Journal of Genetics (2022)101:52
https://doi.org/10.1007/s12041-022-01396-5

RESEARCH ARTICLE

Accelerating imputation of missing genotypes using parallel computing

FARHAD GHAFOURI-KESBI*

Department of Animal Science, Faculty of Agriculture, Bu-Ali Sina University, Hamedan 6517838695, Iran
*E-mail: farhad_ghy@yahoo.com, f.ghafouri@basu.ac.ir.

Received 3 February 2022; revised 8 August 2022; accepted 22 August 2022

Abstract. Owing to massive jump in DNA technology, large-scale genomic datasets, including valuable information, have become available. While this is a prodigious opportunity, and it can also be a big challenge because analysing these large datasets with current computers and software tools is very difficult and may take days or even weeks to complete. Novel approaches such as parallel computing have been suggested to deal with these large datasets. Here, the effect of parallel computing on the performance of random forest (RF) algorithm for imputation of missing genotypes was studied. To this end, the genotypic matrices were simulated for, respectively, 500, 1000, 2000, and 3000 single-nucleotide polymorphism (SNP) for 500, 1000 and 2000 individuals, respectively. Then, 50% of genotypic information was masked and imputed by RF. The per cent of genotypes correctly imputed was used to measure accuracy of genotype imputation. Serial and parallel computing were applied to the data. In comparison to serial computing, parallel computing did not affect the accuracy of imputation, and the accuracy was the same in both scenarios. However, regarding computational time, parallel computing accelerated the analyses significantly in a way that it reduced the running time up to 63%. This was due to the fact that in the serial computing, only 10% of the processing power of the central processing unit (CPU) of the machine was used by the RF, while in the parallel computing, 55% of the processing power of the CPU was utilized. Therefore, as parallel computing significantly reduced the computing time and does not affect the accuracy of the results, this approach should be exploited by researchers to analyse large genomic datasets.

Keywords. serial computing; parallel computing; genotype imputation; random forest.

Introduction

Genomic information from DNA sequencing technologies are valuable resources in different branches of science such as medicine and epidemiology, evolution, and animal and plant breeding. Genomics, the main branch of bioinformatics, is used to study genomic information. From 2007 onward, the advent of next-generation sequencing (NGS) technologies such as whole-genome sequencing (WGS) and whole-exome sequencing (WES) have increased the speed of DNA sequencing significantly. Thanks to NGS technologies, today, the complete genome of an organism can be sequenced in a few hours. However, the output of these new technologies, which includes the sequencing of millions of DNA molecules, is very complicated and bulky. For example, the output of sequencing of a whole individual genome by NGS protocols produce about 100 gigabytes of data (He *et al.* 2017). These genomic data with such a large volume are termed 'Big Data' and challenge conventional data processing methods. Even for the efficient algorithms such as Bayesian methods, analysing such large volume of data may take days or weeks (Guo *et al.* 2018). Under such a situation, parallel computing can be an efficient strategy (figure 1). Traditionally, computer software has been written for serial computing. It means that (i) first, a problem is divided into a serial stream of instructions; (ii) instructions are sent serially to the central processing unit (CPU) and executed sequentially one after the other on a single processor (core). Therefore, one instruction may execute at any moment of time. In parallel computing, a different fashion is followed: (i) first, a problem is divided into several instructions, (ii) these instructions are executed simultaneously on the different processors, and (iii) results of several processors are combined to make a general conclusion. Therefore, in parallel computing, more

Published online: 02 November 2022

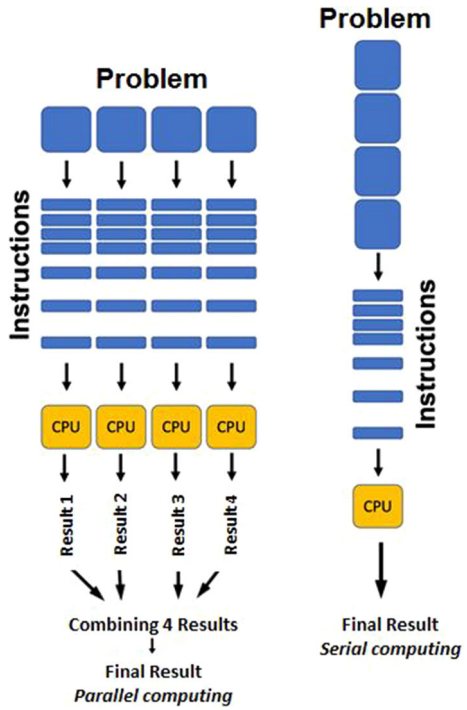

Figure 1. Parallel vs serial computing (https://pythonnumerical-methods.berkeley.edu/notebooks/chapter13.01-Parallel-Computing-Basics.html).

than one instruction is executed at any moment of time (Trobec *et al.* 2018). This has become feasible using hyper treading (HT) technology of Intel Company (Intel® Hyper-Threading Technology 2003) which is the base for parallel computing.

