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 FHDI, we developed a new version of a parallel fractional hot deck imputation (named as P-FHDI) program suitable for curing large incomplete data set. Results show a favorable speedup when P-FHDI is applied to big data sets with up to millions of instances or hundreds of variables. This paper explains the detailed parallel algorithms of P-FHDI for large instances (big-n) and high-dimensionality (big-p) data sets 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 software for handling missing values?

There is a variety of spectrum 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 bias 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 model-based method [7] or the use of an imputation theory 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, SAS, and S-Plus [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]. FHDI can cure multivariate, general incomplete data sets with irregular missing patterns without resorting to distributional assumptions or expert statistical models. It also offers variance estimation using the jackknife replication method.

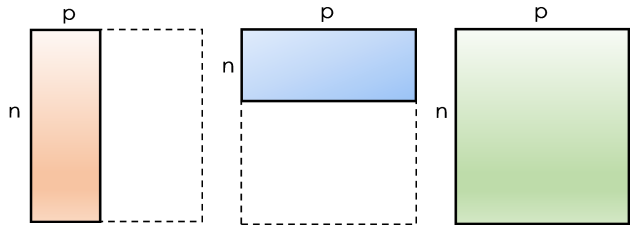
As we enter into the new era of big data [18], harnessing the potential benefits of analyzing a large amount of

- *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. Kim is with the Department of Statistics, Iowa State University, Ames, IA, 50011.*
  E-mail: jkim@iastate.edu

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

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. [19] parallelized single-chain rules of the sampler as a first attempt by using parallel Markov chain Monte Carlo (MCMC) 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. [20] 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 [21]. 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. [22] introduced genotype imputation using 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. [23] 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 the size of the data set. 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 released server not yet implements any paralleled imputation algorithm. The missing values are simply omitted during computation.

Despite its generality and efficiency, the serial version FHDI shows 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 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 (a) and (b). Our program 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) $n \gg p$; (b) $n \leqslant p$; (c) n and p are both very large.

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. For example, the impacts of FHDI on the subsequent machine learning have been investigated by [2]; see Fig. 2. Their results show that FHDI has a noticeable positive influence on improving the subsequent machine learning and statistical prediction compared to a simple naive method, which cures missing data using the mean value of attributes. This prior investigation underpins the significance of P-FHDI on big-data oriented machine learning methods and statistical learning.

The outline of the paper is structured as follows: we briefly demonstrate the backbone theories of 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 P-FHDI. We validate and evaluate the performance of the P-FHDI with the synthetic data sets, and propose a cost model. Finally, we introduce an updated approach embedded in P-FHDI, particularly for imputing big-p data sets. And comprehensive examples in the Appendix illustrate how to use the program easily with simple and practical data.

2 Key Algorithms of Serial FHDI

For a comprehensive description of 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 i^{th} instance of the l^{th} 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 i^{th} 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 i^{th} row of $\mathbf{z}$.

Let $\boldsymbol{A}$ be the index set in the size of n selected from the finite population $\mathbf{U}$. Let $\boldsymbol{A}_M$ denote a set of sample indices with at least one missing values such that $\boldsymbol{A}_M = \{i \in \boldsymbol{A}; \prod_{l=1}^{p} \delta_{il} = 0\}$, alternatively $\boldsymbol{A}_R$ with full observed

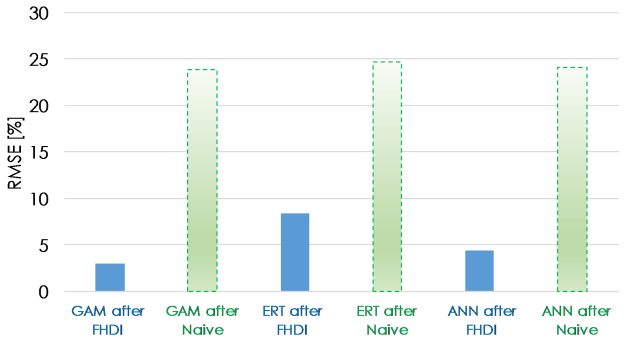


Fig. 2: The impact of data curing on the subsequent machine learning methods [artificial neural networks (ANN) and extremely randomly tree (ERT)] and statistical learning [generalized additive model (GAM)]. The fractional hot deck imputation uses the serial package FHDI and a mean value-based naive data-curing method is used for comparison.

values where $\boldsymbol{A}_R = \{i \in \boldsymbol{A}; \prod_{l=1}^{p} \delta_{il} = 1\}$, where δ_{il} is a response indicator function that 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 $\tilde{\boldsymbol{A}}_R = \{i, j \in \boldsymbol{A}_R \mid \forall i \neq j,\ \boldsymbol{z}_i \neq \boldsymbol{z}_j\}$ in 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, the imputed value y_{il}^* will be selected from the same j^{th} donor to fill in.

On the basis of generated imputation cells, we assume a cell mean model such that:

$$\boldsymbol{y} \mid (\boldsymbol{z} = g) \sim ii(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g),\ g \in \{1, \cdots, G\}, \tag{1}$$

where $\{1, \cdots, G\}$ is the total set of imputation cells, $\sim ii$ denotes independently and identically distributed, $\boldsymbol{\mu}_g$ is the mean vector of $\boldsymbol{y}$ at cell g, and $\boldsymbol{\Sigma}_g$ is the variance-covariance matrix of $\boldsymbol{y}$ in cell g. 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

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

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 proposed an approach to generate imputation cells $\mathbf{z}$ using the estimated sample quantiles in [24].

