This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

Parallel Fractional Hot-Deck Imputation and Variance Estimation for Big Incomplete Data Curing

Yicheng Yang, Jae Kwang Kim, and In Ho Cho*

Abstract—The fractional hot-deck imputation (FHDI) is a general-purpose, assumption-free imputation method for handling multivariate missing data by filling each missing item with multiple observed values without resorting to artificially created values. The corresponding R package FHDI [1] holds generality and efficiency, but it is not adequate for tackling big incomplete data due to the requirement of excessive memory and long running time. As a first step to tackle big incomplete data by leveraging the FHDI, we developed a new version of a parallel fractional hot-deck imputation (named as P-FHDI) program suitable for curing large incomplete datasets. Results show a favorable speedup when the P-FHDI is applied to big datasets with up to millions of instances or 10,000 of variables. This paper explains the detailed parallel algorithms of the P-FHDI for large instances (big-n) or high-dimensionality (big-p) datasets and confirms the favorable scalability. The proposed program inherits all the advantages of the serial FHDI and enables a parallel variance estimation, which will benefit a broad audience in science and engineering.

Index Terms—Parallel fractional hot-deck imputation, incomplete big data, multivariate missing data curing, parallel Jackknife variance estimation.

1 Introduction

Incomplete data problem has been pandemic in nearly all scientific and engineering domains. Inadequate handling of missing data may lead to biased or incorrect statistical inference and subsequent machine learning [2]. In the "imputation" methods, the active research areas of missing data-curing, two major questions arose and have been answered for the past decades. These questions involve "accuracy" and "computational efficiency": how to handle missing values by minimizing the loss of accuracy? Is there any software for handling missing values?

There is a variety of approaches regarding the first question. A simple approach is available in the literature such as removal of instances with missing values [3] and pairwise deletion [4]. Yet, the report by the American Psychological Association strongly discourages the use of removal of missing values, which seriously biases sample statistics [5]. A relatively better simple strategy is to replace the missing values by the conditional expected values obtained from a probabilistic model of incomplete data, which is subsequently fed into particular learning models [6]. Recently, theoretical approaches such as a model based method [7] or the use of an imputation theory have received great attention. Imputation theory is essential to replace a missing value with statistically plausible values. In terms of the number of plausible values for each missing item, there are two distinct branches: single imputation and repeated imputation. Amongst many methods of the "repeated imputation" paradigm, the multiple imputation and fractional imputation have been widely used and investigated in the literature. The multiple imputation (MI) proposed by Rubin [8] retains the advantages of single imputation but overcomes its downside by replacing each missing item with several values representing the distributions of the possible values. MI can handle multivariate missing data using chained equations (MICE), and the imputed values are generated from a set of conditional densities, one for each variable with missing values [9]. Many existing packages support MI and MICE on different platforms, e.g., SOLAS and SAS [10]. However, an inappropriate choice of model for MI may be harmful to its performance. Furthermore, the so-called "congeniality" and "self-efficiency" conditions required for the validity of MI can be quite restrictive in practice [11], [12].

Fractional hot-deck imputation (FHDI), proposed by [13] and extensively discussed by [14] and [15], is a relatively new method to handle item non-response in survey sampling, which creates several imputed values assigned with fractional weights for each missing instance [16]. A serial version R package FHDI was developed by some of the authors of this study [1] to perform the fractional hot-deck imputation and also the fully-efficient fractional imputation (FEFI) [17]. The FHDI can cure multivariate, general incomplete datasets with irregular missing patterns without resorting to distributional assumptions or expert statistical models. It also offers variance estimation using the Jackknife replication method. Besides, other popular imputation methods include multiple imputation with predictive mean matching (PMM) by [18], sequential

- *Y. Yang and I. Cho are with the Department of Civil Engineering, Iowa State University, Ames, IA, 50011.*
  {yicheng, icho}@iastate.edu.
  **: corresponding author, M. IEEE*
- *J.K. Kim is with Department of Statistics, Iowa State University, Ames, IA, 50011.*
  E-mail: jkim@iastate.edu

Manuscript received October 30, 2019; revised October 30, 2019.

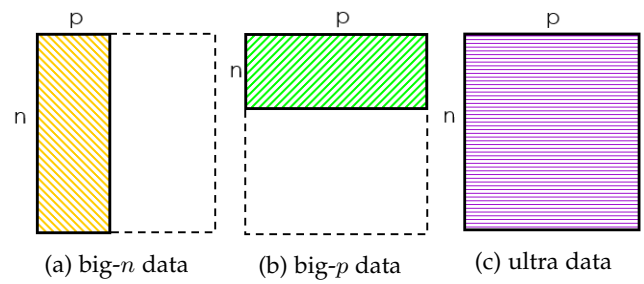
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

regression multivariate imputation by [19], Fuzzy C-Means (FCM) imputation by [20], and K-Harmonic mean imputation by [21].

As we enter into the new era of big data [22], harnessing the potential benefits of analyzing a large amount of data will positively influence science, engineering, and humanity. However, curing the big incomplete data remains a formidable challenge. To the best of our knowledge, a powerful, general-purpose imputation software for big incomplete data is beyond the reach of global researchers. Recently, in some specific areas, parallel computing techniques are gradually adopted for imputation. [23] parallelized single-chain rules of the sampler as a first attempt by using the parallel Markov chain Monte Carlo for Bayesian imputation on missing data. The computational methodology of disk-based shared memory leads to decent scalability. However, a negative aspect of this approach is the need for highly customized, setting-specific, and parallelized software. [24] developed the so-called PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition (PREDICTD) to impute missing experiments for characterizing the epigenome in diverse cell types. Because of the immensely large genome dimension, it employs a parallel algorithm to distribute the tensor across multiple cluster nodes. However, PREDICTED is restricted to imputation for bio-informatics, and they did not present strong parallel efficiency.

Parallelized imputation methods, particularly for untyped genotypes in genetic studies, are investigated by [25]. They applied parallelism to break the entire region into blocks and separately imputed the sub-blocks if the chromosomal regions larger than a size criterion. They showed that the efficacy of the parallel imputation is significantly better than the whole-region imputation. [26] introduced genotype imputation using the parallel processing tool ChunkChromosome, which automatically splits each chromosome into overlapping chunks, allowing the imputation of chromosomes to be run in multiple lower memory. However, this simple parallelism decreases imputation accuracy at the chunk borders. [27] used a GPU to accelerate parallel genotype imputation. The GPU implementation can reach a ten times speedup for a small sample and gradually increases with size of the dataset. The method had a limitation of maximum site number to impute due to an excessive memory issue. All of these recent parallel imputations appear to be restricted to applications in particular bio-informatics domains. The Microsoft R Server 2016 had parallelized many predictive models as well as statistical algorithms. Still, the latest-release server does not yet implement any paralleled imputation algorithm. The missing values are simply omitted during computation.

Despite its generality and efficiency, the serial version of the FHDI may have poor performance for curing "big" incomplete data due to the required excessive memory and prohibitively long execution time. To transform the FHDI into the big data-oriented imputation software, we developed a first version of the parallel fractional hot-deck imputation (named as P-FHDI) program which inherits all the advantages of the serial FHDI and overcomes its computational limitations by leveraging algorithm-oriented parallel computing techniques. Since the algorithmic foundation is the same as that of the serial FHDI, the first version of FHDI is focusing on large data with big instances (the so-called "big-n" data) and that with many variables (the so-called "big-p" data) in Fig. 1. Our program also implements a parallel fully-efficient fractional imputation (denoted as P-FEFI), but it is not recommended in practical big-data application since P-FEFI essentially uses all possible donors for each missing value, which may be prohibitively expensive for big data.



(a) big-n data (b) big-p data (c) ultra data

Fig. 1: Different types of datasets: (a) $n \gg K^p$; (b) $n \leqslant K^p$; (c) n and p are both very large, where K is the number of categories of imputation cells

The significance of this study in the context of data science and machine learning is noteworthy. Reliable imputation may play an important role in potential data-driven research. The impacts of the FHDI on the subsequent machine learning have been investigated in [2]. Following the same methodology, we compare the impact of different data curing methods on machine learning and statistical learning in Fig. 2. The input data is the Energy Efficiency from the UCI Machine Learning Repository [28], which has 768 observations and 9 variables with 30% response rate. Note that all parameters in machine learning models are default settings. Fig. 2 shows that the FHDI, PMM, and FCM imputations have a noticeable positive influence on improving the subsequent machine learning and statistical learning compared to the simple naive method, which fills in missing data using mean values of attributes. The FHDI slightly outperforms the PMM and the FCM imputations. Although this relative performance of the FHDI depends on the adopted incomplete data, this result along with similar prior investigation [2] underpins the positive impact of the P-FHDI on big-data oriented machine learning and statistical learning.

The outline of the paper is structured as follows: we briefly demonstrate backbone theories of the serial FHDI that are segmented by (i) cell construction, (ii) estimation of cell probability, (iii) imputation, and (iv) variance estimation. After an instructive introduction to the adopted parallel computing techniques, we explain key parallel algorithms of the P-FHDI. We validate and evaluate the performance of the P-FHDI with synthetic datasets, and propose a cost model. Finally, we introduce an updated approach embedded in the P-FHDI, particularly for imputing big-p datasets. Comprehensive examples in the Appendix illustrate how to use the program easily with simple and practical data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

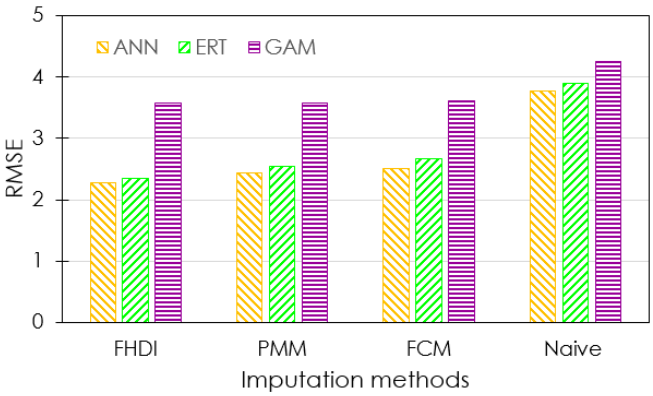


Fig. 2: The impact of data curing methods on the subsequent machine learning methods [artificial neural networks (ANN) and extremely randomly tree (ERT)] and statistical learning [generalized additive model (GAM)].

2 Key Algorithms of the Serial FHDI

For a comprehensive description of the FHDI and P-FHDI, we provide a universal basic setup throughout. Suppose a finite population $\mathbf{U}$ with p-dimensional continuous variables $\mathbf{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p\}$, and y_{il} represents the ith instance of the lth variable where $i \in \{1, 2, \ldots, N\}$ and $l \in \{1, 2, \ldots, p\}$. Then categorize the continuous variables $\mathbf{y}$ to discrete variables $\mathbf{z}$, so-called "imputation cells," where $\mathbf{z}$ takes values within categories $\{1, 2, \cdots, K\}$ for each variable. Express $\boldsymbol{y}_{i,obs}$ and $\boldsymbol{y}_{i,mis}$ to denote the observed and missing part of the ith row of $\mathbf{y}$, respectively. Also, we can write $\boldsymbol{z}_{i,obs}$ and $\boldsymbol{z}_{i,mis}$ as the observed and missing part of the ith row of $\mathbf{z}$.