Random forest (RF, Breiman 2001) is one of the machine learning algorithms for classification and regression problems. The RF grows hundreds or thousands of decision trees on training data, and after training, the trained model is applied to test data. Owing to its high accuracy, RF has been widely used for analysing genomic data (Ogutu *et al.* 2011; Stephan *et al.* 2015; Ghafouri-Kesbi *et al.* 2017). Using the RF framework, Stekhoven (2016) developed *missForest* package to impute missing values. Rutkoski *et al.* (2013) used *missForest* package to impute missing genotypes and reported that while RF ranked as one of the most accurate algorithms, it was very slow in terms of computing speed. Other researchers, including Neves *et al.* (2012) and Ashoori-Banaei *et al.* (2021), who used RF for genomic selection, reported that although RF was highly accurate, it needed a long time to complete the analyses compared to other methods. Therefore, the preliminary hypothesis was that parallelization should speed up the RF. The effect of parallelization on the speed of RF algorithm for imputing missing genotypes has not been studied so far. Hence, this study focussed on RF to determine the extent to which parallelization can increase its speed for imputing missing genotypes.

Material and methods

Simulation of data

Simulation of genomic matrices was done using package *hypred* (Technow 2013) in R software (R Development Core 2021). A chromosome was simulated on which, in different scenarios, respectively, 500, 1000, 2000, and 3000 single-nucleotide polymorphism (SNP) with an initial frequency of 0.5 was distributed. To each SNP with genotype AA code 2, with genotype Aa code 1, and with genotype aa code 0 were assigned. Genotypic matrices were constructed for 500, 1000, and 2000 individuals, respectively. For making data files with missing genotypes, in all scenarios, the 50% of the genotypes from genotype matrices were masked.

Genotype imputation

Random forest regression (Breiman 2001) was used to genotype imputation. In a genotype matrix including missing genotypes, all available data were used to predict the missing values for every marker. The R package *missForest* (Stekhoven 2016) was used for genotype imputation. The imputation was carried out as follows: (i) for genotype matrix **M**, genotypes were ranked from lowest to highest per cent missing, and missing genotypes were initially imputed by the mean neighbour imputation (MNI) algorithm. (ii) At each marker j that had missing genotypes, the nonmissing genotypes (Y) were used to grow 100 random forest regression trees ($\mathbf{\Psi_1}\ldots\mathbf{\Psi_{100}}$). A bootstrapped sample of individuals Y and a random sample of $\sqrt{n-1}$ marker predictors is used to grow each tree (n-1 is the number of markers without marker j). (iii) Missing genotypes at marker j are imputed as follows:

$$\hat{Y}_{mis} = \frac{1}{100}\sum_{1}^{100} h(\mathbf{x}, \Psi).$$

(iv) Matrix **M** is then updated using the predicted values ($\widehat{Y}$). (v) Steps ii–iv are repeated for all markers, including missing genotypes, until all missing genotypes are imputed (Rutkoski *et al.* 2013). The accuracy of genotype imputation was assessed as the percentage of genotypes imputed correctly:

$$\%\text{Correct} = \frac{\text{Number of missing genotypes imputed correctly}}{\text{Total number of missing genotypes}}.$$

Parallel computing

To parallel RF, the package *doParallel* (Calaway *et al.* 2018) was used. Analyses were done with a PC equipped

with the Intel Core i7-6800K CPU and 16 gigabytes of RAM. Two scenarios, (i) serial computing and (ii) parallel computing were used to analyse data. In serial computing, data was executed on one core of the CPU, i.e., of the six cores of the CPU, only one core was involved in the analysis. In parallel computing, in different scenarios, data was executed on 2, 3, 4, 5 and 6 cores, respectively. Computing time in each scenario was recorded with an R function. Different scenarios of the number of cores and size of the genotypic matrix were analysed 10 times, and the mean of 10 runs was reported. The efficiency of parallel computing was assessed by *speedup ratio*, which was calculated as the ratio of the serial runtime for analysing the data to the time taken by the parallel scenario to analyse the same data on n processors.

