Accelerating SARS-CoV-2 low frequency variant calling on ultra deep sequencing datasets

Bryce Kille
Department of Computer Science
Rice University, Houston, Texas
blk6@rice.edu

Yunxi Liu
Department of Computer Science
Rice University, Houston, Texas
yl181@rice.edu

Nicolae Sapoval
Department of Computer Science
Rice University, Houston, Texas
ns58@rice.edu

Michael Nute
Department of Computer Science
Rice University, Houston, Texas
mn56@rice.edu

Lawrence Rauchwerger
Department of Computer Science
University of Illinois, Urbana, Illinois
rwerger@illinois.edu

Nancy Amato
Department of Computer Science
University of Illinois, Urbana, Illinois
namato@illinois.edu

Todd J. Treangen
Department of Computer Science
Rice University, Houston, Texas
treangen@rice.edu

***Abstract*—With recent advances in sequencing technology it has become affordable and practical to sequence genomes to very high depth-of-coverage, allowing researchers to discover low-frequency variants in the genome. However, due to the errors in sequencing it is an active area of research to develop algorithms that can separate noise from the true variants. LoFreq is a state of the art algorithm for low-frequency variant detection but has a relatively long runtime compared to other tools. In addition to this, the interface for running in parallel could be simplified, allowing for multithreading as well as distributing jobs to a cluster. In this work we describe some specific contributions to LoFreq that remedy these issues.**

I. Introduction

Cataloging viral mutations within a sample (intra-host variation) and across samples (inter-host variation) provides critical insights to understanding the dynamics of viral evolution during the COVID-19 pandemic [1]. Single nucleotide variants (SNVs) can result in drastically different protein function and recognition; it is therefore desirable to be able to accurately and efficiently identify SNVs. For example, the SARS-CoV-2 virus was recently shown to have significant underlying diversity [2], [3]; additionally the mutations can change the fitness of the virus [4] increasing it's transmission or pathogenicity potential [5].

Recent high throughput sequencing techniques enable researchers to generate read sets with deep coverage [6], allowing researchers to discover low frequency variants in the population. However, sequencing errors can masquerade as a low-frequency variant, and vice-versa, so distinguishing between the two is necessary [7]. A recent benchmarking study [8] found that on simulated HiSeq data, LoFreq [9] outperformed most variant calling tools, but suffered from long execution times. LoFreq operates by considering, at each position in the genome, the probability that all mismatches to the reference genome are caused by sequencing error, although the specific probability calculation involved is computationally expensive and thus it does not scale well for deep sequencing datasets.

In this work, we describe an improvement to the runtime of LoFreq by using an approximation to shortcut costly dynamic-programming operations when exact values of the probability distribution are not needed. We also add experimental support for OpenMP [10] to enable more efficient parallel operation.

Funded by the C3.ai Digital Transformation Institute COVID-19 award. MN was also funded by a fellowship from the National Library of Medicine Training Program in Biomedical Informatics and Data Science (T15LM007093, PI: Kavraki).

II. Methods

Our approach to improve LoFreq is twofold. First, we implement an approximation heuristic which allows LoFreq to bypass computationally intensive exact probability calculations when not necessary. Second, we reorganize LoFreq to use OpenMP in lieu of a job-submission script for parallel operation.

A. Approximation

As high-throughput sequencers read a DNA strand every position is assigned a quality score corresponding to the probability that the assigned nucleotide is incorrect. Thus, for a given pileup column (i.e. a position in the reference genome and the set of nucleotides at the corresponding position in each read that has been mapped to that location), the total number of sequencing errors is the sum of independent but *not* identically distributed Bernoulli trials, also known as the Poisson binomial distribution.