Let $\boldsymbol{A}$ be the index set with size of n selected from the finite population $\mathbf{U}$. Let $\boldsymbol{A}_M$ denote a set of sample indices with missing values such that $\boldsymbol{A}_M = \{i \in \boldsymbol{A}; \prod_{l=1}^{p} \delta_{il} = 0\}$; alternatively index set with fully observed values is $\boldsymbol{A}_R = \{i \in \boldsymbol{A}; \prod_{l=1}^{p} \delta_{il} = 1\}$. A response indicator function δ_{il} takes value of 1 if y_{il} observed, otherwise $\delta_{il} = 0$. Let $\mathbf{z}$ be discrete values of the continuous variables $\mathbf{y} \in \mathbb{R}^{n \times p}$ to form imputation cells. It consists of missing patterns $\mathbf{z}_M = \{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_M\}$ and observed patterns $\mathbf{z}_R = \{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_R\}$. Let the index set of unique missing pattern be $\tilde{\boldsymbol{A}}_M = \{i, j \in \boldsymbol{A}_M \mid \forall i \neq j,\ \boldsymbol{z}_i \neq \boldsymbol{z}_j\}$ of size $\tilde{n}_M$, alternatively $\tilde{\boldsymbol{A}}_R = \{i, j \in \boldsymbol{A}_R \mid \forall i \neq j,\ \boldsymbol{z}_i \neq \boldsymbol{z}_j\}$ with size of $\tilde{n}_R$ represents the index set of unique observed patterns such that $\tilde{n} = \tilde{n}_M + \tilde{n}_R$ is the size of total unique patterns. Sequentially, $\tilde{\mathbf{z}}_M \in \mathbb{N}^{\tilde{n}_M \times p}$ denotes the unique missing pattern where $\tilde{\mathbf{z}}_M = \{\boldsymbol{z}_i \mid i \in \tilde{\boldsymbol{A}}_M\}$ and $\tilde{\mathbf{z}}_R \in \mathbb{N}^{\tilde{n}_R \times p}$ denotes unique observed pattern where $\tilde{\mathbf{z}}_R = \{\boldsymbol{z}_i \mid i \in \tilde{\boldsymbol{A}}_R\}$. Note that if y_{il} is missing, an imputed value y_{il}^* will be selected from the same j^{th} donor to fill in.

Suppose variable $\boldsymbol{y}_l$ partitioned into H_l groups such that $\boldsymbol{z}_l$ takes values in $\{1, \ldots, H_l\}$. The cross classification of all variables form imputations cells $\mathbf{z}$ and we assume a cell mean model such that:

$$\mathbf{y} \mid (\boldsymbol{z_1} = H_1, \ldots, \boldsymbol{z_p} = H_p) \sim (\boldsymbol{\mu}_{H_1,\ldots,H_p}, \boldsymbol{\Sigma}_{H_1,\ldots,H_p}) \quad (1)$$

where $\boldsymbol{\mu}_{H_1,\ldots,H_p}$ is a vector of cell means and $\boldsymbol{\Sigma}_{H_1,\ldots,H_p}$ is the variance-covariance matrix of $\mathbf{y}$ in cell $(H_1 \ldots H_p)$. Then utilizing a finite mixture model under missing at random (MAR) condition, the conditional distribution of $f(\boldsymbol{y}_{i,mis} \mid \boldsymbol{y}_{i,obs})$ is approximated by

$$f(\boldsymbol{y}_{i,mis} \mid \boldsymbol{y}_{i,obs}) \cong \textstyle\sum_{g=1}^{H} p(\boldsymbol{z}_i = g \mid \boldsymbol{y}_{i,obs}) f(\boldsymbol{y}_{i,mis} \mid \boldsymbol{y}_{i,obs}, \boldsymbol{z}_i = g) \quad (2)$$

where $p(\boldsymbol{z}_i = g \mid \boldsymbol{y}_{i,obs})$ is the conditional cell probability and $f(\boldsymbol{y}_{i,mis} \mid \boldsymbol{y}_{i,obs}, \boldsymbol{z}_i = g)$ is the within-cell conditional density of $\boldsymbol{y}_{i,mis}$ given $\boldsymbol{y}_{i,obs}$ derived from (1). The FHDI consists of four subsections as (i) Cell construction (ii) Estimation of cell probability (iii) Imputation, and (iv) Variance estimation. An illustration of each subsection is presented as follows.

2.1 Cell construction

The pre-determination of the imputation cells had not been discussed by Kim [15]. Im et al. [29] proposed an approach to generate imputation cells $\mathbf{z}$ using the estimated sample quantiles. Suppose we wish to categorize y_l with G categories. Considering the estimated distribution function of y_l:

$$\hat{F}_l(t) = \frac{\sum_{i \in A} \delta_{il} w_i I(y_{il} \leq t)}{\sum_{i \in A} \delta_{il} w_i} \quad (3)$$

where I is an indicator function taking value of one if true and zero if false and w_i represents the sampling weight of unit i. Given a set of proportions $\{a_1, \cdots, a_G\}$ satisfying $0 < a_1 < \cdots < a_G$, the estimated sampling quantile of y_l for a_g is defined as:

$$\hat{q}_{a_g} = min\{t, \hat{F}(t) > a_g\}. \quad (4)$$

Hence y_{il} will be categorized as an imputation cell g if $\hat{q}_{a_g - 1} < y_{il} < \hat{q}_{a_g}$.

We propose a new mathematical notation for the iteration of operations to facilitate the description of the FHDI and P-FHDI. A new mathematical symbol '$\sum\!\!\!\!\!\!\bigcirc$' denotes a loop which repeats a sequence of the same operation $S(x)$ with discrete input augments within a fixed range. The following is the simplest proposal of the loop symbol of an operation S

$$\sum\!\!\!\!\!\!\bigcirc_{i=a}^{b} S(x_i) \equiv \{S(x_a), S(x_{a+1}), \ldots, S(x_{b-1}), S(x_b)\} \quad (5)$$

where $i = a, \ldots, b$. We have an extension of

$$\sum\!\!\!\!\!\!\bigcirc_{i=a}^{b} \sum\!\!\!\!\!\!\bigcirc_{j=c}^{d} S(x_{ij}) = \begin{bmatrix} S(x_{ac}) & S(x_{a(c+1)}) & \ldots & S(x_{ad}) \\ S(x_{(a+1)c}) & S(x_{(a+1)(c+1)}) & \ldots & S(x_{(a+1)d}) \\ \vdots & \vdots & \vdots & \vdots \\ S(x_{bc}) & S(x_{b(c+1)}) & \ldots & S(x_{bd}) \end{bmatrix} \quad (6)$$

where integer indices $i = a, \ldots, b$ and $j = c, \ldots, d$.

However, the initial discretization may not give at least two donors for each recipient to capture the variability from imputation. Identification of the unique observed patterns $\tilde{\mathbf{z}}_R$ and unique missing patterns $\tilde{\mathbf{z}}_M$ is crucial for selection of donors for each recipient. First of all we need to sort missing patterns $\mathbf{z}_M \in \mathbb{N}^{n_M \times p}$ such that $\mathbf{z}_M = \{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_M; \|\boldsymbol{z}_i\| \leqslant \|\boldsymbol{z}_{i+1}\|\}$. Note that $\|\boldsymbol{z}_i\|$ takes the string format of $\boldsymbol{z}_i$ which is comparable by its numerical value. Thereby, the unique missing patterns $\tilde{\mathbf{z}}_M$ will be obtained by

$$\tilde{\mathbf{z}}_M = \sum\!\!\!\!\!\!\bigcirc_{i=1}^{n_M} \boldsymbol{z}_{Mi} I(\|\boldsymbol{z}_{Mi}\| < \|\boldsymbol{z}_{M(i+1)}\|) \quad (7)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

where $\boldsymbol{z}_{Mi} \in \mathbf{z}_M$. Similarly, sort observed patterns $\mathbf{z}_R \in \mathbb{N}^{n_R \times p}$ such that $\mathbf{z}_R = \{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_R; \|\boldsymbol{z}_i\| \leqslant \|\boldsymbol{z}_{i+1}\|\}$. So we have the unique observed patterns $\tilde{\mathbf{z}}_R$ by

$$\tilde{\mathbf{z}}_R = \biguplus_{i=1}^{n_R} \boldsymbol{z}_{Ri} I(\|\boldsymbol{z}_{Ri}\| < \|\boldsymbol{z}_{R(i+1)}\|) \tag{8}$$

where $\boldsymbol{z}_{Ri} \in \mathbf{z}_R$. Suppose $\boldsymbol{D}_i \in \mathbb{N}^{M_i}$ be the set recording indices of donors of the ith recipient $\boldsymbol{z}_i = \{\boldsymbol{z}_{i,obs}, \boldsymbol{z}_{i,mis}\}$ where $\boldsymbol{D}_i = \{j \in \tilde{\boldsymbol{A}}_R \mid \boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}\}$ with size of M_i. M_i represents the number of donors of the ith recipient. By $\tilde{\mathbf{z}}_M$ and $\tilde{\mathbf{z}}_R$, we determine the set of number of total donors of all recipients $\boldsymbol{M} = \{M_i \mid i \in \tilde{\boldsymbol{A}}_M\}$ by using

$$\boldsymbol{M} = \biguplus_{i=1}^{\tilde{n}_M} \left\{ \sum_{j=1}^{\tilde{n}_R} I(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \right\} \tag{9}$$

Note that $\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|$ claims that each entity of string $\|\boldsymbol{z}_{i,obs}\|$ and string $\|\boldsymbol{z}_{j,obs}\|$ should be identical, respectively. Further, the actual index set of donors of all recipients $\mathbf{L} = \{\boldsymbol{D}_i \mid i \in \tilde{\boldsymbol{A}}_M\}\}$ can be written as

$$\mathbf{L} = \biguplus_{i=1}^{\tilde{n}_M} \biguplus_{j=1}^{\tilde{n}_R} jI(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \tag{10}$$

The minimum entity of $\boldsymbol{M} \in \mathbb{N}^{\tilde{n}_M}$ is denoted by m. If $m > 2$, the initial generation of imputation cells have at least two donors as candidates for each recipient. Otherwise, we apply a cell collapsing procedure to adjust the categories of imputation cells to produce more donors and stop only if every recipient has at least two donors. For instance, a dummy $\mathbf{z}$ has a recipient $(NA, 1)$ and all donors are listed in the left panel of Table 1 after initial data categorization. However, the recipient has only one possible donor $(3, 1)$. Hence cell collapsing occurs by merging the original categories 1 and 2 of p_2 as category 1 to complement deficient donors. Now, $(3, 1)$ and $(2, 1)$ both serve as donors for the recipient $(NA, 1)$.

TABLE 1: An illustrative example for cell collapsing with initial categories of 3 for p_1 and p_2. Left: initial imputation cells. Right: final imputation cells after cell collapsing.

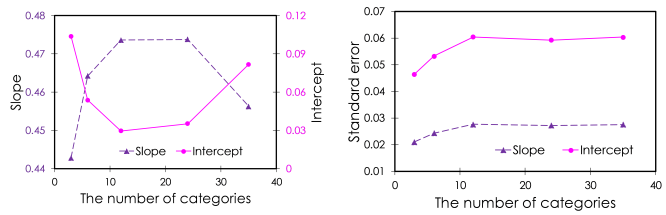
p_1	p_2
3	1
2	2
1	3

p_1	p_2
3	1
2	1
1	2

The number of categories of imputation cells will affect the number of unique observed patterns $\tilde{n}_R$, the number of unique missing patterns $\tilde{n}_M$, and the number of iterations in the cell construction. The input dataset in this simulation has 500 observations and 4 variables with 20% missingness. Table 2 shows that a growing number of categories increases the number of unique patterns until some threshold. Meantime, it requires a larger number of iterations to guarantee at least two donors for all unique missing patterns. In addition, the number of categories will affect subsequent regression analyses. Regression coefficient estimates of y_1 given y_2 in Fig. 3 (a) shows that 12 categories optimize the effect of the FHDI on regression analyses. Note that the true values of the slope and intercept are 0.5 and 0, respectively. Fig. 3 (b) shows that a growing number of categories will increase the standard errors of the intercept and slope. A similar discussion about the effect of number of categories on regression in [30] has a good agreement with our results.

TABLE 2: The number of unique observed patterns ($\tilde{n}_R$), number of unique missing patterns ($\tilde{n}_M$), and the number of iterations of the cell construction with a growing number of categories.

Categories	$\tilde{n}_R$	$\tilde{n}_M$	Iterations
3	54	130	9
6	103	215	112
12	97	243	227
24	90	243	275
35	88	240	286



(a) (b)

Fig. 3: Effect of a growing number of categories on (a) regression coefficient estimates and (b) their standard errors of y_1 given y_2.