Results and discussion

Figure 2, as the output of serial computing, shows that how an increase in the volume of data increased computing time. When the dimension of the genotypic matrix was 500 × 500 (500 individuals genotyped for 500 markers), the computing time was 5.19 min. With the increase in the dimension of the genotypic matrix to 2000 × 3000 (2000 individuals genotyped for 3000 markers), the computing time increased to 636.6 min. Factors such as the size of the data, the algorithm used, and the computing power of the machine affect the computing time. Dealing with big data, even if the algorithm is fast, using serial computing, the analyses may take several days or weeks (Guo *et al.* 2018). In addition, figure 2 also shows that there is no linear relationship between the size of the data and computing time because by doubling the data size, the serial runtime was tripled. Although the latter result is limited to the current study; however, it gives a view regarding the effect of data size on the computing time.

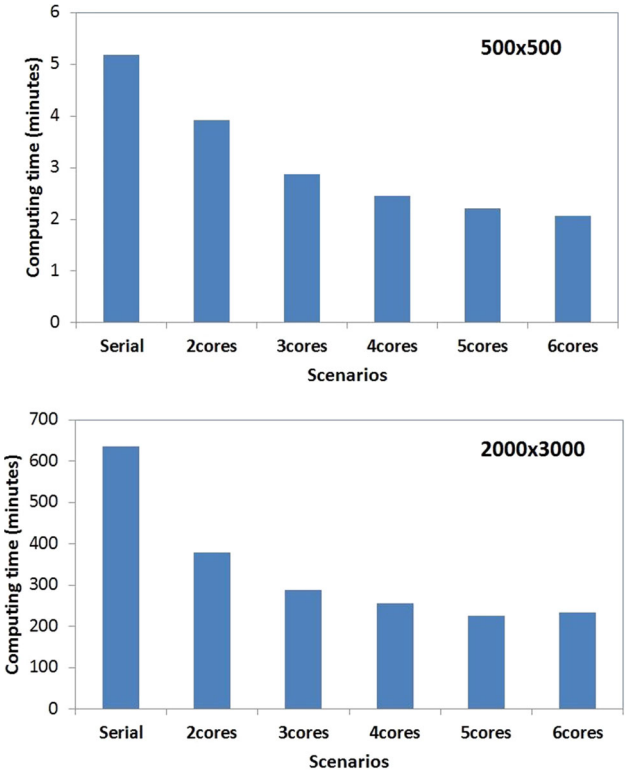

Figure 3. The effect of parallel computing on the computing time for smallest (500 × 500) and biggest (2000 × 3000) datasets.

Figure 3 shows the effect of parallel computing on the computing time for smallest and biggest datasets. Regarding the smallest genotypic matrix (500 × 500), serial runtime was 5.19 min. With parallel computing exploiting all six cores of the CPU, the runtime decreased significantly by 60% and reached 2.07 min. The speedup ratio was 2.51.

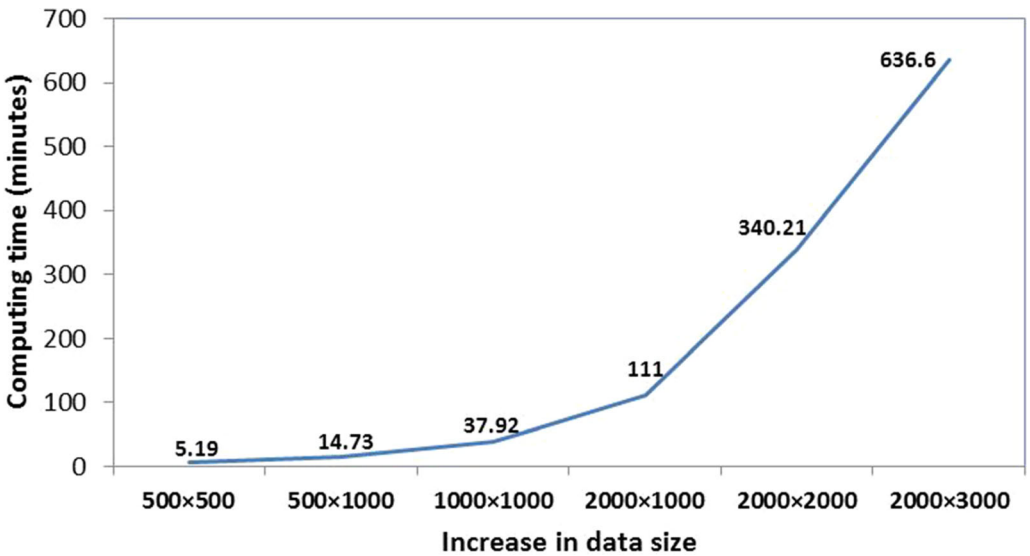

Figure 2. The effect of data size on the serial computing time.

