HECIL: A Hybrid Error Correction Algorithm for Long Reads with Iterative Learning

Olivia Choudhury[*], Ankush Chakrabarty[†], Scott J. Emrich[‡]

Abstract

Second-generation sequencing techniques generate short reads that can result in fragmented genome assemblies. Third-generation sequencing platforms mitigate this limitation by producing longer reads that span across complex and repetitive regions. Currently, the usefulness of such long reads is limited, however, because of high sequencing error rates. To exploit the full potential of these longer reads, it is imperative to correct the underlying errors. We propose HECIL—Hybrid Error Correction with Iterative Learning—a hybrid error correction framework that determines a correction policy for erroneous long reads, based on optimal combinations of decision weights obtained from short read alignments. We demonstrate that HECIL outperforms state-of-the-art error correction algorithms for an overwhelming majority of evaluation metrics on diverse real data sets including *E. coli*, *S. cerevisiae*, and the malaria vector mosquito *A. funestus*. We further improve the performance of HECIL by introducing an iterative learning paradigm that improves the correction policy at each iteration by incorporating knowledge gathered from previous iterations via confidence metrics assigned to prior corrections.

Availability and Implementation: https://github.com/NDBL/HECIL

Contact: semrich@nd.edu

1 Introduction

Current advances in next-generation sequencing (NGS) have fueled genomics-driven research by inexpensively generating highly accurate 'reads' or DNA sequence fragments. Second-generation sequencing technologies, for example Illumina [1] and 454 pyro-sequencing [2], generate short reads that are sometimes not ideal for downstream applications such as assembling complex genomes [3]. To ameliorate this issue, third-generation sequencing techniques introduced by Pacific Biosciences [4, 5] and Oxford Nanopore [6, 7, 8] generate significantly longer reads. These long reads typically contain thousands of base-pairs (see for example [9]) and are not subject to amplification or compositional biases often exhibited by second-generation sequencing [10]. Long reads also overcome issues associated with repetitive regions and large transcript isoforms. In spite of these significant advantages, a critical limitation of long reads produced by third-generation sequencing methods is that they generally exhibit high error rates: for example, up to 20% error has been reported using PacBio [11, 12], and 35% using Oxford Nanopore [13].

Various correction algorithms have been proposed for reducing the currently high error rates prevalent in long reads. For example, HGAP [14] is a self-correcting algorithm (that is, it does not rely on additional

[*]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556
[†]Harvard John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138
[‡]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556

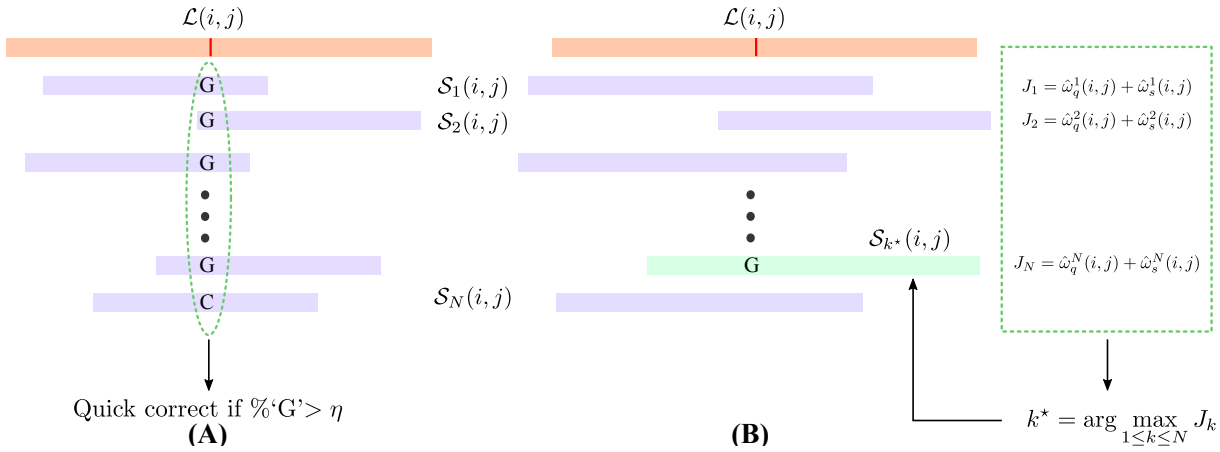

Figure 1: Illustration of HECIL's core algorithm. The orange rectangle denotes an erroneous long read and the purple rectangles represent aligned short reads. (**A**) Quick correction with high consensus. (**B**) Optimization-based correction: The green dashed box depicts the objective function values, from which the optimal short read (green rectangle) is selected for correction.

sequencing data) that performs correction by computing multiple alignments of high coverage long reads. Another class of correction algorithms rely on short reads generated from the same (or related) samples, and is therefore referred as *hybrid correction algorithms*. For instance, the authors in [13] proposed the Nanocorr algorithm in which high-quality Illumina MiSeq reads are used to correct Oxford Nanopore reads.