2.2 Estimation of cell probability

To estimate conditional cell probability using a modified EM algorithm by weighting [31], we partition $\mathbf{z}$ into G groups on basis of A_M, denoted by $\mathbf{z}_1, \ldots, \mathbf{z}_G$. Let the size of all possible donors of $\boldsymbol{z}_{g,mis}$ be H_g. Given a possible imputed value $\boldsymbol{z}_{g,mis}^{*(h)}$ for $\boldsymbol{z}_{g,mis}$, the estimated conditional cell probability is defined in E step:

$$\hat{\pi}_h^{(t)} = \frac{\hat{p}^{(t)}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}_{g,mis}^{*(h)})}{\sum_{h=1}^{H_g} \hat{p}^{(t)}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}_{g,mis}^{*(h)})} \tag{11}$$

where $\hat{p}^{(t)}(\boldsymbol{z})$ is the estimated joint cell probability computed from the full respondents at iteration t. The joint cell probability is updated in M step using weighting:

$$\hat{p}^{(t+1)}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{g,mis}^{*(h)}) = \left(\sum_{i=1}^{n} w_i \right)^{-1} \sum_{i=1}^{n} w_i \hat{\pi}_h^{(t)} \times I(\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{g,obs}) \tag{12}$$

where n is the sample size and w_i is the sampling weight of unit i. An indicator function I takes a value of one if $\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{g,obs}$ and zero otherwise. The EM algorithm will terminate in case of convergence of $\hat{p}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{g,mis}^{*(h)})$ or reaching maximum iteration max defined by users. Note that $\sum_{h=1}^{H_g} \hat{\pi}_h = 1$.

2.3 Imputation

It is necessary to compute fractional weights in advance. Let w_{ij}^* be the fractional weights of the jth donor for the ith

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

recipient given by:

$$w_{ij}^{*} = \hat{\pi}^{(t)}_{\boldsymbol{z}^{*}_{i,mis}|\boldsymbol{z}_{i,obs}} \frac{w_j I\{\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}^{*}_{j,mis}\}}{\sum_{j=1}^{M_i} w_j I\{\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}^{*}_{j,mis}\}} \tag{13}$$

where $\boldsymbol{z}^{*}_{j,mis}$ is the jth imputed value for the ith recipient. In addition, we can verify computation of w^{*}_{ij} by $\sum_{j=1}^{M_i} w^{*}_{ij} = 1$, where M_i denotes number of donors for the ith recipient. In FEFI, all respondents are employed as donors for each recipient and assigned the fractional weights. Once donors and corresponding fractional weights determined, FEFI estimator of Y_l can be written as:

$$\hat{Y}_{l,FEFI} = \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j \in A} w^{*}_{ij,FEFI} y_{jl}\} \tag{14}$$

and the fractional weights of the jth donor for the ith recipient is defined:

$$w^{*}_{ij,FEFI} = \sum_{g=1}^{G} a_{ig} \sum_{h=1}^{H} \hat{\pi}_{h|g} \frac{w_j \delta_j a_{jgh}}{\sum_{l \in A} w_l \delta_l a_{lgh}} \tag{15}$$

where $a_{jgh} = 1$ if $(\boldsymbol{z}_{j,obs}, \boldsymbol{z}_{j,mis}) = (\boldsymbol{z}_{g,obs}, \boldsymbol{z}^{*(h)}_{g,mis})$, otherwise 0. However, it requires too much computation with large input data.

Instead of using all the respondents, the FHDI selects M donors among all FEFI donors using a systematic probability proportional to size (PPS) method to compensate for the recipient. Firstly, it sorts all FEFI donors in y values by the half-ascending and half-descending order. Considering $L_1 = 0$, one can construct the interval of $(L_1, L_{n_{Rg}})$ by

$$L_{j+1} = L_j + M \times w^{*}_{ij,FEFI} \tag{16}$$

for $\forall\ j \in [1 : n_{Rg} - 1]$. Sequentially, successive intervals will be computed by Eq. (16). Let RN_g be a random number from a uniform distribution $U(0, 1)$. Then for $i \in A_{Mg}$, M donors will be selected if:

$$L[j] \leq \frac{RN_g + i - 1}{n_{Mg}} + l - 1 \leq L[j + 1] \tag{17}$$

for $l = 1, \ldots, M$. In case of $n_{Rg} \leqslant M$, we take all FEFI donors for the recipient. The FHDI estimator of y_l with M selected donors is

$$\hat{Y}_{l,FHDI} = \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w^{*}_{ij} y^{*(j)}_{il}\} \tag{18}$$

where $w^{*}_{ij} = M^{-1}$ and $y^{*(j)}_{il}$ is the jth imputed value for y_{il}.

Afterward, we can prepare imputation results $\hat{\mathbf{Y}} = \{y^{*(j)}_{il} \mid i \in \{1, \ldots, n\}; l \in \{1, \ldots, p\}; j \in \{1, \ldots, M_i\}\}$ along with fractional weights $w^{*}_{ij,FHDI}$ as outputs. Let $\acute{\boldsymbol{A}} \in \mathbb{N}^{n_A}$ be index set of $\hat{\mathbf{Y}}$ where $n_A = n_R + \sum_{i=1}^{n_M} M_i$. And $\acute{\boldsymbol{\mathcal{A}}} \in \mathbb{N}^{n_A}$ denotes the index set of sorted $\acute{\boldsymbol{A}}$ in the ascending order. We can record the index mapping $\boldsymbol{\psi} \in \mathbb{N}^{n_A}$ from $\acute{\boldsymbol{A}}$ to $\acute{\boldsymbol{\mathcal{A}}}$ by

$$\boldsymbol{\psi} = \sum_{i=1}^{n_A} \sum_{j=1}^{n_A} j I(\acute{\mathcal{A}}_i = \acute{A}_j) \tag{19}$$

Eventually, imputation results $\hat{\mathbf{Y}}$ can be reorganized with regard to the index mapping $\boldsymbol{\psi}$ easily as outputs.

2.4 Variance estimation

The Jackknife method is implemented for variance estimation of the FHDI estimator. The Jackknife variance estimator of $\hat{Y}_{l,FHDI}$ is determined by the following steps:

S1: Identification of the joint probability $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$ of unique missing patterns $\tilde{\mathbf{z}}_M$ by

$$\begin{aligned} \hat{\mathbf{p}}(\tilde{\mathbf{z}}_M) &= \sum_{j \in \tilde{A}_M} \hat{\boldsymbol{p}}(\boldsymbol{z}_{j,obs}, \boldsymbol{z}^{*}_{j,mis}) \\ &= \sum_{j \in \tilde{A}_M} \left(\sum_{i=1}^{n} w_i \right)^{-1} \sum_{i \in A}^{n} w_i \hat{\boldsymbol{\pi}}_j I(\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}) \end{aligned} \tag{20}$$

where $\hat{\boldsymbol{\pi}}_j$ represents the conditional probability of the jth recipient's donors.

S2: Delete unit $k \in A$. If $k \in A_M$, then w^{*}_{ij} will not be updated. If the size of $\hat{\boldsymbol{p}}(\boldsymbol{z}_{k,obs}, \boldsymbol{z}^{*}_{k,mis})$ is 0, skip to the next iteration.

S3: If $k \notin A_M$, then w^{*}_{ij} for the replicate k will be updated by

$$w^{*(k)}_{ij} = \begin{cases} w^{*}_{ij} - w^{*}_{ij,FEFI} & \text{if } j = r \\ w^{*}_{ij} + (w^{*}_{ir,FEFI}) \frac{w^{*}_{ij,FEFI}}{\sum_{j \neq r} w^{*}_{ij,FEFI}} & \text{if } j \neq r \\ w^{*}_{ij} & \text{if } k \in A_M \end{cases} \tag{21}$$

where $i \in A_R$ and r is the index of the closest donor to k. Eq. (21) updates the fractional weights and will be used in parallel algorithm later, thus is denoted as FW in ease of use. However, measurement of closeness in terms of the scale of the data is challenging. We employ Mahalanobis Distance (denoted as MD) to measure the distance from a replicate to the other M donors by

$$\boldsymbol{MD} = \sqrt{(\boldsymbol{y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})} \tag{22}$$

where $\boldsymbol{y} = \{y_{i1}, \ldots, y_{ip}\}^T$ is the vector of the ith observation of $\mathbf{y}$, $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_p\}^T$ is the vector of mean values of p variables, and $\boldsymbol{\Sigma}$ is a covariance matrix. Hence, r is the index of smallest value among $\boldsymbol{MD}$.

S4: Considering the updated fractional weight $w^{*(k)}_{ij}$, we can compute the variance $\hat{V}(\hat{Y}_{l,FHDI})$ by

$$\hat{V}(\hat{Y}_{l,FHDI}) = \sum_{k=1}^{n} c_k (\hat{Y}^{k}_{l,FHDI} - \hat{Y}_{l,FHDI})^2 \tag{23}$$

where c_k denotes replicate factor associated with $\hat{Y}^{k}_{l,FHDI}$ or $\hat{Y}^{k}_{l,FEFI}$, which is the kth replicate estimator of y_l defined by

$$\hat{Y}^{k}_{l,FHDI} = \sum_{i \in A} w^{k}_{i} \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w^{*(k)}_{ij} y^{*(j)}_{il}\} \tag{24}$$

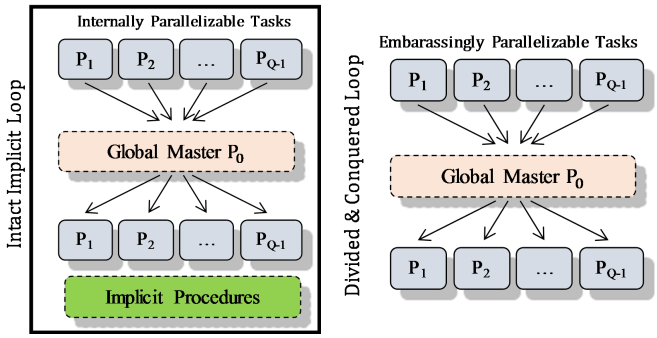
where w^{k}_{i} is the kth replicate of the fractional weight w^{*}_{ij}. The quantity $y^{*(j)}_{il}$ is the jth imputed value of y_{il}. The FHDI estimator is defined in Eq. (18). Similarly, we can determine $\hat{V}(\hat{Y}_{l,FEFI})$ using the same equations.

We evaluate the variance estimation via relative bias of standard error and confidence interval. In the simulations, input data has 500 observations and 4 variables with 30% missingness and Monte Carlo simulation size is 600. The average relative bias less than 5% often indicates a good variance estimation in Table 3. The average coverage rate of 95% confidence interval is close to 95%. Overall, the variance estimation of the FHDI performs in a favorable manner.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

TABLE 3: Monte Carlo results of relative bias of standard error, the average length of 95% confidence interval, and its coverage rate based on 600 simulation runs.

Variable	Rel.Bias (%)	Ave.Length of CI	Coverage rate
Variable 1	-3.62	0.1025	0.95
Variable 2	-2.83	0.1040	0.93
Variable 3	-7.88	0.1362	0.93
Variable 4	-4.28	0.1356	0.95



(a) (b)

Fig. 4: Two parallel schemes: (a) Internal parallelization within the unbreakable implicit loop; (b) Typical divide and conquer for embarrassingly parallelizable explicit loop.

3 Parallel Algorithms for the FHDI

Fig. 4 shows the two parallel schemes adopted for this study. The two distinct schemes are needed since the primary global loops for the majority of tasks are "implicit", and thus a direct divide and conquer scheme is not applicable such that parallelization is focusing on the separately parallelizable internal tasks without breaking the implicit loop. In contrast, some embarrassingly parallelizable tasks such as Jackknife variance estimation are tackled by the typical divide and conquer scheme. Suppose we have in total Q processors indexed by $\{0, 1, 2, \ldots, Q-1\}$. The first processor indexed by 0 (called master processor) will collect work from all other processors (called slave processors) via communication by the basic operations MPI_Send and MPI_Recv (denoted as MPI_SR in conjunction). The pieces of distributed work will be assembled in the master processor by MPI_Gather (denoted as Ω). The Ω_i^j $(i \neq 0)$ represents that the master processor gathers pieces of work distributed to all slave processors ranking i to j. Generally, if the entire domain Υ of a problem is divided into disjoint domain Υ_q and no memory overlapping is required, thus all work is done concurrently by

$$\Upsilon = \Omega_{q=1}^{Q-1} \left(\bigoplus_{x_i \in \Upsilon_q} S(x_i) \right) \tag{25}$$

Another challenge of parallel computing is how a problem can be partitioned efficiently to be tackled concurrently, and how we can manage balanced computing. To this end, we employ two work distribution schemes in the P-FHDI, i.e., uniform distribution (denoted as UniformDistr) and cyclic distribution (denoted as CyclicDistr). They compute the beginning (denoted as s) and ending index (denoted as e) of distributed work on processor q. Fig. 5 illustrates the contrast between two work distribution schemes visually. These two schemes are summarized in Algorithms 1 and 2.