More specifically, consider a single column with read depth d: let each read i have a probability p_i of having a sequencing error in this column (implied by the quality score), then the total error

978-1-6654-3577-2/21/$31.00 ©2021 IEEE
DOI 10.1109/IPDPSW52791.2021.00038

count is distributed as a Poisson Binomial with probabilities $\{p_i\}_{i=1}^{d}$. Although simple to parameterize, this distribution is computationally non-trivial, particularly for values of the cumulative distribution function (CDF) [11], [12]. Computing the probability of having *at least* K sequencing errors by chance requires the following recurrence relation:

$$P_n(X{=}k) = P_{n-1}(X{=}k)(1{-}p_n) + P_{n-1}(X{=}k{-}1)p_n$$

where $P_n(X{=}k)$ is the probability of observing k errors in the first n bases at the given position. Now, if K reads in this column contain bases that differ from the reference genome, we can use the Poisson binomial CDF to calculate a p-value for the null hypothesis that the variants are due only to sequencing error: $p = \Sigma_{k \geq K} P_d(X{=}k)$. LoFreq operates in exactly this way, calling a variant in a given column if the p-value falls below a specified critical value (Figure 1b).

Unfortunately, this method of calculating the Poisson binomial takes $O(d^2)$ time; more recent algorithms may improve on this but remain complex [11], [12]. But for the vast majority of columns the p-value is far away from the decision threshold because either so few or so many variants are present, so bypassing the CDF calculation for these cases is a potential speedup.

We do this by using an $O(d)$ approximation to the Poisson binomial as a first pass filter: if the approximate p-value, $\hat{p}$, is sufficiently far from the critical value the exact CDF computation is skipped and no variant is called, otherwise the standard calculation is used (Figure 1b). We have used the Poisson approximation where the mean is given by $\lambda = \sum_{i=1}^{d} p_i$ [13], specifically using the GNU scientific library implementation [14]. For our experiments, the significance threshold was left at the LoFreq default $\epsilon = 0.05$ and the first pass filter required that $\hat{p} \geq \epsilon + 0.01$ in order to skip the expensive exact calculation.

B. Parallelization

The LoFreq algorithm operates by iterating through each pileup column checking for SNVs. The current parallel implementation uses an external script to parse the input files and and partition the columns equally and subsequently spawning an independent LoFreq process for partition. In an experimental branch of LoFreq, we implemented the same strategy using OpenMP rather than separate processes. This OpenMP version relies on a parallel `for` loop across chunks of columns, using an independent `.bam` file reader for each thread.

III. Results

A. Performance Impact of the Poisson Approximation

Experiments were ran using BAM files ranging from 1MB to 25GB generated from the FastQ data available in [15]. Both versions of LoFreq, the original and our improved version without the OpenMP functionality, were ran on an Intel Xeon Gold 6138 CPU with 64 threads for the benchmarks. Table I shows the speedup observed with our approximation shortcut. The number of variants called by each version of LoFreq were identical for all five of the test data sets.

TABLE I: Execution times of the original and improved versions of LoFreq. In all cases the number of variants called was identical between versions. Note that while the true depth of the 25Gb file is likely around 5 million reads, LoFreq by default limits columns to 1 million.

Input size	Avg. depth	Run Time		Speed- up
		Orig.	New	
58M	1,000x	52 (s)	51 (s)	1.0x
237M	30,000x	58 (m)	26 (m)	2.6x
935M	100,000x	14 (h)	4 (h)	3.3x
2G	300,000x	55 (h)	12 (h)	4.6x
25G	1,000,000x	415 (h)	111 (h)	3.7x

B. Profiling the OpenMP Implementation

We utilized the HPC-toolkit [17] for visualizing the performance of the experimental OpenMP version of the improved LoFreq on a Knights Landing 2nd Generation Intel Xeon Phi processor with 128 threads. We observed in Figure 2, as expected, that time spent coordinating threads is minimal and the process is trivially parallel over the columns in the input. We also notice that the time spent iterating over the `.bam` file is substantial. While our goal with OpenMP was to reduce load imbalance via dynamic scheduling, we see that encountering partitions with high concentrations of variants near the end still results in a significant imbalance.