Considering the estimated distribution function of y_l:

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

where I is an indicator function taking the value of one if true and zero if false and w_i represents the sampling weight of element i. For given $a_1, \cdots, a_G$ satisfying $0 < a_1 < \cdots < a_G$, the estimated sampling quantile of y_l for a_k is defined as:

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

Once given estimated sample quantiles $\hat{q}_{(a_k)}$, define $z_{il} = g$ only if y_{il} falls into the corresponding sample quantiles range. Therefore y_{il} will be categorized as the 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 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

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

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

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

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. 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

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

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

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

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 i^{th} recipient $\boldsymbol{z}_i = \{\boldsymbol{z}_{i,obs}, \boldsymbol{z}_{i,mis}\}$

where $\boldsymbol{D}_i = \{j \in \tilde{\boldsymbol{A}}_R \mid \boldsymbol{z}_{i,obs} = \boldsymbol{z}_{j,obs}\}$ in size of M_i. M_i represents the number of donors of i^{th} 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

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

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

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

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 recursively 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 collapse 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 of $(NA, 1)$.

TABLE 1: An illustrative example for cell collapsing with initial categories of 3 for p_1 and p_2. Left: initial categorized 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

2.2 Estimation of cell probability

To estimate conditional cell probability using modified EM algorithm by weighting [25], we consider subgroups of A_M on basis of $\boldsymbol{z}$ into G groups, denoted by $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_G$ corresponding patterns $\{\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{g,mis}\}, \{\boldsymbol{z}_{g,mis}, \boldsymbol{z}_{g,obs}\}$ and $\{\boldsymbol{z}_{g,mis}, \boldsymbol{z}_{g,mis}\}$. Therefore, we will have $A_{Mg} = \{j \in A_M; \boldsymbol{z}_{j,obs} = \boldsymbol{z}_{g,obs}\}$ with a size of n_{Mg} and $A_{Rg} = \{j \in A_R; \boldsymbol{z}_{j,obs} = \boldsymbol{z}_{g,obs}\}$ in a size of n_{Rg}. The size of all possible donors of $\boldsymbol{z}_{g,mis}$ is H_g. If one possible imputed value $\boldsymbol{z}_{g,mis}^{*(h)}$ is imputed for $\boldsymbol{z}_{g,mis}$, the the estimated conditional cell probability is defined in E step:

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

where $\hat{P}^{(t)}(\boldsymbol{z}_{g,mis}^{*(h)})$ is an estimated joint cell probability of $\boldsymbol{z}_{g,mis}^{*(h)}$ computed from the full respondents at iteration t. $\hat{\pi}_h^{(t)}$ will be sorted regarding global index set $\boldsymbol{A}$ in the ascending order. Then joint cell probability is updated in M step using weighting:

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

where n is the size of sample $\boldsymbol{A}$ and w_i is replicate weight. I represents an indicator function that takes value of one if $\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{g,obs}$ is true 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 $\sum_{h=1}^{H_g} \pi_h = 1$.

2.3 Imputation

It is necessary to determine fractional weight priorly. Let w_{ij}^* be the fractional weights of j^{th} donor for the i^{th} 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 j^{th} imputed value of M_i possible donors for i^{th} recipient. In addition, we can verify computation of w_{ij}^* by

$$\sum_{j=1}^{M_i} w_{ij}^* = 1 \tag{14}$$

where M_i denotes number of donors for i^{th} 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:

$$\hat{Y}_{l,FEFI} = \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j \in A} w_{ij,FEFI}^* y_{jl}\} \tag{15}$$

and $w_{ij,FEFI}^*$ is the fractional weights of the j^{th} donor for the i^{th} recipient as

$$w_{ij,FEFI}^* = \sum_{g=1}^{G} a_{ig} \sum_{h=1}^{H} \hat{\pi}_{h|g} \frac{w_j \delta_j a_{jgh}}{\sum_{l \in A} w_l \delta_l a_{lgh}} \tag{16}$$

where $a_{jgh} = 1$ if $(\boldsymbol{z}_{j,obs}, \boldsymbol{z}_{j,mis}) = (\boldsymbol{z}_{g,obs}, \boldsymbol{z}_{g,mis}^{*(h)})$, otherwise 0. w_i is replicate weight. However, it requires too much computation cost in practical with large input data. Instead of using all the respondents, FHDI selects M donors among all FEFI donors using a systematic probability proportional to size (PPS) method to compensate for recipient as followings:

(a) Sort all possible FEFI donors.

(b) Construct the interval of $(L_j, L_{n_{Rg}})$.

(b1) $L_1 = 0$

(b2) for $\forall\, j \in [1 : n_{Rg}]$
$L_{j+1} = L_j + M \times w_{ij,FEFI}^*$;

(c) Select M donors in terms of the constructed interval.

In step (a), we sort all possible FEFI donors in terms of values of z by half-ascending and half-descending order. In hope of computing n_{Rg} breakpoints within the interval, L_j will be updated recursively regarding (b2). Let RN_g be a random number from a uniform distribution $U(0,1)$. Then for $i \in A_{Mg}$, the M donors will be selected as:

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

where $l \in [1:M]$. In case of $n_{Rg} \leqslant M$, we take all possible FEFI donors for the recipient.

Once selected M donors regarding probability proportional to w_{ij}^*, then FHDI estimator of y_l is

$$\hat{Y}_{l,FHDI} = \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w_{ij}^* y_{il}^{*(j)}\} \quad (17)$$

where $w_{ij}^* = M^{-1}$, w_i is replicate weight and $y_{il}^{*(j)}$ is the j^{th} imputed value of y_{il}.