Algorithm 1 Uniform job distribution

Input: number of total instances n, number of available processors Q, index of current processor q,
Output: boundary indices s and e

1: $N_1 = \text{floor}(\frac{n}{Q-1})$
2: $N_2 = n - N_1(Q-2)$
3: $s = (q-1)N_1$
4: $e = (q-1)N_1 + N_2$

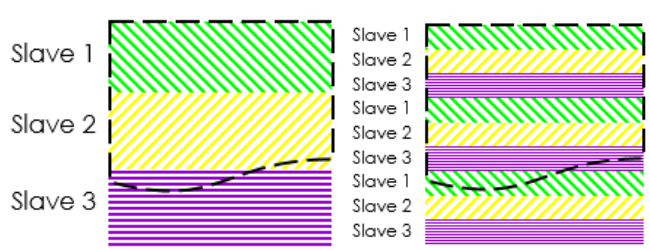
The uniform distribution scheme in Algorithm 1 averages work in N_1 pieces over slave processors and squeezes the residuals to the last available processor as N_2. Then the boundary indices s and e of distributed work in the current processor q can be computed in lines 3 and 4, respectively. Alternatively, the core of the cyclic distribution scheme in Algorithm 2 is to assign the work to $Q-1$ slave processors recursively.

Algorithm 2 Cyclic job distribution

Input: number of total instances n, number of available processors Q, index of current processor q, and split processor q_s
Output: boundary indices s and e

1: $N_3 = \text{floor}(\frac{n}{Q-1})$
2: $N_4 = \frac{(n-N_3)q_s}{(Q-q_s-1)}$
3: **if** $q \leq q_s$ **then**
4: $\quad s = (q-1)N_3$
5: $\quad e = qN_3$
6: **end if**
7: **if** $q > q_s$ **then**
8: $\quad s = qN_3 + (q-q_s-1)N_4$
9: $\quad e = qN_3 + (q-1)N_4$
10: **end if**

Although it is easy to implement the uniform distribution, this uniform decomposition will lead to considerable work imbalance when the problem domain Υ is severely irregular [32]. See Fig. 5 (a) for an illustration; an irregular work domain Υ causes heavy workload in slave processors 1 and 2 while slave processor 3 is almost idle. As a successful remedy to this problem, cyclic distribution can effectively balance the work domain among slave processors. Fig. 5 (b) shows such a balanced situation with all three slave processors being busy. Depending upon the parallel tasks, we adopt the best choice of parallel job distributions.



(a) Uniform distribution (b) Cyclic distribution

Fig. 5: Different methods distributing work to the slave processors. A work domain Υ (represented by an irregular area enclosed by dashed line) is handled by three slave processors: Slave 1 (downward hatching), Slave 2 (upward hatching) and Slave 3 (horizontal hatching).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

3.1 Parallel cell construction and estimation of cell probability

The determination of initial imputation cells is computationally cheap, i.e., Eqs. (3) and (4). However, it may take considerably large iterations for the cell collapsing process to guarantee at least two donors for each recipient, i.e., Eqs. (7) to (10). The current algorithm of cell collapsing is an implicit process which is non-parallelizable. Considering the inevitable obstacle, we employ the internal parallelization within the unbreakable implicit iterations in Algorithm 3. In line 2 of Algorithm 3, we categorize input data $\mathbf{y}$ to imputation cells $\mathbf{z}$ using the estimated sample quantiles defined in Eq. (4). Before proceeding to line 4 of Algorithm 3, the function ZMAT is explained explicitly in Algorithm 4 to identify the unique patterns $\tilde{\mathbf{z}}_M$ and $\tilde{\mathbf{z}}_R$ for the selection of donors. In lines 1 and 2 of Algorithm 4, we employ the cyclic distribution for job division and adjust the boundary indices accordingly. The function tunePoints has been frequently used in the P-FHDI to adjust the border indices (s and e) of work in each processor slightly to be adjusted indices (s_a and e_a). The current parallel work distribution scheme may lead to the boundary mismatch issue. For example, we have a sample set $\{4, 2, 3, 3, 2, 4, 1\}$ indexed by $\{1, \ldots, 7\}$ and sorted as $\{1, 2, 2, 3, 3, 4, 4\}$. The index mapping from the original sample set to the sorted sample set will be $\boldsymbol{\psi} = \{7, 2, 5, 3, 4, 1, 6\}$. However, the uniform distribution upon four processors in the left panel of Table 4 will result in the incorrect mappings $\boldsymbol{\psi}^1 = \{7, 2\}$, $\boldsymbol{\psi}^2 = \{2, 3\}$, and $\boldsymbol{\psi}^3 = \{3, 1, 6\}$. Whereas, the job distribution after adjustment will guarantee the identical indices assigned to the same processor shown in the right panel of Table 4. Hence the updated mapping will be $\boldsymbol{\psi}^1 = \{7, 2, 5\}$, $\boldsymbol{\psi}^2 = \{3, 4\}$ and $\boldsymbol{\psi}^3 = \{1, 6\}$. We identify

TABLE 4: Left: Uniform job distribution. Right: uniform job distribution after adjustments by function tunePoints with slave processors 1, 2 and 3 (denoted as S1, S2 and S3).

S1		S2		S3		
1	2	2	3	3	4	4

S1		S2		S3	
1	2	2	3	3	4	4

$\tilde{\mathbf{z}}_M^{(q)}$ in each processor q in line 4 of Algorithm 4 by

$$\tilde{\mathbf{z}}_M^{(q)} = \bigoplus_{i=s_a}^{e_a} \boldsymbol{z}_{Mi} I(\|\boldsymbol{z}_{Mi}\| < \|\boldsymbol{z}_{M(i+1)}\|) \tag{26}$$

where $\boldsymbol{z}_{Mi} \in \mathbf{z}_M$. After master-slave communications in line 6, we obtain $\tilde{\mathbf{z}}_M$ by gathering operation in line 7. Similarly, we have $\tilde{\mathbf{z}}_R^{(q)}$ in each processor in lines 10 to 12 by

$$\tilde{\mathbf{z}}_R^{(q)} = \bigoplus_{i=s_a}^{e_a} \boldsymbol{z}_{Ri} I(\|\boldsymbol{z}_{Ri}\| < \|\boldsymbol{z}_{R(i+1)})\| \tag{27}$$

where $\boldsymbol{z}_{Ri} \in \mathbf{z}_R$. we assemble $\tilde{\mathbf{z}}_R$ after communication in lines 13 and 14 and broadcast to all slave processors in line 15.

Before proceeding to line 5 of Algorithm 3, the function nDAU is explicitly demonstrated in Algorithm 5 to get the minimum number of donors m of recipients. We distribute $\tilde{n}_M$ to each processor cyclically in line 1 of Algorithm 5. The set of number of total donors for all recipients is obtained in line 3 by

$$\boldsymbol{M}^{(q)} = \bigoplus_{i=s}^{e} \left\{ \sum_{j=1}^{\tilde{n}_R} I(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \right\} \tag{28}$$

Meantime, the actual indices of donors for all recipients will be stored in line 4 by

$$\mathbf{L}^{(q)} = \bigoplus_{i=s}^{e} \bigoplus_{j=1}^{\tilde{n}_R} j I(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \tag{29}$$

Then communication is processed between lines 6 and 8 to assemble $\boldsymbol{M}$ and $\mathbf{L}$. Note that all processors require results of cell construction to continue cell probability estimation. Thus we broadcast results from the master processor to slave processors in line 9.

After explicit explanation of function nDAU in line 4 of Algorithm 3, it will perform cell collapsing in case of $m < 2$ in lines 5 to 7. The recursive iterations will terminate if $m \geqslant 2$ in lines 8 to 10. Otherwise, the iterations will continue to the max iteration (i.e., $2n$).

Algorithm 3 Parallel cell construction

Input: raw data $\mathbf{y}$
Output: imputation cells $\mathbf{z}$

1: **for** $\forall\, i$ in $0 : max\_iteration$ **do**
2: Categorize raw data $\mathbf{y}$ to imputation cells $\mathbf{z}$
3: Invoke function ZMAT($\mathbf{z}$) $\rightarrow (\tilde{\mathbf{z}}_R, \tilde{\mathbf{z}}_M)$
4: Invoke function nDAU($\tilde{\mathbf{z}}_R, \tilde{\mathbf{z}}_M$) $\rightarrow (\boldsymbol{M}, m, \mathbf{L})$
5: **if** $m < 2$ **then**
6: Perform cell collapsing on $\mathbf{z}$ described in Table 1
7: **end if**
8: **if** $m \geqslant 2 \;\;||\;\; i = max\_iteration$ **then**
9: **break**
10: **end if**
11: **end for**

Algorithm 4 Parallel function ZMAT

Input: imputation cells $\mathbf{z}$
Output: unique observed pattern $\tilde{\mathbf{z}}_R$ and missing pattern $\tilde{\mathbf{z}}_M$

1: CyclicDistr(n_R) $\rightarrow (s, e)$
2: tunePoints(s, e) $\rightarrow (s_a, e_a)$
3: **for** $\forall\, i$ in $s_a : e_a$ **do**
4: Identify $\tilde{\mathbf{z}}_M^{(q)}$
5: **end for**
6: MPI_SR($\tilde{\mathbf{z}}_M^{(q)}$)
7: $\tilde{\mathbf{z}}_M = \Omega_1^{Q-1} \tilde{\mathbf{z}}_M^{(q)}$
8: CyclicDistr(n_M) $\rightarrow (s, e)$
9: tunePoints(s, e) $\rightarrow (s_a, e_a)$
10: **for** $\forall\, i$ in $s_a : e_a$ **do**
11: Identify $\tilde{\mathbf{z}}_R^{(q)}$
12: **end for**
13: MPI_SR($\tilde{\mathbf{z}}_R^{(q)}$)
14: $\tilde{\mathbf{z}}_R = \Omega_1^{Q-1} \tilde{\mathbf{z}}_R^{(q)}$
15: MPI_Bcast($\tilde{\mathbf{z}}_M; \tilde{\mathbf{z}}_R$)

Algorithm 5 Parallel function nDAU

Input: unique observed pattern $\tilde{\mathbf{z}}_R$ and missing pattern $\tilde{\mathbf{z}}_M$
Output: number set of donors $\boldsymbol{M}$

1: CyclicDistr($\tilde{n}_M$) $\rightarrow (s, e)$
2: **for** $\forall\, i$ in $s : e$ **do**
3: $\boldsymbol{M}^{(q)}$.add(M_i)
4: $\mathbf{L}^{(q)}$.add($\boldsymbol{D}_i$)
5: **end for**
6: MPI_SR($\boldsymbol{M}^{(q)}; \mathbf{L}^{(q)}$)
7: $\boldsymbol{M} = \Omega_1^{Q-1} \boldsymbol{M}^{(q)}$
8: $\mathbf{L} = \Omega_1^{Q-1} \mathbf{L}^{(q)}$
9: MPI_Bcast($\boldsymbol{M}; \mathbf{L}$)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

Similarly, estimation of cell probability is an implicit and iterative process, which does not support simple parallelism. The EM iterations start in line 1 of Algorithm 6 and terminate if the joint probability of $\boldsymbol{z}^* = (\boldsymbol{z}_{obs}, \boldsymbol{z}^*_{mis})$ converges in lines 6 to 8. Let $\boldsymbol{A}_\pi = \{1, \ldots, H_G\}$ be local index set of initial conditional probabilities $\hat{\boldsymbol{\pi}}^{(t)}$ with size of n_π. We update $\boldsymbol{w} \in \mathbb{N}^{n_\pi}$ and $\hat{\boldsymbol{\pi}}^{(t)} \in \mathbb{R}^{n_\pi}$ in line 2 where $\hat{\boldsymbol{\pi}}^{(t)}$ is

$$\hat{\boldsymbol{\pi}}^{(t)} = \sum_{h=1}^{H_g} \frac{\hat{p}^{(t)}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}^{*(h)}_{g,mis})}{\sum_{h=1}^{H_g} \hat{p}^{(t)}(\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{i,mis} = \boldsymbol{z}^{*(h)}_{g,mis})} \tag{30}$$

Note that $\hat{\boldsymbol{\pi}}^{(t)}$ requires reorganization regarding the index set $\boldsymbol{\mathcal{A}}_\pi = \{\mathcal{A}_{\pi i} \in \boldsymbol{A}_\pi \mid \forall i, \mathcal{A}_{\pi(i+1)} > \mathcal{A}_{\pi i}\}$ in the ascending order. Before proceeding to line 4 of Algorithm 6, we explain the function Order in Algorithm 7 to generate index mapping $\boldsymbol{\xi}$ of $\boldsymbol{\mathcal{A}}_\pi$ in lines 4 to 10 on each processor by

$$\boldsymbol{\xi}^{(q)} = \sum_{i=s_a}^{e_a} \sum_{j=1}^{n_\pi} jI(\mathcal{A}^{(q)}_{\pi i} = A_{\pi j}) \tag{31}$$

After communication in line 11, $\boldsymbol{\xi}$ will be assembled in line 12. By leveraging $\boldsymbol{\xi}$ in line 4 of Algorithm 6, $\hat{\boldsymbol{\pi}}^{(t)}$ can be rearranged to update $\hat{\boldsymbol{p}}^{(t+1)}(\boldsymbol{z}^*)$ in line 5 by

$$\hat{\boldsymbol{p}}^{(t+1)}(\boldsymbol{z}^*) = \left(\sum_{i=1}^{n} w_i\right)^{-1} \sum_{i=1}^{n} w_i \hat{\boldsymbol{\pi}}^{(t)} I(z_{i,obs} = \boldsymbol{z}_{g,obs}) \tag{32}$$

In line 6, the EM algorithm will terminate if $\hat{\boldsymbol{p}}^{(t+1)}(\boldsymbol{z}^*)$ converges to a threshold ϵ (e.g., 10^{-6}).