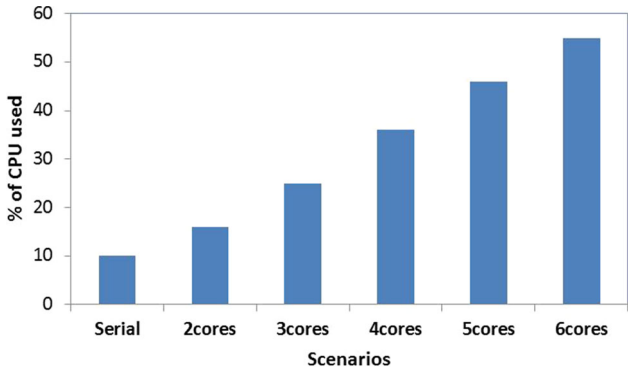

Figure 4. Per cent of the processing power of CPU used in different scenarios of parallel computing.

Considering the genotypic matrix with the largest dimension (2000 × 3000), the serial computing completed after 636.6 min, while parallel computing, executed on six cores, lasted 234 min. In other words, with a speedup ratio of 2.71, the computing time decreased by 63%. One of the earlier studies in the context of parallel computing of genomic data was done by Calborg *et al.* (2001). They applied serial and parallel computing for detecting quantitative trait loci (QTL) by interval mapping. The serial runtime was 13.7 h and decreased to 5.8 h, 3.6 h, 2.6 h, and 2.1 h when parallel computing using 3, 6, 9, and 18 cores was applied to the data. Agapito *et al.* (2016) worked on human genomic data and reported a linear relationship between the number of cores and the speed of analyses in such a way that by increasing the number of cores involved from one to six cores, the analyses were done six times faster. Wu *et al.* (2012) parallelized Bayesian methods by Hidden Markov Chain in genomic evaluation of rib Eye in a population of beef cattle. While serial computing lasted 1386 min, in parallel computing, runtime decreased by 7.78 times and reached 178 min. Guo *et al.* (2018) used Hidden Markov Chain to parallelize different Bayesian methods and reported that by increasing the number of chains from 1 to 18, the computating time decreased significantly. Their estimate of the speedup ratio was between three (BayesA) and 13 (BayesCπ).

Figure 4 shows how parallel computing decreases the computing time. As shown, in parallel computing, the processing power of the CPU was utilized much more efficiently than in serial computing. In serial computing, only 10% of the processing power of the CPU was used. In contrast, in parallel computing, by increasing the number of cores to 2, 3, 4, 5 and 6 cores, 16%, 25%, 36%, 46%, and 55% of the processing power of the CPU was exploited, respectively. It could reach 100% if the size of the memory was larger.

Figure 3 also shows that the trend of decrease in computing time following the increase in the number of cores involved is not linear. For example, in the scenario of the biggest data size (2000 × 3000), parallel computing with three cores decreased computing time by 24%, while by increasing the number of cores involved from three to four, the computing time decreased by 11%. A similar trend was also observed by increasing the number of cores from four to five and then to six cores in such a way that the difference between five and six cores was almost equal. Similar results were reported by Calborg *et al.* (2001), Wu *et al.* (2012), and Guo *et al.* (2018). The justification is that, in addition to the time taken for executing the instructions on different cores, the step of combining the results is also time-consuming. When six cores are involved, six results are produced. But when five cores are involved, there would be five results that should be combined to produce a conclusion. In other words, the longer combining time decreases the efficiency of parallel computing while moving from five cores to six cores. It can explain why the difference between five and six cores scenarios is not significant. Rastogi and Zaheer (2018) reported that in parallel computing, the computing time did not follow a linear trend by increasing the number of cores involved because there are factors like synchronization, load balancing, etc., involved, so there is a decrease in speedup after a point in reality.

Figure 5. Accuracy of genotype imputation in serial and parallel computing for smallest (500 × 500) and biggest (2000 × 3000) datasets.