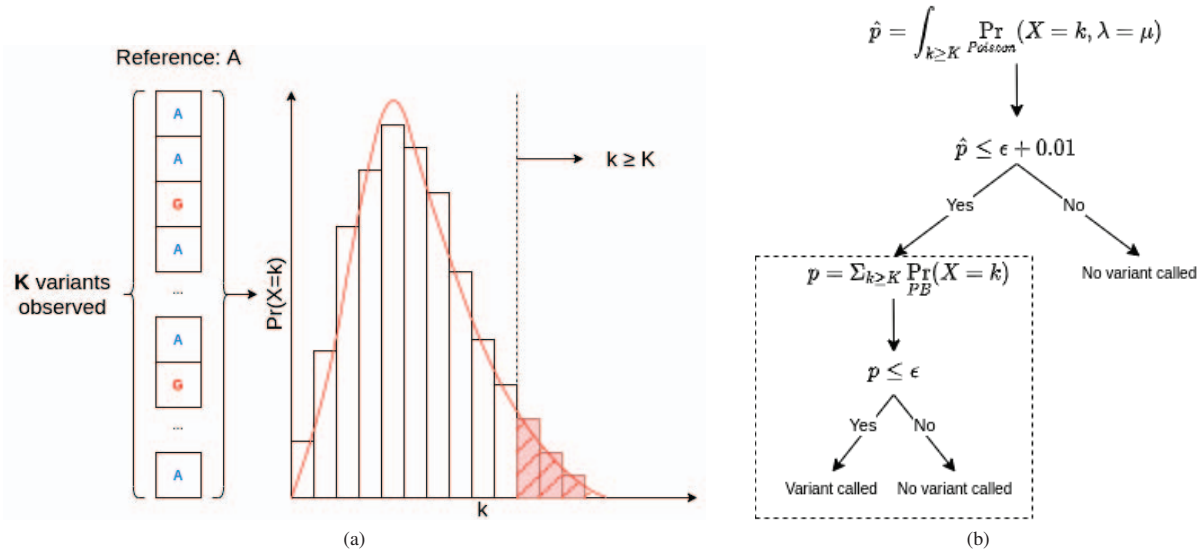
C. SARS-CoV-2 single nucleotide variant analysis

Finally, we performed an comparative analysis of SNVs identified across our COVID-19 datasets. We found from 134 (min) to 885 (max) SNVs in each dataset (Figure 3). The two highest depth-of-coverage datasets, 300,000X and 1,000,000X, shared the most variants for any pair. The 100,000X dataset had the most unique SNVs at 735 total. Only two SNVs were found to be shared across all five datasets.

IV. Discussion

In order to guarantee that the optimizations come at no cost to accuracy, we compared the output of the previous version of LoFreq to our improved version. Our improved version can only cause false negatives with respect to the original's variant calls, as we are using the approximation only to skip columns. Therefore, we only need to compare the count of variant calls in a sample across LoFreq versions. For all benchmarking datasets we observed the same number of variant calls in both versions of LoFreq.

Our results show that the approximation shortcut yields improved run-time over the original at no cost to accuracy in our samples. Since there is potential for our method to introduce false negatives into the results, we used an intentionally conservative threshold of 0.01 above the critical value. No experimentation or fine-tuning was done to optimize this parameter or examine the feasibility of the same approach for very low values of $\hat{p}$, so that is a possible avenue for additional performance improvement. One approach could be to have the threshold vary according to read depth because the accuracy of the Poisson approximation increases at higher



(a) (b)

Fig. 1: **The continuous Poisson approximation (red line) to the Poisson binomial (bars) distribution**. **(a):** The test statistic for the Poisson approximation is the right tail integral (shaded) and for the Poisson binomial it is the right tail sum (red bars). **(b):** The workflow of the improved LoFreq algorithm. The original LoFreq workflow is denoted by the dotted box. Here $\Pr_{PB}$ denotes probability under the Poisson-Binomial distribution. We first compute the tail integral over the Poisson distribution to get an approximate p-value we denote as $\hat{p}$. If $\hat{p} > \epsilon + 0.01$, then we have high confidence that $p > \epsilon$ and therefore we do not call a variant.

depth. As seen in the results, the approximation results in a 1-4x speedup in CPU time, particularly when run on large files with a low variant count. We also note that the approximation is more accurate when the error probabilities p_i are higher, so a specific version of the algorithm could be optimized for high-error, long read sequencing data.