Afterward, we can prepare imputation results $\hat{\mathbf{Y}} = \{y_{il}^{*(j)} \mid i \in \{1,\ldots,n\}; l \in \{1,\ldots,p\}; j \in \{1,\ldots,M_i\}\}$ 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

$$\boldsymbol{\psi} = \sum_{i=1}^{n_A} \sum_{j=1}^{n_A} j I(\acute{\mathcal{A}}_i = \acute{A}_j) \quad (18)$$

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

S1: Identification of the joint probability $\hat{\mathbf{p}}(\mathbf{z}^*)$ of unique patterns $\mathbf{z}^* = \tilde{\mathbf{z}}_R \cup \tilde{\mathbf{z}}_M$ of size $\tilde{n}$ by

$$\hat{\mathbf{p}}(\mathbf{z}^*) = \sum_{j=1}^{\tilde{n}} \hat{\boldsymbol{p}}(\boldsymbol{z}_j^*) = \sum_{j=1}^{\tilde{n}} \left(\sum_{i \in A}^{n} w_i \right)^{-1} \sum_{i \in A}^{n} w_i \hat{\boldsymbol{\pi}}^* I(\boldsymbol{z}_{i,obs} = \boldsymbol{z}_{g,obs}), \quad (19)$$

where $\tilde{n} = \tilde{n}_M + \tilde{n}_R$, $\hat{\boldsymbol{\pi}}^*$ represent the conditional probability in case of the convergence of modified EM algorithm. Note that if the size of $\hat{\boldsymbol{p}}(\boldsymbol{z}_j^*)$ is 0, skip to the next iteration.

S2: Delete unit $k \in A$, if $k \in A_M$, then w_{ij}^* will not be updated.

S3: if $k \notin A_M$, then w_{ij}^* for replicate k will be updated by the function FW:

$$w_{ij}^{*(k)} = \begin{cases} w_{ij}^* - w_{ij,FEFI}^* & \text{if } j = r \\ w_{ij}^* + (w_{ir,FEFI}^*) \frac{w_{ij,FEFI}^*}{\sum_{j \neq r} w_{ij,FEFI}^*} & \text{if } j \neq r \\ w_{ij}^* & \text{if } k \in A_M \end{cases} \quad (20)$$

where $i \in A_R$, r is the index of the closest donor to k. However, measurement of closeness in terms of the scale of the data is challenging. We employ Mahalanobis Distance (denoted as D_M) to measure the distance from replicate k to the other M donors by

$$\boldsymbol{D}_M = \sqrt{(y - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})} \quad (21)$$

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

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

$$\hat{V}(\hat{Y}_{l,FHDI}) = \sum_{k=1}^{n} c_k (\hat{Y}_{l,FHDI}^k - \hat{Y}_{l,FHDI})^2 \quad (22)$$

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

$$\hat{Y}_{l,FHDI}^k = \sum_{i \in A} w_i^k \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w_{ij}^{*(k)} y_{il}^{*(j)}\} \quad (23)$$

where w_i^k is the k^{th} replicate of the imputation fractional weight w_{ij}^*. And $y_{il}^{*(j)}$ is the j^{th} imputed value of y_{il}. The FHDI estimator is defined in Eq. (17). Similarly, we can determine the $\hat{V}(\hat{Y}_{l,FEFI})$ using the same equations.

3 Parallel Algorithms for FHDI

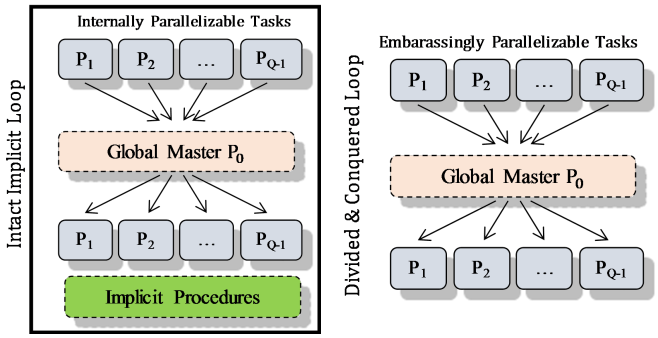


Fig. 3: Two parallel schemes: (a) Internal parallelization within the unbreakable implicit loop; (b) Typical divide and conquer for embarrassingly parallelizable explicit loop.

Fig. 3 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 the 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 works 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 works distributed to all slave processors ranking i to j. Generally, if the entire domain Υ of a problem divided into disjoint domain Υ_q, no memory overlapping is required and thus all work is done concurrently by

$$\Upsilon = \Omega_{q=1}^{Q-1} \left(\sum_{x_i \in \Upsilon_q} S(x_i) \right) \tag{24}$$

where $q \in [1, Q-1]$. 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 aim, we employ two work distribution schemes in 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 works on processor q. Fig. 4 illustrates visually the contrast between two work distribution schemes. These two schemes are summarized in Algorithm 1 and 2.

Algorithm 1 Uniform job distribution

Input: number of total instance 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: $e = (q-1)N_1 + N_2$

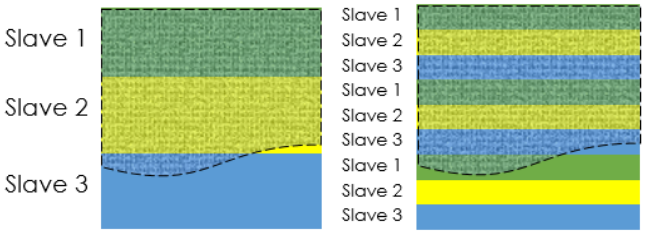
The uniform distribution scheme in Algorithm 1 averages works in N_1 pieces over slave processors and squeezes the residuals to the last available processor as N_2. Then the boundary indices s and e of jobs in the current processor q can be computed in lines 3 and 4, respectively. Alternatively, the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

Algorithm 2 Cyclic job distribution

Input: number of total instance 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 uniform distribution, this uniform decomposition will lead to considerable work imbalance when the problem domain Υ is severely irregular [26]. See Fig. 4 (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. 4 (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. 4: Different methods distributing works 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 (Green), Slave 2 (Yellow) and Slave 3 (Blue).

3.1 Parallel cell construction and estimation of cell probability

The determination of initial imputations 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 parallel techniques within the iterations in Algorithm 3. In line 2 of Algorithm 3, we categorize input data $\mathbf{y}$ to categorical data $\mathbf{z}$ 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 line 1 and 2 of Algorithm 4, we employ cyclic distribution for job division and adjust boundary accordingly. The function tunePoints has been frequently used in the P-FHDI to adjust the border indices (s and e) of works 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 of 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 2 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 2. Hence the updated mapping will be $\boldsymbol{\psi}^1 = \{7,2,5\}$, $\boldsymbol{\psi}^2 = \{3,4\}$ and $\boldsymbol{\psi}^3 = \{1,6\}$.

