, given independent variables $V_1, V_2, \ldots, V_{m-1}$ and dependent variable V_m, we employ RMSE, i.e.,

$$RMSE(M, R_{test}) = \sqrt{\frac{\sum_{r \in R_{test}} (M(r[V_1], \ldots, r[V_{m-1}]) - r[V_m])^2}{|R_{test}|}}.$$

The lower the RMSE is, the better the alignment and learned models are.

A.4.5 Implementation Details. In our **S**imilarity **A**lignment under **M**odel **C**onstraint (SAMC), the model M is initialized by **S**imilarity **A**lignment (SA), as illustrated in Figure 2. The learned model M is then utilized and updated in SAMC as described in Section 2.1.

The experiments are conducted on an Ubuntu 16.04 LTS machine with 16 2.1GHZ cores and 128 GB memory. The large-scale data alignment is executed on a cluster of 3 nodes with Apache Spark 2.2.3. Each node has 32 GB memory and 16 2.6GHz cores. Experimental code and data as well as proofs are available in [3].