As observed in figure 5, the imputation accuracies of different scenarios were approximately equal. For the biggest genotypic matrix, the imputation accuracy ranged between 84% and 86%, and for the smallest genotypic matrix, the imputation accuracy was in the range between 70% and 73%. In other words, compared to serial computing, parallelization decreased the computing time without affecting accuracy. Guo *et al.* (2018) reported that parallel computing with Bayesian methods did not affect the accuracy of genomic evaluation of Simental cow for carcass traits. Figure 5 also shows that the accuracy of imputation was higher when the size of genotypic matrix increased. This is because at a fixed amount of missing genotypes, in a larger matrix, a larger percentage of markers with known genotypes will be available to the algorithm for imputing missing genotypes.

In conclusion, the results of this research showed significant advantages of parallel computing over serial computing regarding computing time. In addition, serial and parallel computing imputed missing genotypes with the same accuracy. Therefore, when real-time results are needed, parallel computing could be an excellent strategy to deal with large datasets.

References

Agapito G., Guzzi P. H. and Cannataro M. 2016 Parallel processing of genomics data. *AIP. Conf. Proc.* **1776**, 7–8.

Ashoori-Banaei S., Ghafouri-Kesbi F. and Ahmadi A. 2021 Comparison of regression tree-based methods in genomic selection. *J. Genet.* **100**, 85.

Breiman L. 2001 Random forests. *Machine Learning* **45**, 5–32.

Calaway R., Weston S. and Tenenbaum D. 2018 doParallel: Foreach parallel adaptor for the 'parallel' package (available at: https://cran.project.org/web/packages/doParallel/index.html).

Carlborg Ö., Andersson-Eklund L. and Andersson L. 2001 Parallel computing in interval mapping of quantitative trait loci. *J. Hered.* **92**, 449–451.

Ghafouri-Kesbi F., Rahimi-Mianji G., Honarvar M. and Nejati-Javaremi A. 2017 Predictive ability of random forests, boosting, support vector machines and genomic best linear unbiased prediction in different scenarios of genomic evaluation. *Anim. Prod. Sci.* **57**, 229–236.

Guo P., Zhu B., Niu H., Wang Z., Liang Y., Chen Y. *et al*. 2018 Fast genomic prediction of breeding values using parallel markov chain monte carlo with convergence diagnosis. *BMC Bioinf.* **19**, 3.

He K., Ge D. and He M. 2017 Big data analytics for genomic medicine. *Int. J. Mol. Sci.* **18**, 1–18.

Intel Hyper-Threading Technology 2003 *Technical user's guide* (available at: http://www.cslab.ece.ntua.gr/courses/advcomparch/2007/material/readings/Intel%20Hyper-Threading%20Technology.pdf).

Neves H. H. R., Carvalheiro R. and Queiroz S. A. 2012 A comparison of statistical methods for genomic selection in a mice population. *BMC Genetics* **13**, 100.

Ogutu J. O., Piepho H. P. and Schulz-Streeck T. 2011 A comparison of random forests, boosting and support vector machines for genomic selection. *BMC Proc.* **5**, 11.

R Development Core Team 2021 R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Rastogi S. and Zaheer H. 2018 Significance of parallel computation over serial computation using OpenMP, MPI, and CUDA. Quality, IT and Business Operations, Springer Proceedings in Business and Economics. pp 359–367.

Rutkoski J. E., Poland J., Jannink J. L. and Sorrells M. E. 2013 Imputation of unordered markers and the impact on genomic selection accuracy. *Gen. Genom. Genet.* **3**, 427–439.

Stekhoven D. 2016 Nonparametric missing value imputation using random forest (available at: https://cran.r-project.org/web/packages/missForest/missForest.pdf).

Stephan J., Stegle O. and Beyer A. 2015 A random forest approach to capture genetic effects in the presence of population structure. *Nat. Com.* **6**, 7432.

Technow F. 2013 hypred: Simulation of genomic data in applied genetics (available at : https://www.uni-hohenheim.de/fileadmin/einrichtungen/plant-breeding/pdfs/hypred_vignettes.pdf).

Trobec R., Slivnik B., Bulić P. and Robič B. 2018 Introduction to parallel computing: from algorithms to programming on state-of-the-art platforms, First edition, pp. 268. Springer publishing.

Wu X. L., Sun C., Beissinger T. M., Rosa G. J., Weigel K. A., Gatti N. *et al*. 2012 Parallel markov chain monte carlo-bridging the gap to high-performance bayesian computation in animal breeding and genetics. *Genet. Sel. Evol.* **44**, 29.

Corresponding editor: Durgadas P. Kasbekar