We observe that for input data with low read-depth this heuristic is actually ill-suited, and can even be both less accurate and slower. For one, as noted above the error in the Poisson approximation vanishes asymptotically as d increases. Also, the existing version of LoFreq includes some conditions for early stopping that work especially well on shallow columns. To remedy this, our implementation only uses the approximation heuristic for columns with a read depth of at least 100. When read depth is below 100, the dynamic programming array used for the Poisson-Binomial fits inside the cache, which itself provides speedup comparable to the approximation. Importantly though, low-coverage sequencing input is inherently ill-suited to discovering variants present at low-frequency; for samples having depth below 100 throughout, a faster, less-sensitive SNV detector may be more appropriate.

The original LoFreq has not been optimized for cache performance, particularly on larger files. Since the computation of the Poisson binomial uses $O(d)$ memory, we quickly begin to spill over our shared cache when running in parallel files with depth $d > 1e5$. Our improved version of LoFreq has much better cache performance, with a cache miss rate below 15% compared to over 70% originally. Bypassing exact probability computations accounts for much of this as they repeatedly iterate over an array that does not fit in the cache. This also contributes to scalability in the new version since now, on average, only a small subset of running threads will need $O(d)$ memory at a given time.

We were successfully able to replicate the behavior of the parallelization script in OpenMP. While this improvement does not reduce aggregate CPU time in it's current form on the experiments we have run, it does offer other advantages. The OpenMP implementation addresses a minor bug mentioned in a variant caller review article [8] where the original implementation results in the output running through two stages of filtering when run in parallel: once for each individual process and then again on the combination of the variants from all of the processes. Unless set by the user, filter values are dynamically set during a LoFreq run, which causes the aforementioned filtering bug to produce inconsistent results. Our approach of using OpenMP to move all of the variant calling to the same process seems to remedy this problem.

Additionally, the parallelization script from the original LoFreq approach could still be used to partition the input for submission to a cluster, making it possible to parallelize across both shared and distributed memory environments. The OpenMP implementation also has the potential to avoid load imbalances that were possible previously by using smaller partitions towards the end of the run.