Algorithm 6 Parallel estimation of cell probability

Input: Imputation cells $\mathbf{z}$, sampling weight $\boldsymbol{w}$
Output: cell probability $\hat{\boldsymbol{p}}(\boldsymbol{z}^*)$

1: **for** t in $0 : max$ **do**
2: Update $\boldsymbol{w}$ and $\hat{\boldsymbol{\pi}}^{(t)}$
3: Order($\boldsymbol{A}_\pi$) $\rightarrow \boldsymbol{\xi}$
4: $\boldsymbol{\xi} \rightarrow \hat{\boldsymbol{\pi}}^{(t)}$
5: Compute $\hat{\boldsymbol{p}}^{(t+1)}(\boldsymbol{z}^*)$
6: **if** $\hat{\boldsymbol{p}}^{(t+1)}$ converged **then**
7: Stop
8: **end if**
9: **end for**

Algorithm 7 Function Order

Input: index $\boldsymbol{A}_\pi$ with size of n_π
Output: index mapping $\boldsymbol{\xi}$

1: UniformDistr(n_π)$\rightarrow (s, e)$
2: tunePoints(s, e)$\rightarrow (s_a, e_a)$
3: Sort($\boldsymbol{A}^{(q)}_\pi$) $\rightarrow \boldsymbol{\mathcal{A}}^{(q)}_\pi$
4: **for** i in $s_a : e_a$ **do**
5: **for** j in $0 : n_\pi$ **do**
6: **if** $\mathcal{A}^{(q)}_{\pi i} = A_{\pi j}$ **then**
7: Record j to $\boldsymbol{\xi}^{(q)}$
8: **end if**
9: **end for**
10: **end for**
11: MPI_SR($\boldsymbol{\xi}^{(q)}$)
12: $\boldsymbol{\xi} = \Omega_1^{Q-1} \boldsymbol{\xi}^{(q)}$

3.2 Parallel imputation

Imputation of the P-FHDI aims at selecting M donors for each recipient in $\tilde{\mathbf{z}}_M$ in lines 1 to 6 of Algorithm 8. The FEFI fractional weights for all possible donors assigned to each recipient are computed in line 2. We employ the PPS method to randomly select M donors in lines 3 to 5 such that $\boldsymbol{w}^*_{ij} = \{w^*_{ij} \mid i \in \tilde{\boldsymbol{A}}_M; j \in \{1, \ldots, M_i\}\}$. In particular, it sorts all FEFI donors in y values by the half-ascending and half-descending order to construct successive intervals in lines 3 and 4. To prepare the results of fractional imputed

Algorithm 8 Parallel imputation

Input: raw data $\mathbf{y}$, imputation cells $\mathbf{z}$, number set of donors $\boldsymbol{M}$, cell probability $\hat{\boldsymbol{p}}(z_{obs})$
Output: imputed values $\hat{\mathbf{Y}}$

1: **for** i in $0 : \tilde{n}_M$ **do**
2: Compute $\boldsymbol{w}^*_{ij}$
3: Sort FEFI donors
4: Construct $(L_j, L_{n_{Rg}})$
5: Select M_i donors
6: **end for**
7: Order($\acute{\boldsymbol{A}}$) $\rightarrow \boldsymbol{\psi}$
8: Prepare $\hat{\mathbf{Y}}$

values $\hat{\mathbf{Y}}$, we obtain the index mapping $\boldsymbol{\psi}$ of sorted $\acute{\boldsymbol{A}}$ by Order function in line 7 by

$$\boldsymbol{\psi} = \Omega_1^{Q-1} \sum_{i=s}^{e} \sum_{j=1}^{n_A} jI(\acute{\mathcal{A}}^{(q)}_i = \acute{A}_j) \tag{33}$$

Note that the majority of computational cost occurs in line 7. Finally $\hat{\mathbf{Y}}$ has been enumerated in accordance with the index mapping $\boldsymbol{\psi}$ in line 8.

3.3 Parallel variance estimation

The majority of expensive computation happens in variance estimation because of the Jackknife estimate method of the FHDI. The parallelized variance estimation is summarized in Algorithm 9.

Algorithm 9 Parallel variance estimation

Input: raw data $\mathbf{y}$, imputation cells $\mathbf{z}$, sampling weights $\boldsymbol{w}$, index set $\boldsymbol{A}$, number of donors M, imputed values $\hat{\mathbf{Y}}$
Output: variance $\hat{\boldsymbol{V}}(\hat{Y}_{FHDI})$

1: Rep_CellP($\mathbf{z}$) $\rightarrow \hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$
2: UniformDistr(n)$\rightarrow (s, e)$
3: **for** $\forall$ k in $s : e$ **do**
4: **if** $\hat{\boldsymbol{p}}(\boldsymbol{z}^*_k) = 0$ **then**
5: Skip
6: **end if**
7: **if** $n_p > 0$ **then**
8: FW($w^{*(k)}_{ij}$) $\rightarrow w^{*(k)}_{ij}$
9: **end if**
10: Compute $\hat{\mathbf{Y}}^{K,q}_{FHDI}$
11: **end for**
12: MPI_SR($\hat{\mathbf{Y}}^{K,q}_{FHDI}$)
13: $\hat{\mathbf{Y}}^{K}_{FHDI} = \Omega_1^{Q-1}(\mathbf{Y}^{K,q}_{FHDI})$
14: Compute $\hat{\boldsymbol{Y}}_{FHDI}$
15: Compute $\hat{\boldsymbol{V}}(\hat{\boldsymbol{Y}}_{FHDI})$

Algorithm 10 Function Rep_CellP

Input: imputation cells $\mathbf{z}$
Output: cell probability $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$

1: CyclicDistr($\tilde{n}_M$)$\rightarrow (s, e)$
2: **for** $\forall$ j in $s : e$ **do**
3: Compute $\hat{\mathbf{p}}^{(q)}(\tilde{\mathbf{z}}_M)$
4: **end for**
5: MPI_SR ($\hat{\mathbf{p}}^{(q)}(\tilde{\mathbf{z}}_M)$)
6: $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M) = \Omega_1^{Q-1} \hat{\mathbf{p}}^{(q)}(\tilde{\mathbf{z}}_M)$
7: MPI_Bcast($\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$)

Before proceeding to line 2 of Algorithm 9, a preprocessing function Rep_CellP is explicitly explained to compute cell probability for unique missing patterns recursively. By parallelizing $\tilde{n}_M$ in line 1 of Algorithm 10, we can determine $\hat{\mathbf{p}}^{(q)}(\tilde{\mathbf{z}}_M)$ on each processor q in line 3 by

$$\hat{\mathbf{p}}^{(q)}(\tilde{\mathbf{z}}_M) = \sum_{j=s}^{e} \left\{ \left(\sum_{i=1}^{n} w_i\right)^{-1} \sum_{i=1}^{n} w_i \hat{\boldsymbol{\pi}}_j I(\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}) \right\} \tag{34}$$

where $\hat{\boldsymbol{\pi}}_j$ represents the conditional probability of the jth recipient's donors. Note that each $\hat{\boldsymbol{\pi}}_j$ has to be rearranged

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

according to its index mapping in Eq. (31) in advance. For instance, let $\boldsymbol{A}_\pi$ be the index set of $\hat{\boldsymbol{\pi}}$ and $\boldsymbol{\mathcal{A}}_\pi$ denotes the index set of sorted $\boldsymbol{A}_\pi$. Because of the heavy computational cost of linear searching, we implement binary search (denoted as BS) [33] to record the mapping $\boldsymbol{\xi}$ of $\mathbf{A}_\pi$ after sorting. Considering m (i.e., $n_\pi/2$) as a middle index, $\mathcal{A}_{\pi m}$ denotes the middle element of $\boldsymbol{\mathcal{A}}_\pi$. We set a left index L to be 1 and a right index R to be n_π. If $\mathcal{A}_{\pi m} \leqslant A_{\pi i}$ where $i \in \{1, 2, \ldots, n_\pi\}$, set L to be m. Otherwise, we set the R to be m. If $\mathcal{A}_{\pi m} = A_{\pi i}$, m will be the output of the BS function. Thus, we have the $\boldsymbol{\xi}^{(q)}$ by

$$\boldsymbol{\xi}^{(q)} = \sum_{i=s_a}^{e_a} \text{BS}(\mathcal{A}_{\pi i}^{(q)}) \tag{35}$$

After communication in line 5, $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$ will be assembled in line 6 and broadcast to all slave processors in line 7. After demonstration of function Rep_CellP in line 1 of Algorithm 9, by leveraging the unifrom job distribution in line 2, we can compute replicate estimator matrix $\hat{\mathbf{Y}}_{FHDI}^{K,q} \in \mathbb{R}^{(e-s)\times p}$ on processor q (Note superscript K is a simple notation for distinction) in line 10 by

$$\hat{\mathbf{Y}}_{FHDI}^{K,q} = \sum_{k=s}^{e} \hat{\boldsymbol{Y}}_{FHDI}^{k} = \sum_{k=s}^{e} \left(\sum_{l=1}^{p} \hat{Y}_{l,FHDI}^{k} \right) \tag{36}$$

where $\hat{Y}_{l,FHDI}^{k}$ is k^{th} replicate estimator of y_l. By substituting Eq. (24), we have

$$\hat{\mathbf{Y}}_{FHDI}^{K,q} = \sum_{k=s}^{e} \sum_{l=1}^{p} \left\{ \sum_{i \in A} w_i^k \{\delta_{il} y_{il} + (1-\delta_{il}) \sum_{j=1}^{M} w_{ij}^{*(k)} y_{il}^{*(j)}\} \right\} \tag{37}$$

The function FW in Eq. (21) will update $w_{ij}^{*(k)}$ if $k \notin A_M$. After communication of $\hat{\mathbf{Y}}_{FHDI}^{K,q}$ in line 12, $\hat{\mathbf{Y}}_{FHDI}^{K}$ is assembled in line 13. Also, the FHDI estimator $\hat{\boldsymbol{Y}}_{FHDI} \in \mathbb{R}^p$ is computed in line 14 by substituting Eq. (18)

$$\begin{aligned} \hat{\boldsymbol{Y}}_{FHDI} &= \sum_{l=1}^{p} \hat{Y}_{l,FHDI} \\ &= \sum_{l=1}^{p} \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1-\delta_{il}) \sum_{j=1}^{M} w_{ij}^{*} y_{il}^{*(j)}\} \end{aligned} \tag{38}$$

Eventually, we determine $\hat{\boldsymbol{V}}(\hat{\mathbf{Y}}_{FHDI})$ by

$$\hat{\boldsymbol{V}}(\hat{\boldsymbol{Y}}_{FHDI}) = \sum_{k=1}^{n} c_k (\hat{\boldsymbol{Y}}_{FHDI}^{k} - \hat{\boldsymbol{Y}}_{FHDI})^2 \tag{39}$$

where $c_k = (n-1)/n$. Similarly, we can determine $\hat{\boldsymbol{V}}(\hat{\boldsymbol{Y}}_{FEFI})$ using the same algorithm.