TABLE 2: 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

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

$$\tilde{\mathbf{z}}_M^{(q)} = \bigotimes_{i=s_a}^{e_a} \boldsymbol{z}_{Mi} I(\|\boldsymbol{z}_{Mi}\| < \|\boldsymbol{z}_{M(i+1)}\|) \tag{25}$$

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

$$\tilde{\mathbf{z}}_R^{(q)} = \bigotimes_{i=s_a}^{e_a} \boldsymbol{z}_{Ri} I(\|\boldsymbol{z}_{Ri}\| < \|\boldsymbol{z}_{R(i+1)})\| \tag{26}$$

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 have the minimum number of donors of recipients m. 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 lines 3 by

$$\boldsymbol{M}^{(q)} = \bigotimes_{i=s}^{e} \left\{ \sum_{j=1}^{\tilde{n}_R} I(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \right\} \tag{27}$$

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

$$\mathbf{L}^{(q)} = \bigotimes_{i=s}^{e} \bigotimes_{j=1}^{\tilde{n}_R} jI(\|\boldsymbol{z}_{i,obs}\| = \|\boldsymbol{z}_{j,obs}\|) \tag{28}$$

Then communication is processed between line 6 and line 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 processors 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 and 9. Otherwise, the iterations will continue to the max iteration (i.e., $2n$).

Similarly, estimation of cell probability is an implicit and iterative process, which does not support simple parallelism. In line 1 of Algorithm 6, $\tilde{\mathbf{z}}_M$ and $\tilde{\mathbf{z}}_R$ have been reproduced. The EM iterations start in line 2 till terminating in case of convergence of $\hat{\boldsymbol{p}}(z_{obs}, z^*_{mis})$ in lines 7 to 9. Let $\boldsymbol{A}^\pi = \{1, \ldots, H_G\}$ be local index set of $\hat{\boldsymbol{\pi}}^{(t)}$ in 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 3 by Eq. (11), where $\hat{\boldsymbol{\pi}}^{(t)}$ is

$$\hat{\boldsymbol{\pi}}^{(t)} = \bigotimes_{g=1}^{G} \sum_{h=1}^{H_g} \hat{\pi}_h^{(t)} \tag{29}$$

Note that $\hat{\boldsymbol{\pi}}^{(t)}$ requires reorganization regarding the index set $\boldsymbol{\mathcal{A}}_\pi = \{\mathcal{A}_{\pi i} \in \boldsymbol{A}_\pi \mid \mathcal{A}_{\pi(i+1)} > \mathcal{A}_{\pi i}\}$ in ascending order.

Algorithm 3 Parallel cell construction

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

1: **for** $\forall\ i$ in $0 : max\_iteration$ **do**
2: Categorize raw data $\mathbf{y}$ to categorical data $\mathbf{z}$
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: Categorical $\mathbf{z}$
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: Record $\tilde{\mathbf{z}}_M^{(q)}$
5: **end for**
6: MPI_SR($\tilde{\mathbf{z}}_M^{(q)}$)
7: $\tilde{\mathbf{z}}_M = \Omega_1^Q \tilde{\mathbf{z}}_M^{(q)}$
8: CyclicDistr(n_M) $\rightarrow (s, e)$
9: tunePoints(s, e) $\rightarrow (s_a, e_a)$
10: **for** $\forall\ i$ in $s : e$ **do**
11: Record $\tilde{\mathbf{z}}_R^{(q)}$
12: **end for**
13: MPI_SR($\tilde{\mathbf{z}}_R^{(q)}$)
14: $\tilde{\mathbf{z}}_R = \Omega_1^Q \tilde{\mathbf{z}}_R^{(q)}$
15: MPI_Bcast($\tilde{\mathbf{z}}_M; \tilde{\mathbf{z}}_R$)

Before proceeding to line 5 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

$$\boldsymbol{\xi}^{(q)} = \bigotimes_{i=s}^{e} \bigotimes_{j=0}^{n_\pi} jI(\mathcal{A}_{\pi i}^{(q)} = A_{\pi j}) \tag{30}$$

After communication in line 11, $\boldsymbol{\xi}$ will be produced in line 12. By leveraging $\boldsymbol{\xi}$ in line 5 of Algorithm 6, $\hat{\boldsymbol{\pi}}^t$ can be queued for later computation of $\hat{\boldsymbol{P}}^{(i)}(z_{obs}, z^*_{mis})$ in line 6 by

$$\hat{\boldsymbol{P}}^{(i)}(z_{obs}) = \left(\sum_{i \in A}^{n^*} w_i \right)^{-1} \sum_{i \in A}^{n^*} w_i \hat{\boldsymbol{\pi}}^{(t)} I(z_{i,obs} = \boldsymbol{z}_{g,obs}). \tag{31}$$

where $n^* = n_R + n_M \times M_i$ representing size of augmented imputed cells. In line 7, if difference of $\hat{\boldsymbol{P}}^{(i)}(z_{obs}, z^*_{mis})$ between iterations is lesser than a threshold ϵ (e.g., 10^{-6}), the EM algorithm will be terminated.