Popular hybrid correction algorithms for PacBio data include: LSC [3], PacBioToCA [9], LoRDEC [15], proovread [16], and CoLoRMap [17]. Most of the methods listed here do not systematically utilize localized information such as base quality of the short reads or variant information between individuals. The importance of incorporating base quality in correcting noisy sequencing data, for example, has been emphasized in [18].

This motivates our proposed algorithm, HECIL: a novel hybrid error correction framework that computes a correction policy by selecting an optimal combination of decision weights based on base quality and mapping identity of aligned short reads. We compare HECIL's performance to existing hybrid correction algorithms on real prokaryotic and eukaryotic data and, for an overwhelming majority of the evaluation metrics, show that HECIL outperforms its competitors. We also propose an iterative learning framework that improves HECIL's correction accuracy both in terms of alignment and assembly-based metrics by incorporating knowledge derived from high-confidence corrections made in prior iterations. We speculate that the proposed iterative learning formalism can be incorporated into other contemporary hybrid error correction algorithms to improve performance, provided that each iteration does not require prohibitive execution time.

2 Methods

HECIL works under the standard assumption that all reads are derived from highly similar individuals. We begin by aligning the given set of short reads to the long reads. For each alignment, we compute normalized weights using base quality information and alignment identity of the underlying short reads. The short read that maximizes the sum of these normalized weights is used for correction. In this manner, we tend to select higher quality short reads that have a suitable degree of overlap with a long read. This

forms the *core algorithm* of HECIL.

Next, we optionally define a subset of these corrections as "high confidence" and correct only these high-confidence errors. By introducing elitism to the correction procedure based on confidence, the updated long reads now exhibit slightly higher consensus (or similarity) with the short reads. Therefore, we expect to obtain slightly higher quality alignments for fixing lower confidence corrections in subsequent iterations: this is the intuition behind the iterative learning procedure. Herein, we discuss each of these steps in detail.

2.1 HECIL's Core Algorithm

2.1.1 Step 1: Quick Correction

As recommended in [16] and [17], we first obtain read alignments using BWA-MEM [19] and then mark positions with disagreements (for example: mismatches, insertions, and deletions) on long reads as *questionable*. For each questionable position on the long read, we investigate the set of short reads that align to it. If there is strong consensus (determined by a threshold $0 \ll \eta \leq 1$ selected by the user), we replace the questionable base on the long read with the respective aligned base of the short read. This *quick correction* step is illustrated in Fig. 1-(A). This step is inspired by the majority voting method proposed in [3]. However, contrary to corrections based on a simple majority, we adopt a stricter threshold of at least 90% consensus ($\eta = 0.9$) to be eligible for quick correction. Shifting from majority voting to strong consensus prevents spurious corrections made on the basis of high-frequency, low-quality short reads. Note that quick correction also reduces the search space in the next step of HECIL's core algorithm.

2.1.2 Step 2: Optimization-based Correction

For the remaining questionable bases, we employ an optimization-based correction framework. Let $\mathcal{L}(i,j)$ be the jth questionable base corresponding to the ith long read. Suppose N short reads align to this $\mathcal{L}(i,j)$; $\{\mathcal{S}_k(i,j)\}_{k=1}^{N}$ denotes the set of aligning short reads. For each $k = 1, 2, \ldots, N$ we assign two normalized weights $\hat{\omega}_q^k(i,j)$ and $\hat{\omega}_s^k(i,j)$, representing the quality and similarity of the kth short read, respectively.

The normalized quality weight is given by

$$\hat{\omega}_q^k(i,j) := \frac{\omega_q^k(i,j)}{\max_{1\leq k\leq N} \omega_q^k(i,j)},$$

where the scalar $\omega_q(i,j)$ is determined by extracting the PHRED quality score readily available from FASTQ files. The normalized similarity weight $\hat{\omega}_s^k(i,j)$ is obtained by calculating the alignment identity, defined as the number of exact matches of the kth short read $\mathcal{S}_k(i,j)$ to the long read $\mathcal{L}(i,j)$, divided by the length of $\mathcal{S}_k(i,j)$. Untrimmed short reads, therefore, may result in a lower estimated $\hat{\omega}_s^k(i,j)$, which is why we adhere to trimmed short reads in this study. For each short read, we compute a cost by taking a convex combination of the two normalized weights

$$J_k(i,j) = \frac{1}{2}\left(\hat{\omega}_q^k(i,j) + \hat{\omega}_s^k(i,j)\right).$$