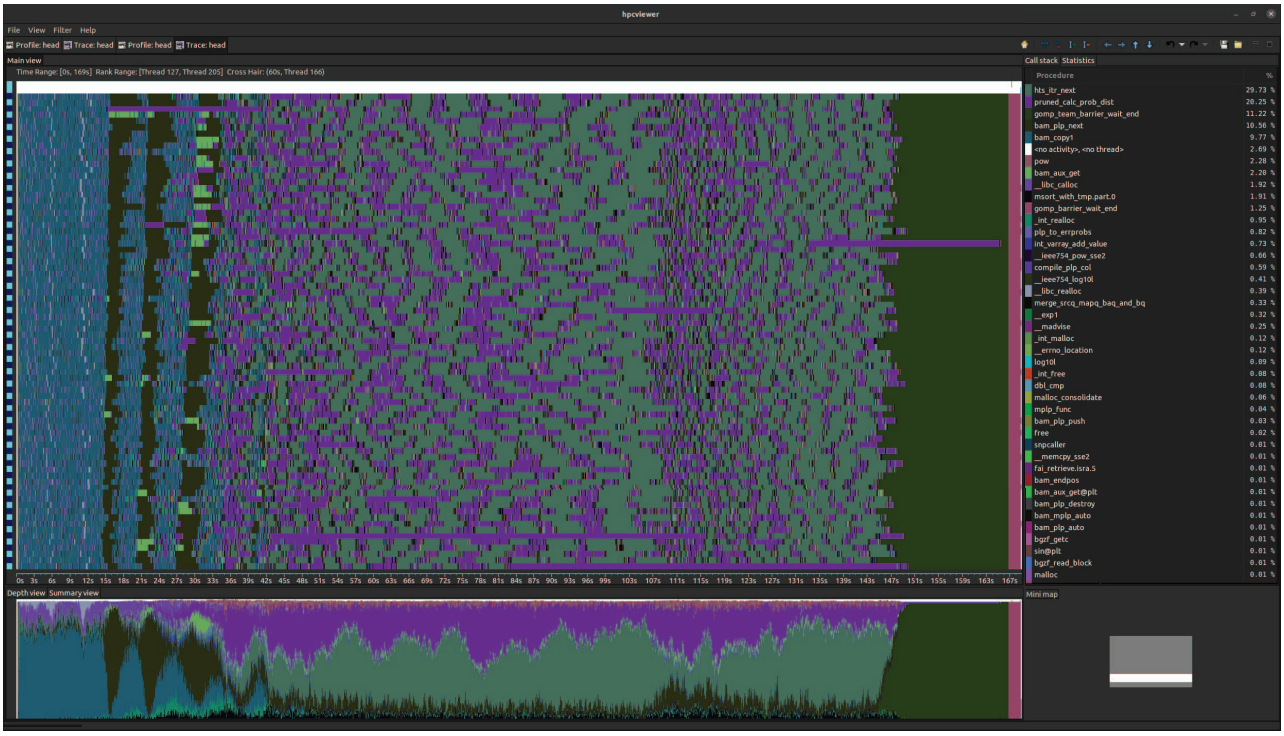


Fig. 2: **HPC-toolkit trace results.** X-axis is execution timeline and the Y-axis corresponds to the threads. The window at the bottom is the distribution of work across tasks. Pink is probability computation, teal is BAM file iteration, light blue at left is file decompression, dark green at right is the thread barrier. The image shows one thread causing a load imbalance due to a high-cost column.

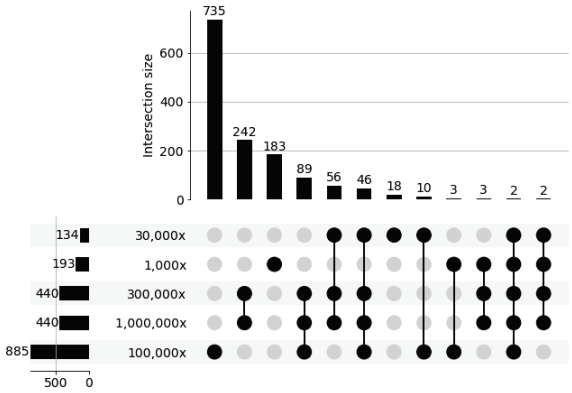


Fig. 3: **Upset plot [16] highlighting the shared low frequency variants across all five datasets.** Rows of upset plot indicate the depth-of-coverage per dataset, the columns indicate the intersection of shared single nucleotide variants across datasets. The bar plots located at the bottom left side represent the total number of SNVs identified per dataset.

V. Conclusion

These modifications to two small parts of the LoFreq source code have improved an existing, widely used variant calling software and would not have been possible without a flexible and well-documented code base, which along with their spirit of collaboration is a credit to the developers. The effect of it is the continuous improvement of a single software, a welcome change from the more common pattern where bioinformatics tools proliferate every time a minor modification is made to an algorithm.

Our heuristic improvements to LoFreq have been merged to the main repository, available at https://github.com/CSB5/lofreq. The experimental OpenMP version is available at https://gitlab.com/treangenlab/lofreq/-/tree/openmp/

VI. Acknowledgments

We'd like to thank the authors of LoFreq, Andreas Wilm and Niranjan Nagarajan, for their help and advice as well as merging our improvements into the LoFreq repository. We'd also like to thank John Mellor-Crummey for assistance with HPC-Toolkit and access to additional computing resources.