3.2 Parallel imputation

The parallel imputation is illustrated as below:

Imputation of 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

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

Algorithm 6 Parallel estimation of cell probability

Input: Categorized data $\mathbf{z}$, sampling weight $\boldsymbol{w}$
Output: cell probability $\hat{\boldsymbol{p}}(z_{obs})$

1: Memorize $\tilde{\mathbf{z}}_R, \tilde{\mathbf{z}}_M$
2: **for** i in $0 : max\_i$ **do**
3: $\quad$ Update $\boldsymbol{w}$ and $\hat{\boldsymbol{\pi}}^{(t)}$
4: $\quad$ Order$(\boldsymbol{A}_\pi) \rightarrow \boldsymbol{\xi}$
5: $\quad \boldsymbol{\xi} \rightarrow \hat{\boldsymbol{\pi}}^{(t)}$
6: $\quad$ Compute $\hat{\boldsymbol{p}}^{(i)}(z_{obs}, z^*_{mis})$
7: $\quad$ **if** $\hat{\boldsymbol{p}}^{(i+1)}(z_{obs}, z^*_{mis})$ converged **then**
8: $\quad\quad$ Stop
9: $\quad$ **end if**
10: **end for**

Algorithm 7 Function Order

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

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

Algorithm 8 Parallel imputation

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

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

recipient are computed in line 2. We employ PPS method to randomly select M donors in lines 3 to 5 with $\boldsymbol{w}^*_{ij} = \{w^*_{ij} \mid i \in \tilde{\boldsymbol{A}}_M; j \in \{1, \ldots, M_i\}\}$. Particularly, it sorts $\acute{\boldsymbol{A}}$ in half-ascending and half-descending order in line 3. To prepare the results of fractional imputed values $\hat{\mathbf{Y}}$, we obtain the index mapping $\boldsymbol{\psi}$ of particularly sorted $\acute{\boldsymbol{A}}$ by Order function in line 7

$$\boldsymbol{\psi} = \Omega_1^p \sum_{i=s}^{e} \sum_{j=0}^{\zeta} j I(\acute{A}_i = \tilde{A}_j) \tag{32}$$

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 and excessive memory issues happen in variance estimation because of the Jackknife estimate method of FHDI. The parallelized variance estimation is summarized in Algorithm 9.

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

$$\begin{aligned} \hat{\mathbf{p}}^{(q)}(\mathbf{z}^*) &= \sum_{j=s}^{e} \hat{\boldsymbol{p}}(\boldsymbol{z}^*_j) \\ &= \sum_{j=s}^{e} \left\{ \left(\sum_{i \in A}^{n} w_i \right)^{-1} \sum_{i \in A}^{n} w_i \hat{\boldsymbol{\pi}}^* I(z_{i,obs} = \boldsymbol{z}_{g,obs}) \right\} \end{aligned} \tag{33}$$

where $\hat{\boldsymbol{\pi}}^*$ represents the conditional probabilities after convergence. Note that $\hat{\boldsymbol{\pi}}^*$ in size of n_π have to be queued according to the index mapping $\boldsymbol{\xi}$ defined in the Eq. (30) in advance. Recall that $\boldsymbol{A}_\pi$ is 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) [27] 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 $\mathcal{A}_{\pi m} \leqslant \mathcal{A}_{\pi i}$ where $i \in \{1, 2, \ldots, n_\pi\}$, set L to be m. Otherwise, we set the R to be m. If $\mathcal{A}_{\pi m} = \mathcal{A}_{\pi i}$, m will be the output of the BS function. Thus, we have the $\boldsymbol{\xi}^{(q)}$ by

$$\boldsymbol{\xi}^{(q)} = \sum_{i=s}^{e} \text{BS}(\mathcal{A}_{\pi i}) \tag{34}$$

After communication in line 5, $\hat{\mathbf{p}}(\boldsymbol{z}^*) \in \mathbb{R}^{\tilde{n} \times \tilde{n}_R}$ 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 $\hat{\mathbf{Y}}^{K,q}_{FHDI} \in \mathbb{R}^{(e-s) \times p}$ on processor q (Note superscript K is a simple notation for distinction) in line 10 by

$$\hat{\mathbf{Y}}^{K,q}_{FHDI} = \sum_{k=s}^{e} \hat{\boldsymbol{Y}}^{k}_{FHDI} = \sum_{k=s}^{e} \left(\sum_{l=1}^{p} \hat{Y}^{k}_{l,FHDI} \right) \tag{35}$$

where $\hat{Y}^{k}_{l,FHDI}$ is k^{th} replicate estimator of y_l. By substituting Eq. (23), we have

$$\hat{\mathbf{Y}}^{K,q}_{FHDI} = \sum_{k=s}^{e} \sum_{l=1}^{p} \left\{ \sum_{i \in A} w^k_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w^{*(k)}_{ij} y^{*(j)}_{il}\} \right\} \tag{36}$$

The function FW defined in Eq. (20) will update $w^{*(k)}_{ij}$ if the number of all deleted donors $n_p > 0$. After communication

Algorithm 9 Parallel variance estimation

Input: raw data $\mathbf{y}$, categorized data $\mathbf{z}$, 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 $\rightarrow \hat{\mathbf{p}}(\mathbf{z}^*)$
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($w_{ij}^{*(k)}$) $\rightarrow w_{ij}^{*(k)}$
9: **end if**
10: Compute $\hat{\mathbf{Y}}_{FHDI}^{K,q}$
11: **end for**
12: MPI_SR($\hat{\mathbf{Y}}_{FHDI}^{K,q}$)
13: $\hat{\mathbf{Y}}_{FHDI}^{K} = \Omega_1^Q(\mathbf{Y}_{FHDI}^{K,q})$
14: Compute $\hat{\boldsymbol{Y}}_{FHDI}$
15: Compute $\hat{\boldsymbol{V}}(\hat{\boldsymbol{Y}}_{FHDI})$

Algorithm 10 Function Rep_CellP

Input: number of unique patterns $\tilde{n}$
Output: cell probability $\hat{\mathbf{p}}(\mathbf{z}^*)$

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

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. (17)

$$\begin{aligned}\hat{\boldsymbol{Y}}_{FHDI} &= \sum_{l=1}^{p} \hat{Y}_{l,FHDI} \\ &= \sum_{l=1}^{p} \sum_{i \in A} w_i \{\delta_{il} y_{il} + (1 - \delta_{il}) \sum_{j=1}^{M} w_{ij}^* y_{il}^{*(j)}\}\end{aligned} \tag{37}$$