4 Validation

To validate the P-FHDI, it should have the same outputs as the FHDI (e.g., standard errors) using an identical dataset. The platform used in the paper for all the parallel computers is Condo 2017 of Iowa State University in [34], consisting of 192 SuperMicro servers and expandable to 324 servers. Each server has two 8-core Intel Haswell processors, 128 GB of memory, and 2.5 TB local disk. For a convenient description of synthetic datasets, let $\mathbf{U}(n, p, \eta)$ denote a finite population with n instances and p variables issued by η missing rate in proportion.

We generate $\boldsymbol{y}_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4})$, $i = 1, \ldots, 1000$, from $y_1 = 1 + e_1$, $y_2 = 2 + 0.5e_1 + 0.866e_2$, $y_3 = y_1 + e_3$, and $y_4 = -1 + 0.5y_3 + e_4$, where e_1, e_2, e_4 are randomly generated by the standard normal distribution, and e_3 by the gamma distribution. Missingness is addressed by the Bernoulli distribution as $\delta_{i1} \sim Ber(0.6)$, $\delta_{i2} \sim Ber(0.7)$, $\delta_{i3} \sim Ber(0.8)$, $\delta_{i4} \sim Ber(0.9)$ independently to each variable. We create 25% missingness to the synthetic dataset.

TABLE 5: Standard errors of naive, serial and parallel estimators.

Estimator	y_1	y_2	y_3	y_4
Naive	0.0419	0.0390	0.0490	0.0472
FEFI	0.0367	0.0378	0.0460	0.0457
P-FEFI	0.0367	0.0378	0.0460	0.0457
FHDI	0.0383	0.0372	0.0451	0.0453
P-FHDI	0.0384	0.0370	0.0449	0.0450

TABLE 6: Standard errors of the naive and the P-FHDI estimators with datasets $\mathbf{U}$(instances, variables, missing rate).

Data	Method	y_1	y_2	y_3	y_4
$\mathbf{U}(0.1M, 4, 0.25)$	Naive	0.0029	0.0039	0.0047	0.0043
	P-FHDI	0.0037	0.0034	0.0045	0.0045
$\mathbf{U}(0.5M, 4, 0.25)$	Naive	0.0013	0.0018	0.0021	0.0019
	P-FHDI	0.0017	0.0015	0.0020	0.0020
$\mathbf{U}(1M, 4, 0.25)$	Naive	0.0009	0.0012	0.0015	0.0013
	P-FHDI	0.0012	0.0011	0.0014	0.0014

On the basis of the dataset $\mathbf{U}(1000, 4, 0.25)$, we apply the naive estimator, fractional imputation estimator, and parallel fractional imputation estimator to be in contrast. The naive estimator is a simple mean estimator computed using only observed values. The results in Table 5 shows that P-FEFI produces the same standard errors of $\boldsymbol{y}$ as serial FEFI. Results generated by the P-FHDI slightly differ from results of the serial FHDI in a tolerable manner. The random number generators used for the selection of donors in the FHDI and P-FHDI are library-dependent, which is the main reason leading to the tiny residuals. Note that the standard errors of the P-FHDI decrease asymptotically as n increases gradually in Table 6. As expected, both the P-FHDI and P-FEFI often outperform the naive method in terms of standard errors. Some exceptions (e.g., y_1 and y_4 of Table 6) may be attributed to the large missing rate and simple synthetic model used in the study case. We provide the step-by-step illustration of the use of the P-FHDI in APPENDIX I. All the developed codes are shared with GPL-2, and codes, examples, and documents are available in [35].

To maximize the benefit of researchers, we provide eleven adopted datasets in the supplementary material, as summarized in Table 7. Synthetic data 1 is the dataset used to validate the P-FHDI. Synthetic data 2 is a big-n dataset with millions of instances. Synthetic data 3 to 5 are used to compare the performance of the big-n and big-p algorithms. We apply Synthetic data 6 to 8 to test the extreme cases of the P-FHDI on high-dimensional datasets. Three practical incomplete datasets, i.e., Air Quality, Nursery, and Appliance Energy, are included [28]. The hybrid dataset (Air Quality) is used to illustrate how to apply the P-FHDI to real data in APPENDIX I. Note that Air Quality consists

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

of a non-collapsible categorical variable (y_1) and three continuous variables ($y_2 \sim y_4$). We test the extension of the P-FHDI to categorical variables by Nursery. We investigate behaviors of the big-p algorithm with a growing number of selected variables by Appliance Energy. Users can practice using the P-FHDI with these example datasets.

TABLE 7: Summary of the adopted datasets $\mathbf{U}$(instances, variables, missing rate).

Dataset	Variable type	Dimension
Synthetic data 1	Continuous	$\mathbf{U}(1000, 4, 0.25)$
Synthetic data 2	Continuous	$\mathbf{U}(10^6, 4, 0.25)$
Air Quality	Hybrid	$\mathbf{U}(41757, 4, 0.1)$
Nursery	Categorical	$\mathbf{U}(12960, 5, 0.3)$
Synthetic data 3	Continuous	$\mathbf{U}(15000, 12, 0.15)$
Synthetic data 4	Continuous	$\mathbf{U}(15000, 16, 0.15)$
Synthetic data 5	Continuous	$\mathbf{U}(15000, 100, 0.15)$
Synthetic data 6	Continuous	$\mathbf{U}(1000, 100, 0.3)$
Synthetic data 7	Continuous	$\mathbf{U}(1000, 1000, 0.3)$
Synthetic data 8	Continuous	$\mathbf{U}(1000, 10000, 0.3)$
Appliance Energy	Continuous	$\mathbf{U}(19735, 26, 0.15)$

5 Cost Analysis and Scalability

It is crucial to build a cost model of the P-FHDI. It will not only evaluate the performance of the P-FHDI but also reveal potential memory risks. Note that when it comes to the Big O notation of the function $g(x)$, we are usually talking about the worst-case scenario leading to the upper time boundary $O(g(x))$ such that $g(x) < O(g(x))$ as $x \to \infty$. Considering a constant time for a unit operation in the algorithm, we can estimate the time complexity by computing the number of elementary operations. It is essential to define the elementary cost involved in computation and communication. Let α be the computational cost per unit, β be the transfer cost per unit of communication and $\mathcal{L}$ be communication startup cost. Total running time $T(Q)$ with Q available processors consists of computational cost $\mathcal{C}$ and communication cost $\mathcal{H}$, which reads:

$$T(Q) \approx \frac{\alpha^{'}}{Q} + \beta^{'} Q \tag{40}$$

where $\alpha^{'} = \alpha\psi_1(n,p)O(n^2) + \alpha O(n^3)$ and $\beta' = \beta\psi_1(n,p)O(n)$. The quantity $\psi_1(n,p)$ is the number of total iterations in cell construction, which guarantees at least two donors for each recipient. Because of the implicit property of cell construction and probability estimation, we can only make an empirical approximation that $\psi_1(n,p) \propto n\ln(p)$. In conjunction with Eq. (40), we have the scalability of $\frac{T(c_pQ)}{T(Q)}$ by

$$\frac{T(Q)}{T(c_pQ)} = c_p \times \frac{\alpha^{'} + Q^2\beta^{'}}{\alpha^{'} + c_p^2 Q^2 \beta^{'}} \tag{41}$$

where $c_p \in \mathbb{N}^+$. Detailed derivations of Eqs. (40) and (41) are presented in APPENDIX II.

To evaluate the performance of the P-FHDI in practice, we perform parametric studies to investigate the impacts of the number of instances n, the number of variables p and the missing rate η on speedup and running time. We generate synthetic datasets in the form of $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{ip}), i = 1, \ldots, n$ using the same method presented in Section 4, and the missingness is dynamically issued by $\delta_p \sim Ber(\eta_p)$. By fixing the other two parameters, nine synthetic datasets will be generated by varying the target parameter in Table 8.

TABLE 8: Datasets $\mathbf{U}$(instances, variables, missing rate) prepared for parametric studies.

Parameter	Dataset 1	Dataset 2	Dataset 3
n	$\mathbf{U}(0.5M, 4, 0.25)$	$\mathbf{U}(0.8M, 4, 0.25)$	$\mathbf{U}(1M, 4, 0.25)$
η	$\mathbf{U}(1M, 4, 0.15)$	$\mathbf{U}(1M, 4, 0.25)$	$\mathbf{U}(1M, 4, 0.35)$
p	$\mathbf{U}(15K, 12, 0.15)$	$\mathbf{U}(15K, 16, 0.15)$	$\mathbf{U}(15K, 100, 0.15)$

Note that the P-FHDI is particularly powerful towards big data with massive instances and high missing rates. The parametric studies on the number of variables will be discussed in Section 6. An increasing number of instances n and missing rate η increase the running time of the P-FHDI in Figs. 7 and 9. But n and η are not likely to significantly affect the speedup of the P-FHDI in Figs. 6 and 8. We provide speedups and running time of imputation or variance estimation of the P-FHDI separately in APPENDIX II. Overall, the number of instances is the dominating parameter positively affecting speedup and running time. All of these parametric studies have a good agreement with the cost models of speedup and running time.

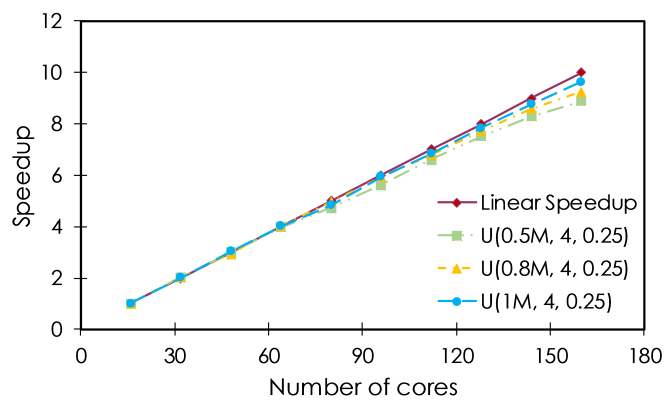


Fig. 6: Impact of the number of instances n on speedups of the entire P-FHDI (i.e., imputation and variance estimation) with datasets $\mathbf{U}(n, 4, 0.25)$: 4 variables and 25% missing rate by varying n.

6 Variable Reduction for Big-p Datasets

The current algorithm of the P-FHDI maybe not adequate for imputing big-p data with too many variables (e.g., $p > 100$). We performed a parametric study with increasing p and the results show weak speedups with the big-n algorithm in Fig. 10. A gradual increment of p significantly increases the total running time, which may be attributed to the fact that many variables can lead to exponentially increasing iterations to guarantee at least two donors during the current cell construction algorithm. This issue is theoretically related to so-called the curse of dimensionality. There is a huge literature on the problem of variable selection. To name a few, the least absolute

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

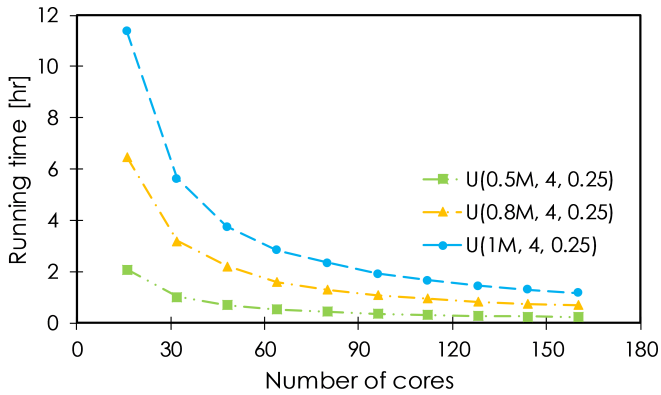


Fig. 7: Impact of the number of instances n on running time of the entire P-FHDI (i.e., imputation and variance estimation) with datasets $\mathbf{U}(n, 4, 0.25)$: 4 variables and 25% missing rate by varying n.