References

[1] Lucy van Dorp, Mislav Acman, Damien Richard, Liam P Shaw, Charlotte E Ford, Louise Ormond, Christopher J Owen, Juanita Pang, Cedric CS Tan, Florencia AT Boshier, et al. Emergence of genomic diversity and recurrent mutations in sars-cov-2. *Infection, Genetics and Evolution*, 83:104351, 2020.

[2] Nicolae Sapoval, Medhat Mahmoud, Michael Jochum, Yunxi Liu, RA Leo Elworth, Qi Wang, Dreycey Albin, Huw Ogilvie, Michael D Lee, Sonia Villapol, et al. Hidden genomic diversity of sars-cov-2: implications for qrt-pcr diagnostics and transmission. *Genome Research*, pages gr–268961, 2021.

[3] Timokratis Karamitros, Gethsimani Papadopoulou, Maria Bousali, Anastasios Mexias, Sotirios Tsiodras, and Andreas Mentis. Sars-cov-2 exhibits intra-host genomic plasticity and low-frequency polymorphic quasispecies. *Journal of Clinical Virology*, 131:104585, 2020.

[4] Jessica A Plante, Yang Liu, Jianying Liu, Hongjie Xia, Bryan A Johnson, Kumari G Lokugamage, Xianwen Zhang, Antonio E Muruato, Jing Zou, Camila R Fontes-Garfias, et al. Spike mutation D614G alters SARS-CoV-2 fitness. *Nature*, pages 1–6, 2020.

[5] Nicholas G. Davies, Christopher I. Jarvis, W. John Edmunds, Nicholas P. Jewell, Karla Diaz-Ordaz, and Ruth H. Keogh. Increased hazard of death in community-tested cases of SARS-CoV-2 variant of concern 202012/01. *medRxiv*, 2021.

[6] Jason A Reuter, Damek V Spacek, and Michael P Snyder. High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597, 2015.

[7] Ting Ting Wang, Sagi Abelson, Jinfeng Zou, Tiantian Li, Zhen Zhao, John E Dick, Liran I Shlush, Trevor J Pugh, and Scott V Bratman. High efficiency error suppression for accurate detection of low-frequency variants. *Nucleic acids research*, 47(15):e87–e87, 2019.

[8] Sarah Sandmann, Aniek O De Graaf, Mohsen Karimi, Bert A Van Der Reijden, Eva Hellström-Lindberg, Joop H Jansen, and Martin Dugas. Evaluating variant calling tools for non-matched next-generation sequencing data. *Scientific Reports*, 7:43169, 2017.

[9] Andreas Wilm, Pauline Poh Kim Aw, Denis Bertrand, Grace Hui Ting Yeo, Swee Hoe Ong, Chang Hua Wong, Chiea Chuen Khor, Rosemary Petric, Martin Lloyd Hibberd, and Niranjan Nagarajan. LoFreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-population heterogeneity from high-throughput sequencing datasets. *Nucleic Acids Research*, 40(22):11189–11201, 2012.

[10] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[11] William Biscarri, Sihai Dave Zhao, and Robert J. Brunner. A simple and fast method for computing the poisson binomial distribution function. *Computational Statistics and Data Analysis*, 122:92–100, 2018.

[12] Yili Hong. On computing the distribution function for the poisson binomial distribution. *Computational Statistics and Data Analysis*, 59:41–51, 2013.

[13] Joseph L Hodges and Lucien Le Cam. The Poisson approximation to the Poisson binomial distribution. *The Annals of Mathematical Statistics*, 31(3):737–740, 1960.

[14] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, Fabrice Rossi, and Rhys Ulerich. *GNU scientific library*. Citeseer, 2002.

[15] D. Butler, C. Mozsary, C Meydan, et al. Shotgun transcriptome, spatial omics, and isothermal profiling of sars-cov-2 infection reveals unique host responses, viral diversification, and drug interactions. *Nature Communications*, 12(1):1660, 2021.

[16] Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992, 2014.

[17] Laksono Adhianto, Sinchan Banerjee, Mike Fagan, Mark Krentel, Gabriel Marin, John Mellor-Crummey, and Nathan R Tallent. Hpctoolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.