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

$$\hat{\boldsymbol{V}}(\hat{\boldsymbol{Y}}_{FHDI}) = \sum_{k=1}^{n} c_k (\hat{\boldsymbol{Y}}^k_{FHDI} - \hat{\boldsymbol{Y}}_{FHDI})^2 \tag{38}$$

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 supposes to have the same outputs with FHDI (e.g., standard errors) using identical synthetic data. The platform used in the paper for all the parallel computers is Condo 2017 of Iowa State University in [28], 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 the synthetic datasets, let $\mathbf{U}(n, p, \eta)$ denotes a finite population with n instances and p variables issued by η missing rate in percentage. Here, we adopts the synthetic data $\mathbf{y}_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4})$ where $i = 1, \ldots, 1000$ generated with 25% missing rate by

$$\begin{aligned} Y_1 &= 1 + e_1, \\ Y_2 &= 2 + 0.5e_1 + 0.866e_2, \\ Y_3 &= Y_1 + e_3 \\ Y_4 &= -1 + 0.5Y_3 + e_4 \end{aligned}$$

where e_1, e_2, e_4 are randomly generated by normal distribution, and e_3 by gamma distribution. The 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.

TABLE 3: 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
FHDI	0.0383	0.0372	0.0451	0.0453
P-FEFI	0.0367	0.0378	0.0460	0.0457
P-FHDI	0.0384	0.0370	0.0449	0.0450

TABLE 4: Standard errors of naive and 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 data $\mathbf{U}(1000, 4, 0.25)$, 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 3 presents that P-FEFI produces the same stand errors of $\boldsymbol{y}$ with serial FEFI. The results generated by P-FHDI slightly differ from the results of serial FHDI in a tolerable manner. The random number generator for the selection of donors in FHDI and P-FHDI is library-dependent, in which is the main reason leading to the tiny residuals. Note that the stand errors of P-FHDI decrease asymptotically as n increases gradually in Table 4. As expected, both 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 4) 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 P-FHDI in Appendix A. All the developed codes are shared with GPL-2, and codes, examples, and documents are available in [29]. Practical example data sets are obtained from UCI machine learning repository [30].

5 Cost analysis and scalability

It is crucial to perform computational complexity to the P-FHDI algorithm. It will not only evaluate the performance

of P-FHDI but also reveal the potential memory risks. Note that when it comes to the Big O notation of 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 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 is consist of computational cost $\mathcal{C}$ and communication cost $\mathcal{H}$, which reads:

$$T(Q) \approx \frac{\alpha^{'}}{Q} + \beta^{'} Q \tag{39}$$

where $\alpha^{'} = \alpha\psi_1(n,p)O(n^2) + \alpha O(n^3)$ and $\beta' = \beta\psi_1(n,p)O(n)$. $\psi_1(n,p)$ is the terminational 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 the equation for $T(Q)$, we have the scalability of $\frac{T(c_pQ)}{T(Q)}$ by

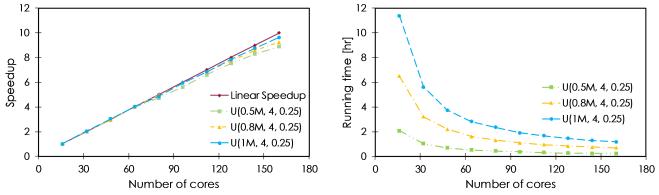
$$\frac{T(Q)}{T(c_pQ)} = c_p \times \frac{\alpha^{'} + Q^2\beta^{'}}{\alpha^{'} + c_p^2Q^2\beta^{'}} \tag{40}$$

where $c_p \in \mathbb{N}^+$. Detailed derivations of Eqs. (39) and (40) are presented in Appendix B. To evaluate the performance of P-FHDI especially with a large number of instances in practical, 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 the synthetic data sets in 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 data sets will be generated by varying the target parameters in Table 5.

TABLE 5: Data sets $\mathbf{U}$(instances, variables, missing rate) prepared for parametric studies.

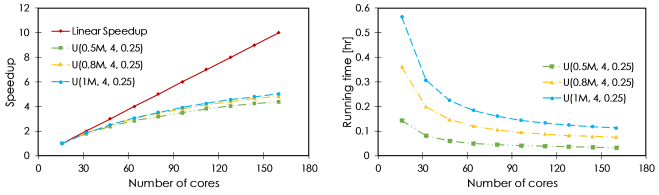
Parameter	Data set 1	Data set 2	Data set 3
n	$\mathbf{U}(0.5M, 4, 0.25)$	$\mathbf{U}(0.8M, 4, 0.25)$	$\mathbf{U}(1M, 4, 0.25)$
η	$\mathbf{U}(1M, 4, 0.15)$	$\mathbf{U}(1M, 4, 0.25)$	$\mathbf{U}(1M, 4, 0.35)$
p	$\mathbf{U}(15K, 8, 0.15)$	$\mathbf{U}(15K, 12, 0.15)$	$\mathbf{U}(15K, 16, 0.15)$

Note that 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. For a convenient description of results, the cell construction, estimation of cell probability and imputation are compressed as a single part named after the imputation in this section. Overall, larger n increases the running time of the imputation and variance estimation, resulting in the increments of total running time from Fig. 5 to 7. But n is not likely to significantly affect the speedup of imputation and variance estimation. The larger η affects the running time of the imputation positively rather than variance estimation, leading to slightly increments of total running time from Fig. 8 to 10. And η apparently influences the speedup of imputation only. In general, the number of instances n is the dominating parameter positively affecting speedup and running time. All of these parametric studies have a good agreement with the cost model of speedup and running time.



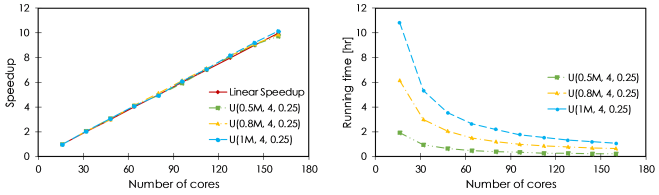
(a) Speedup (b) Running time

Fig. 5: Impact of the number of instances n on speed up and 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.