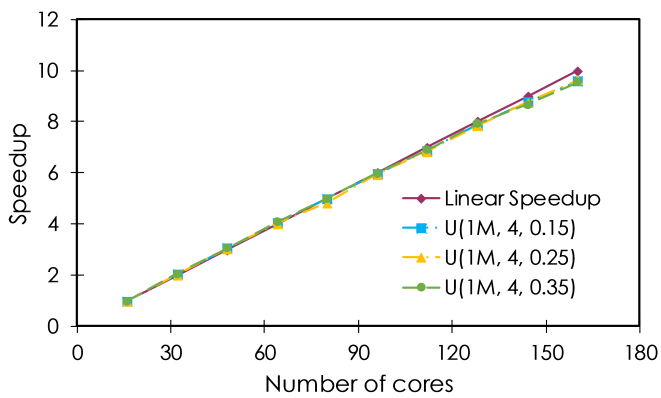


Fig. 8: Impact of the missing rate η on speedups of the entire P-FHDI (i.e., imputation and variance estimation) with datasets $\mathbf{U}(1M, 4, \eta)$: 1 million instances and 4 variables by varying η.

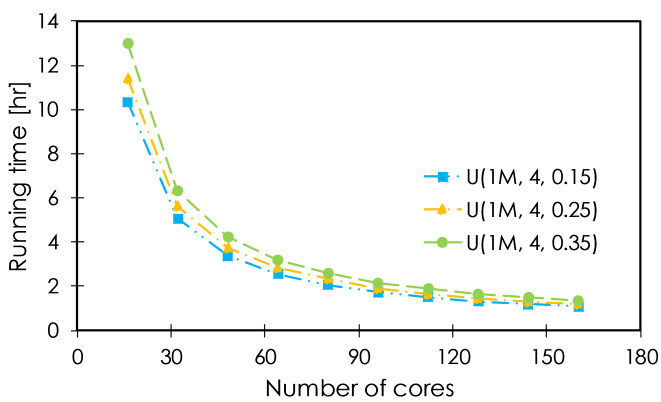


Fig. 9: Impact of the missing rate η on running time of the entire P-FHDI (i.e., imputation and variance estimation) with datasets $\mathbf{U}(1M, 4, \eta)$: 1 million instances and 4 variables by varying η.

shrinkage selection operator (LASSO) in [36], ridge in [37], principle component analysis (PCA) in [38] and smoothly clipped absolute deviation (SCAD) method in [39]. The sure independence screening (SIS) proposed in [40] is popular for ultrahigh dimensional variable reduction. It filters out the variables that have weak correlations with the response based on correlation learning. To explain the idea, consider a linear regression model

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{e} \tag{42}$$

where $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)^T$ is vector of responses, $\mathbf{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_p)$ is an $n \times p$ random design matrix with independent and identically distributed (IID) elements. $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^T$ is a vector of parameters and $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)^T$ is a IID random errors. Let $M_* = \{1 \leqslant i \leqslant p; \beta_i \neq 0\}$ be the true sparse model. The covariates $\boldsymbol{X}_i$ with $\beta_i \neq 0$ are so-called important variables, otherwise as noise variables. SIS is consist of two steps:

(1) Screening step: choose a subset of v variables such that $v < p$. For any given $\gamma \in (0, 1)$, sort the correlations in a descending order and define sub-models

$$M_\gamma = \{1 \leqslant i \leqslant p; \quad |r_i| \text{ is among the top of largest ones}\} \tag{43}$$

where $r_i = \text{corr}(\boldsymbol{X}_i, \boldsymbol{Y})$ is the sample correlation between $\boldsymbol{X}_i$ and $\boldsymbol{Y}$.

(2) Selection step: using the covariates in M_γ, apply a penalized regression method to obtain the best model.

By "sure screening property", we know that all important variables retain after applying a variable screen procedure with probability tending to 1 by

$$P(M_* \subset M_\gamma) \to 1 \tag{44}$$

as $n \to \infty$ for some given γ. Inspired by the screening step of SIS, we introduce a variable reduction method with multivariate responses embedded into the P-FHDI algorithm (so-called big-p algorithm). By assumption of a cell mean model, an instance $\{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_R\}$ serves as a donor of $\{\boldsymbol{z}_j \mid j \in \boldsymbol{A}_M\}$ unless all observed variables of $\boldsymbol{z}_j$ is identical to corresponding variables in $\boldsymbol{z}_i$. It will be difficult to guarantee at least two donors for a recipient if p is large. We apply a correlation-based screening step like SIS to filter out those variables that have weak correlations with the response variables (i.e., missing variables). Consequently, an instance $\{\boldsymbol{z}_i \mid i \in \boldsymbol{A}_R\}$ serves as a donor of $\{\boldsymbol{z}_j \mid j \in \boldsymbol{A}_M\}$ if the selected variables of $\boldsymbol{z}_j$ is identical to the corresponding variables in $\boldsymbol{z}_i$. Suppose we select v variables for a recipient such that $v < p$. Let $\mathbf{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_q\}$ be always observed and $\mathbf{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_w\}$ be subject to missingness such that $p = q + w$. Consider $\boldsymbol{r}_k = (r_k^1, r_k^2, \ldots, r_k^q)$ be a vector of sample correlations of $\boldsymbol{X}_i$, $i = \{1, \ldots, q\}$, given $\boldsymbol{Y}_k$. The proposed big-p algorithm consists of four steps:

(1) Compute correlation vectors $\boldsymbol{r}_k$ where $k \in \{1, \ldots, w\}$ and sort it in descending order.

(2) Define sub-covariate set M_k for imputing $\boldsymbol{Y}_k, k \in \{1, \ldots, w\}$ such that

$$M_k = \{1 \leqslant i \leqslant q; \quad \left|r_k^i\right| \text{ is among the top of largest } v, \text{where } r_k^i \in \boldsymbol{r}_k\}. \tag{45}$$

(3) We are implicitly assuming that

$$P(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_w \mid \boldsymbol{X}_1, \ldots, \boldsymbol{X}_q) = P(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_w \mid \boldsymbol{X}^*) \tag{46}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

where $\boldsymbol{X}^*$ is the covariates such that $M = \cap_{k=1}^{w} M_k$ or $M = \cup_{k=1}^{w} M_k$.

(4) If number of selected covariates in M equals v, then stop. Otherwise, repeat step (2) and (3) by setting $v = v + 1$ until we obtain v selected variables.

By iterations of these steps for each recipient, we can obtain the selected variables for all recipients. Following sure screening property, the probability of the true model among the built model is assumed to be

$$P(M_* \subset M) \to 1 \tag{47}$$

as $n \to \infty$.

Besides, the temporary storage of $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$ in line 1 of Algorithm 9 may need excessive memory for big-p data. Hence, we jointly embed the function Rep_CellP into the L-Replication process of variance estimation to avoid excessive storage of $\hat{\mathbf{p}}(\tilde{\mathbf{z}}_M)$. As a result of the implementation of the big-p algorithm, the weak speedups have been enhanced to the linear speedups in Fig. 10.

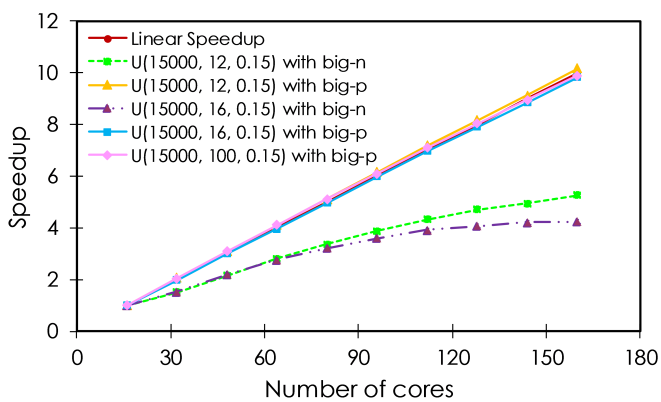


Fig. 10: Impact of the number of variables p on speedups of the P-FHDI using the big-n algorithm and the big-p algorithm, with datasets **U**(15000, p, 0.15): 15000 instances with 15% missing rate by varying p. Note that we adopt 4, 5, and 6 selected variables for **U**(15000, 12, 0.15), **U**(15000, 16, 0.15), and **U**(15000, 100, 0.15) with big-p algorithm, respectively.

We further investigate the behaviors of estimates of the big-p algorithm. Let $\boldsymbol{U}$ be a finite population of size N and $\boldsymbol{U}_g$ be subsets of the finite population with size of N_g, where $g = 1, \ldots, G$. From Eq. (III.7) in the APPENDIX III, the ratio of the means is given as

$$\frac{\mathrm{E}(Y_N) + \mathrm{E}\left(\sum_{g=1}^{G_v} \sum_{i \in \boldsymbol{U}_g} (R_g^{-1}\delta_i - 1)(y_i - \mu_g)\right)}{\mathrm{E}(Y_N)} \to 1 \tag{48}$$

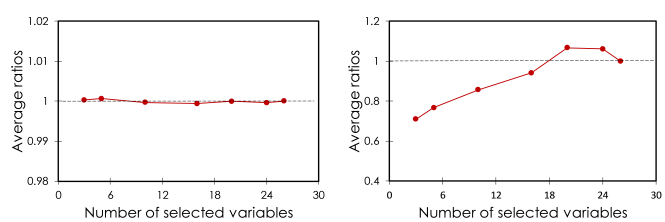
as $v \to p$, where $R_g = N_{R_g}/N_g$, $N_{R_g} = \sum_{i \in \boldsymbol{U}_g} \delta_i$, $Y_N = \sum_{i=1}^{N} y_i$, G_v is the number of imputation groups using v selected variables, and μ_g is the weighted average of y_i's in the subgroup g in the population. The parametric studies of the number of selected variables in Fig. 11 (a) show that the adoption of reduced variables won't affect the average ratio of means over all variables significantly. The choice of reduced variables will slightly change G_v of each variable, leading to additional bias in the second term of numerator. That is, the slight fluctuation in Fig. 11 (a) is explained. In future work, a theoretical number of v for a desired level of bias will be addressed. Now consider the behaviors of variance using the big-p algorithm. Let $\mathcal{W}$ be

$$\mathcal{W} = \sum_{j \neq r}^{M} \left(\frac{w^*_{ij,FEFI}}{\sum_{j \neq r} w^*_{ij,FEFI}} y_i^{*(j)} \right) - y_i^{*(r)} \tag{49}$$

where r is the index of closest donor to the kth replicate and $w^*_{ij,FEFI}$ is defined in Eq. (15). Considering the update of $w_{ij}^{*(k)}$ in Eq. (21), the ratios of $\hat{V}(\hat{Y}_{FHDI})$ using v selected variables to that using all observed variables can be derived by Eq. (III.13) in APPENDIX III as

$$\frac{\sum_{k=1}^{n} \left(\sum_{i \in A_M} w^*_{ir,FEFI,v} \mathcal{W}_v \right)^2}{\sum_{k=1}^{n} \left(\sum_{i \in A_M} w^*_{ir,FEFI,p} \mathcal{W}_p \right)^2} \to 1 \tag{50}$$

where $w^*_{ir,FEFI,v}$ and W_v are built upon v selected variables and $w^*_{ir,FEFI,p}$ and W_p are built upon all variables, respectively. The ratio will approach 1 if $v \to p$. Fig. 11 (b) shows that the average ratio of the Jackknife variance over all variables incrementally approaches 1 with a growing number of reduced variables. The sum of $w^*_{ij,FEFI}$ of the ith recipient equals one. If v is much smaller than p, a recipient will have more donors with v reduced variables such that $w^*_{ir,FEFI,v} \ll w^*_{ir,FEFI,p}$, which is the dominating factor leading to the ratio of $\hat{V}(\hat{Y}_{FHDI})$ less than 1. The distance W measures how far $y_i^{*(r)}$ is away from the mean of $M - 1$ donors of the ith recipient. Though $\hat{Y}_{FHDI}$ has additional variance due to the donor selection procedure [15]. It leads to W_p slightly smaller than W_v, which is trivial if v is much smaller than p. As $v \to p$, $w^*_{ir,FEFI,v} \simeq w^*_{ir,FEFI,p}$. Sequentially, $W_p < W_v$ will play an important role such that the ratio of $\hat{V}(\hat{Y}_{FHDI})$ can be slightly greater than 1. That is, the trend of the average ratio of $\hat{V}(\hat{Y}_{FHDI})$ in Fig. 11 (b) is explained.



(a) (b)

Fig. 11: The average ratios of (a) mean and (b) Jackknife variance estimator of the practical dataset $\mathbf{U}(19735, 26, 0.15)$ using a growing number of selected variables.