(a) Speedup (b) Running time

Fig. 6: Impact of the number of instances n on speedup and running time of the imputation only with datasets $\mathbf{U}(n, 4, 0.25)$: 4 variables and 25% missing rate by varying n.

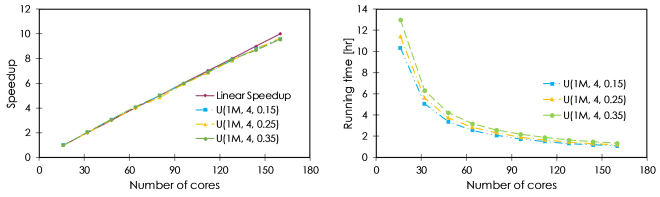


(a) Speedup (b) Running time

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

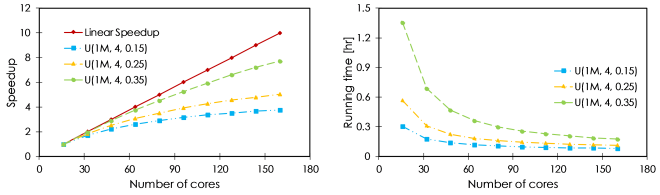
6 Variable Reduction for Big-p Data Set

The current algorithm of P-FHDI maybe not adequate for imputing big-p data with too many variables (e.g., $p > 100$). We performed parametric studies with increasing p and results show weak speedups as shown in Fig. 11 (a). 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



(a) Speedup (b) Running time

Fig. 8: Impact of the missing rate η on speedup and 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.



(a) Speedup (b) Running time

Fig. 9: Impact of the missing rate η on speedup and running time of the imputation only with datasets $\mathbf{U}(1M, 4, \eta)$: 1 million instances and 4 variables by varying η.

to guarantee at least two donors during the current cell construction algorithm. This issue is theoretically related to the so-called curse of high dimensionality. There is a huge literature on the problem of variable selection. To name a few, the least absolute shrinkage selection operator (LASSO) in [31], ridge in [32], principle component analysis (PCA) in [33] and smoothly clipped absolute deviation (SCAD) method in [34]. The sure independence screening (SIS) proposed in [35] is popular for ultrahigh variable reduction. It filters out the variables that have weak correlations with the response based on correlation learning. Consider a linear regression model

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{e} \tag{41}$$

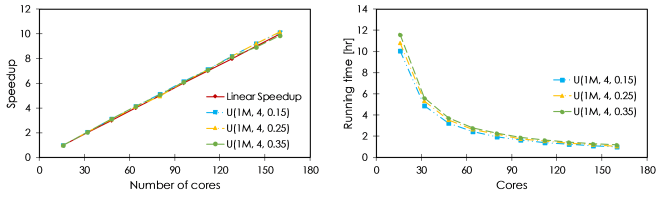
where $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)^T$ is vector of responses, $\mathbf{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_p)$ is an $n \times p$ random design matrix with independent and identically distributed (IID) elements. $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^T$ 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

$$M_\gamma = \{1 \leqslant i \leqslant p; \quad |r_i| \text{ is among the top of largest ones}\} \tag{42}$$

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.



(a) Speedup (b) Running time

Fig. 10: Impact of the missing rate η on speedup and running time of the variance estimation only with datasets $\mathbf{U}(1M, 4, \eta)$: 1 million instances and 4 variables by varying η

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

$$P(M_* \subset M_\gamma) \to 1 \tag{43}$$

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 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). Sequentially, 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 $\boldsymbol{r}_k = (r_1, r_2, \ldots, r_q)$ 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 in descending order.

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

$$M_k = \{1 \leqslant i \leqslant p; \quad |r_i| \text{ is among the top of largest } v, \text{where } r_i \in \boldsymbol{r}_k\}. \tag{44}$$

(3) We are implicitly assuming that

$$P(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_w \mid \boldsymbol{X}_1, \ldots, \boldsymbol{X}_q) = P(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_w \mid \boldsymbol{X}^*) \tag{45}$$

where $\boldsymbol{X}^*$ is the covariates corresponding to $M = \cap_{k=1}^{w} M_k$.

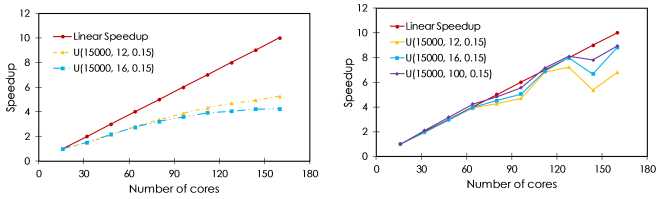
(4) If number of selected covariates in M equals v, then stop. Otherwise, repeat step (2) and (3) by setting $v = v + 1$ until we obtain v selected variables.

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

$$P(M_* \subset M) \to 1 \tag{46}$$

as $n \to \infty$.

Particularly, the temporary storage of $\hat{\mathbf{p}}(\boldsymbol{z}^*) \in \tilde{n} \times \tilde{n}_R$ in line 1 of Algorithm 9 most likely needs excessive memory for big-p data. The maximum value of $\tilde{n}$ can be 4^p if given three categories, resulting in $\tilde{n} \to n$ as p increases. Hence, we jointly embed the function Rep_CellP into the L-Replication process of variance estimation to avoid excessive storage of $\hat{\mathbf{p}}(\boldsymbol{z}^*)$. As a result of the implementation of the big-p algorithm in Fig. 11 (b), the weak speedups have been enhanced to the linear speedups. Note that the adoption of uniform job distribution results in the jagged speedup curves.



(a) Speedup (b) Speedup

Fig. 11: The impact of the number of variables p on speedups using (a) big-n algorithm and (b) 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), respectively in (b).