The extremely high-dimensional data we have tested with the big-p algorithm so far has 10,000 variables. Fig. 12 shows that the speedups of the P-FHDI on extremely high-dimensional data can scale up slightly lower than the linear speedup. This performance may be attributed to the fact that both parallel cell construction and estimation of cell probability use an internal parallelization within the unbreakable implicit iterations. The running time of parallel cell construction and estimation of cell probability take the majority of total execution time with the dataset $\mathbf{U}(1000, 10000, 0.3)$. Also, the adoption of the uniform distribution scheme in parallel variance estimation results in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3029146, IEEE Transactions on Knowledge and Data Engineering

the fluctuation of the speedup due to considerable workload imbalance when n is too small. Our HPC environment (8GB memory per core) had some difficulty in handling datasets with more than $10,000$ variables because of several "out of memory" issues, which may be overcome by a large-memory HPC environment. As a compromise, the present P-FHDI will give users warning messages if the requirement of memory is about to exceed the limitations.

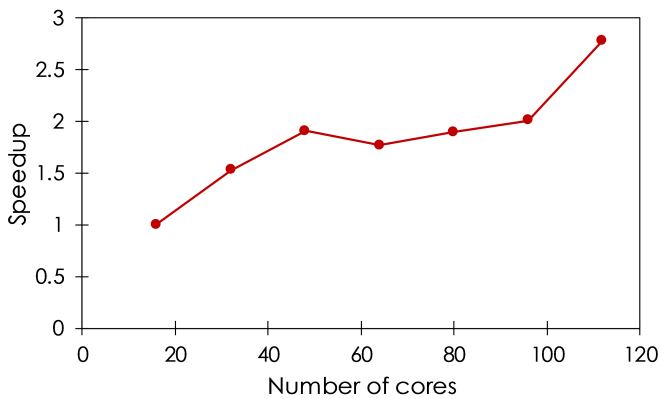


Fig. 12: Speedups of the P-FHDI with an extremely high-dimensional dataset $\mathbf{U}(1000, 10000, 0.3)$: 1000 instances and $10,000$ variables with 30% missingness. Note that we adopt 3 selected variables with the big-p algorithm.

7 Future Research

In future work, a theoretical choice of the number of selected variables v in the big-p algorithm which minimizes the Jackknife estimate of variance will be addressed. Also, the present big-p algorithm of the P-FHDI is not adequate for extremely high-dimensional categorical variables. The categorical variables violate assumptions for computing correlation, of which all variables should be continuous. One set of possible solutions rely on distance metrics such that they can find associations between categorical variables. Other possible proposals span various statistical metrics (e.g., chi-squared statistics). The marginal association measurement proposed in [41] other than correlation learning will also be a good candidate criterion to filter out unimportant categorical variables. Departing from the current program, the next theoretical (e.g., cell construction with k-nearest neighbors or fractional imputation using density ratio model) and computational advancements (e.g., prudent data distribution algorithm) shall be focusing on the ultra datasets and high-dimensional categorical data.

8 Conclusion

As we enter into the new era of big data, it is of paramount importance to establish the big data-oriented imputation paradigm. By inheriting the strengths of the general-purpose, assumption-free serial fractional hot-deck imputation program FHDI, this paper presents the first version of the parallel fractional hot-deck imputation program, named as P-FHDI. We document the full details regarding the algorithm-oriented parallelisms, computational implementations, and various examples of the P-FHDI to benefit a broad audience in the science and engineering domain. The developed P-FHDI is suitable to tackle large-instance (so-called big-n) or large-variable (so-called big-p) missing data with irregular and complex missing patterns. The validations and analytical cost models confirm that the P-FHDI exhibits promising scalability for big incomplete datasets regardless of various missing rates. This program will help to impute incomplete data, enable parallel variance estimation, and ultimately improve the subsequent statistical inference and machine learning with the cured big datasets. The next version of P-FHDI will focus on ultra datasets with both large instances and many variables, which will call for specialized theoretical and computational advancements. Toward the future extension, this first version of P-FHDI will serve as a concrete foundation. All the developed codes are shared with GPL-2, and codes, examples, and documents are available in [35].

Acknowledgments

This research is supported by the research funding of Department of Civil, Construction, and Environmental Engineering of Iowa State University. The high-performance computing facility used for this research is partially supported by the HPC@ISU equipment at ISU, some of which has been purchased through funding provided by NSF under MRI grant number CNS 1229081 and CRI grant number 1205413. This research is also supported by NSF under grants CBET-1605275 and OAC-1931380.

References

[1] J. Im, I. Cho, and J. K. Kim, "An R package for fractional hot deck imputation," *The R Journal*, vol. 10, no. 1, pp. 140–154, 2018.
[2] I. Song, Y. Yang, J. Im, T. Tong, C. Halil, and I. Cho, "Impacts of fractional hot-deck imputation on learning and prediction of engineering data," *IEEE Transactions on Knowledge and Data Engineering*, 2019.
[3] T. A. Myers, "Goodbye, listwise deletion: Presenting hot deck imputation as an easy and efficient tool for handling missing data," *Communication Methods and Measures*, vol. 5, no. 4, pp. 297–310, 1999.
[4] J. W. Graham, "Missing data analysis: Making it work in the real world," *Annual review of psychology*, vol. 60, pp. 549–576, 2009.
[5] L. Wilkinson, "Statistical methods in psychology journals: Guidelines and explanations," *American psychologist*, vol. 54, no. 8, pp. 594–604, 1999.
[6] A. J. Smola, S. Vishwanathan, and H. Thomas, "Kernel methods for missing variables," *AISTATS*, pp. 325–332, 2005.
[7] D. Williams, X. Liao, Y. Xue, and L. Carin, "Incomplete-data classification using logistic regression," *Proceedings of the 22nd International Conference on Machine Learning*, pp. 972–979, 2005.
[8] D. Rubin, "Multiple imputation after 18+ years," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 473–489, 1996.
[9] I. White, P. Royston, and A. Wood, "Multiple imputation using chained equations: Issues and guidance for practice," *Journal of the American Statistical Association*, vol. 30, no. 4, pp. 377–399, 2011.
[10] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice," *The American Statistician*, vol. 55, no. 3, pp. 244–254, 2001.
[11] S. Yang and J. K. Kim, "A note on multiple imputation for method of moments estimation," *Biometrika*, vol. 103, no. 1, pp. 244–251, 2016.
[12] X. Xie and X.-L. Meng, "Dissecting multiple imputation from a multi-phase inference perspective: what happens when god's, imputer's and analyst's models are uncongenial?" *Statistica Sinica*, vol. 27, no. 4, pp. 1485–1594, 2017.
[13] K. Graham and L. Kish, "Some efficient random imputation methods," *Communication in Statistic-Theory and Methods*, vol. 13, no. 16, pp. 1919–1939, 1984.

[14] R. E. Fay, "Alternative paradigms for the analysis of imputed survey data," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 490–498, 1996.

[15] J. K. Kim and W. Fuller, "Fractional hot deck imputation," *Biometrika*, vol. 91, no. 3, pp. 559–578, 2004.

[16] S. Yang and J. K. Kim, "Fractional imputation in survey sampling: A comparative review," *Statistical Science*, vol. 31, no. 3, 2016.

[17] W. A. Fuller and J. K. Kim, "Hot deck imputation for the response model," *Survey Methodology*, vol. 31, no. 2, pp. 139–149, 2005.

[18] E. F. Akmam, T. Siswantining, S. M. Soemartojo, and D. Sarwinda, "Multiple imputation with predictive mean matching method for numerical missing data," *International Conference on Informatics and Computational Science*, pp. 1–6, 2019.

[19] Nurzaman, T. Siswantining, S. M. Soemartojo, and D. Sarwinda, "Application of sequential regression multivariate imputation method on multivariate normal missing data," *International Conference on Informatics and Computational Science*, pp. 1–6, 2019.

[20] K. Aristiawati, T. Siswantining, D. Sarwinda, and S. M. Soemartojo, "Missing values imputation based on fuzzy c-means algorithm for classification of chronic obstructive pulmonary disease," *AIP Conference Proceeding*, vol. 2192, no. 1, p. 060003, 2019.

[21] T. Anwar, T. Siswantining, D. Sarwinda, S. M. Soemartojo, and A. Bustamam, "A study on missing values imputation using k-harmonic means algorithm: Mixed datasets," *AIP Conference Proceeding*, vol. 2202, no. 1, p. 020038, 2019.

[22] IBM, 2012. [Online]. Available: http:// www-01.ibm.com/software/data/bigdata/

[23] B. Caffo, R. Peng, F. Dominici, T. A. Louis, and S. Zeger, Eds., *Parallel MCMC Imputation for Multiple Distributed Lag Models: A Case Study in Environmental Epidemiology*. The Handbook of Markov Chain Monte Carlo, 2011.

[24] T. J. Durham, M. W. Libbrecht, J. Howbert, J. Bilmes, and W. S. Noble, "Predictd parallel epigenomics data imputation with cloud-based tensor decomposition," *Nature communication*, vol. 9, no. 1, pp. 1402–1402, 2018.

[25] B. Zhang, D. Zhi, K. Zhang, G. Gao, N. N. Limdi, and N. Liu, "Practical consideration of genotype imputation: Sample size, window size, reference choice, and untyped rate," *Statistics and its interface*, vol. 4, no. 3, pp. 339–352, 2011.

[26] E. Porcu, S. Sanna, C. Fuchsberger, and L. G. Fritsche, "Genotype imputation in genome-wide association studies," *Current protocols in human genetics*, vol. 78, no. 1, pp. 1–25, 2013.

[27] X. Hu, "Acceleration genotype imputation for large dataset on gpu," *Procedia Environmental Science*, vol. 8, pp. 457–463, 2011.

[28] D. Dheeru and G. Casey, "Uci machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[29] J. Im, J. K. Kim, and W. A. Fuller, "Two-phase sampling approach to fractional hot deck imputation," *In Proceedings of Survey Research Methodology Section*, pp. 1030–1043, 2015.

[30] S. Z. Christopher, T. Siswantining, D. Sarwinda, and A. Bustaman, "Missing value analysis of numerical data using fractional hot deck imputation," *International Conference on Informatics and Computational Science*, pp. 1–6, 2019.

[31] J. G. Ibrahim, "Incomplete data in generalized linear models," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 765–769, 1990.

[32] I. Cho and K. A. Porter, "Multilayered grouping parallel algorithm for multiple-level multiscale analyses," *International Journal for Numerical Methods in Engineering*, vol. 100, no. 12, pp. 914–932, 2014.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Eds., *Introduction to Algorithms*. London: MIT press, 2009.

[34] Condo, "Condo2017: Iowa state university high-performance computing cluster system," 2017. [Online]. Available: https://www.hpc.iastate.edu/guides/condo-2017/slurm-job-script-generator-for-condo

[35] I. Cho, "Data-driven, computational science and engineering platforms repository," 2017. [Online]. Available: https://sites.google.com/site/ichoddcse2017/home/type-of-trainings/parallel-fhdi-p-fhdi-1

[36] T. Robert, "Regression shrinkage and selection via lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

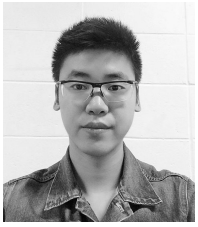
[37] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[38] L. Jolliffe, Ed., *Principle component analysis*. Berlin Heidelberg: Springer, 2011.

[39] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[40] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911, 2008.

[41] D. Huang, R. Li, and H. Wang, "Feature screening for ultrahigh dimensional categorical data with applications," *Journal of Business Economic Statistics*, vol. 32, no. 2, pp. 237–244, 2014.



Yicheng Yang received the MS degree in civil engineering and minored in computer science from Iowa State University (ISU) in 2018. He is currently a PhD student in civil engineering (specialization: intelligent infrastructure engineering) from the department of civil, construction and environmental engineering (CCEE) of ISU in 2019. His research interests include parallel imputation, machine learning, and data-driven engineering.

Jae-Kwang Kim received the PhD degree in Statistics from ISU in 2000. He is a fellow of American Statistical Association and Institute of Mathematical Statistics and currently a LAS Dean's professor in the department of statistics at ISU. His research interests include survey sampling, statistical analysis with missing data, measurement error models, multi-level models, causal Inference, data integration, and machine Learning.

In Ho Cho (corresponding author) received the PhD degree in civil engineering and minor in Computational Sci and Eng from California Institute of Technology, USA in 2012. He is currently an assistant professor of the department of CCEE at ISU. His research interests include data-driven engineering and science, computational statistics, parallel computing, parallel multi-scale finite element analysis, computational and engineering mechanics for soft micro-robotics and nano-scale materials.