The parametric studies of the number of selected variables on the ratio of the means in Fig. 12 (a) shows that the adoption of reduced variables won't affect the ratio of means significantly. Let $\mathbf{U}_g$ be subsets of the finite population with a size of N_g, where $g = 1, \ldots, G$. From the Eq. (C.4) in the Appendix C, the ratio of the means is given as

$$\frac{\mathrm{E}(Y_N) + \mathrm{E}\left(\sum_{g=1}^{G_v} \sum_{i \in U_g} (R_g^{-1}\delta_i - 1)(y_i - \mu_g)\right)}{\mathrm{E}(Y_N)} \to 1 \quad (47)$$

as $v \to p$ where $R_g = N_{R_g}/N_g$, and G_v is the number of imputation groups using selected variables. The choice of reduced variables will slightly differ G_v, leading to additional bias in the second term of numerator. That is, the slight fluctuation in Fig. 12 (a) is explained. In future work, a theoretical number of v for a desired level of bias will be addressed.

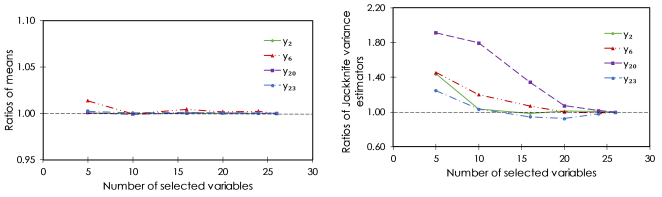
Now consider the behaviors of variance using big-p algorithm. Let $\mathcal{W}$ be

$$\mathcal{W} = \sum_{j \neq r}^{M} \left(\frac{w^*_{ij,FEFI}}{\sum_{j \neq r} w^*_{ij,FEFI}} y_i^{*(j)}\right) - y_i^{*(r)} \quad (48)$$

where r is the index of closest donor to k and $w^*_{ij,FEFI}$ is defined in Eq. (16). Considering the update of $w_{ij}^{*(k)}$ in Eq. (20), the ratios of $\hat{V}(\hat{Y}_{FHDI})$ using v selected variables to that using all observed variables is presented in Eq. (C.10) in Appendix C as

$$\frac{\sum_{k=1}^{n} \left(\sum_{i \in A_M} w_i w^*_{ir,FEFI} \mathcal{W}_v\right)^2}{\sum_{k=1}^{n} \left(\sum_{i \in A_M} w_i w^*_{ir,FEFI} \mathcal{W}_p\right)^2} \geqslant 1 \quad (49)$$

where $\mathcal{W}_v$ and $\mathcal{W}_p$ are built upon v selected variables and all observed variables, respectively. The ratio will approach 1 if $v \to p$. It is proved that $\hat{Y}_{FHDI}$ has additional variance due to the donor selection procedure in [15]. The prototype of big-p algorithm is to utilize only reduced variables for each recipient to select donors, resulting in higher variance. The $\mathcal{W}$ measures how far $y_i^{*(r)}$ is away from the mean of other donors. Hence, a higher variance of selected donors will significantly affect $\mathcal{W}_v$, as a result of the increment of the numerator of Eq. (49). On the contrary, the adoption of all variables minimizes the variance in selecting donors, leading to $\mathcal{W}_p$ in denominator minimized. Therefore, the model M built upon a lesser number of selected variables will have higher variance as shown in Fig. 12 (b). And there will be a theoretical choice of v that can reasonably minimize the ratio defined in Eq. (49), which will be addressed in future work.



(a) The ratios of the mean (b) The ratios of the Jackknife variances

Fig. 12: The ratios of mean and Jackknife variance estimator of the practical data set $\mathbf{U}(19735, 26, 0.15)$ using different number of selected variables

7 Future Research

In the future work, a theoretical choice of the number of selected variables v in the big-p algorithm which minimizes the Jackknife estimate of variance and stabilizes the mean will be addressed. Also, the present big-p algorithm of P-FHDI is not adequate for the categorical variables. The categorical variables violate assumptions for computing correlation, of which all variables should be continuous. One set of possible solutions to find associations between categorical variables rely on distance metrics. Other possible proposals span various statistical metrics (e.g., chi-squared statistics). The marginal association measurement proposed in [36] 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 and computational advancements shall be focusing on the ultra data sets 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 developed and shared the first version of the parallel fractional hot deck imputation program, named as P-FHDI. We document the

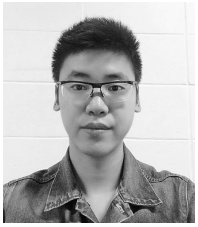
full details regarding the algorithm-oriented parallelisms, computational implementations, and various examples of P-FHDI to benefit a broad audience in the science and engineering domain. The developed P-FHDI is suitable to tackle large-instance (so-called big-n) and large-variable (so-called big-p) missing data with irregular, complex missing patterns. The validations and analytical cost models confirm that P-FHDI exhibits fairly good scalability for big incomplete data set regardless of various missing rates. This program P-FHDI will help to impute incomplete data, enable parallel variance estimation, and ultimately improve the subsequent statistical inference and machine learning with the cured big data sets. The next version of P-FHDI will focus on ultra data sets 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.

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-frck 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, 2015.
[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] IBM, 2012. [Online]. Available: http:// www-01.ibm.com/software/data/bigdata/
[19] 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.
[20] 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.
[21] 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.
[22] 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.
[23] X. Hu, "Acceleration genotype imputation for large dataset on gpu," *Procedia Environmental Science*, vol. 8, pp. 457–463, 2011.
[24] 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.
[25] J. G. Ibrahim, "Incomplete data in generalized linear models," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 765–769, 1990.
[26] 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.
[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Eds., *Introduction to Algorithms*. London: MIT press, 2009.
[28] 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
[29] 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
[30] D. Dheeru and G. Casey, "Uci machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[31] 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.
[32] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
[33] L. Jolliffe, Ed., *Principle component analysis*. Berlin Heidelberg: Springer, 2011.
[34] 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.
[35] 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.
[36] 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 under the supervision of professor Wayne A. Fuller in the summer of 2000. He is currently a professor of 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.