We then solve the following optimization problem:

$$k^\star = \underset{1\leq k\leq N}{\arg\max}\ J_k(i,j). \tag{1}$$

which yields the index $k^\star$ of the short read $S_{k^\star}(i,j)$ that exhibits the maximum combined quality and similarity weight. In case there is a conflict amongst maximizers, the short read with highest quality is

selected to be the winner. Note that the optimal cost for each $\mathcal{L}(i,j)$ is denoted by $J_{k^\star}(i,j)$. Subsequently, we replace the erroneous base $\mathcal{L}(i,j)$ on the long read with the corresponding base of the short read $\mathcal{S}_{k^\star}(i,j)$. This procedure is illustrated in Fig. 1(B).

If perfect consensus (that is, $\eta = 1$ in Step 1) is reached amongst all the short reads, there is no need to perform Step 2, because both steps will yield identical corrections. Similarly, if we select a consensus threshold $\eta \in (0,1)$, then the probability that the quick correction value matches the optimization-based correction value is η, irrespective of the cost function selected. Therefore, choosing η close to 1 ensures that quick correction matches optimization-based correction with high-probability. We do not set η strictly equal to 1 hypothesizing that achievement of perfect consensus is rare in practice.

Also note that the quality of a short read and its alignment identity with the long read are not contending objectives. That is, a high quality read does not always imply low similarity and vice versa. Therefore, we consider a convex combination of these objectives as in equation (1) rather than formulating a multi-objective optimization problem and searching for Pareto-optimal solutions.

2.2 Improving Correction Performance via Iterative Learning

A definition of iterative learning that closely resembles our proposed approach in this paper is found in [20]: iterative learning "considers systems that repetitively perform the same task with a view to sequentially improve accuracy". Here, the *same task* refers to the core algorithm of HECIL, and the goal is to improve error corrections in the ℓth iteration by learning from high-confidence corrections in the $(\ell-1)$th iteration (see Fig. 2). Unlike the iterative proovread algorithm, our proposed iterative scheme leverages confidence metrics assigned to prior corrections for informing later decisions.

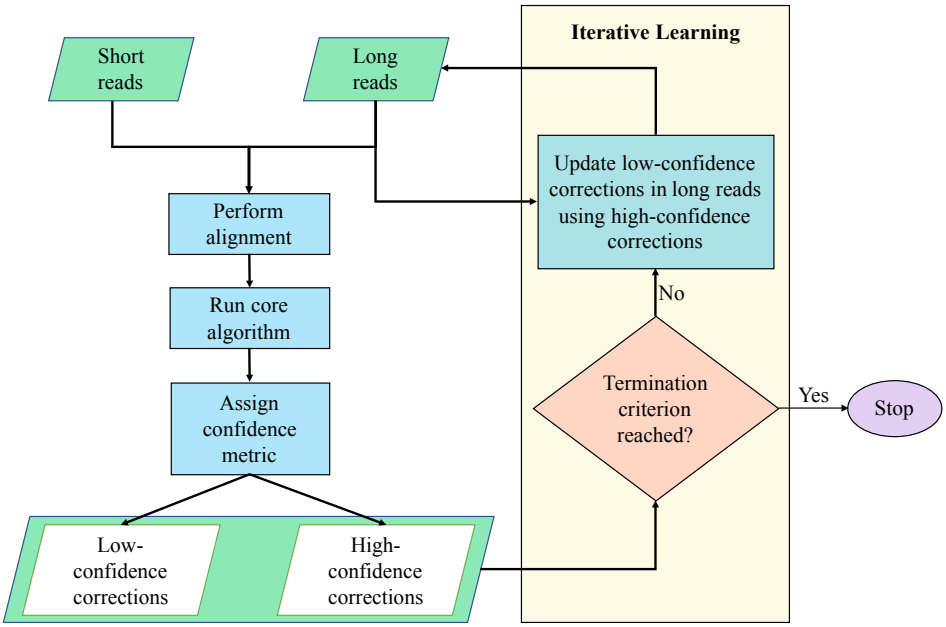

Figure 2: Iterative learning procedure with the HECIL core algorithm for error correction. Alternative hybrid-error correction algorithms could seamlessly replace the core algorithm.

2.2.1 Assignment of confidence

For each $\mathcal{L}(i,j)$ in the ℓth iteration, suppose the corresponding optimal cost obtained by solving (1) be denoted by $J^{(\ell)}_{k^\star}(i,j)$, and let $\mu^{(\ell)}$ denote the α-percentile computed over all these optimal costs. Here we select $\alpha > 0.85$ so that a small percentage of the optimal corrections are considered to be of high confidence. Selecting a high value of α ensures that only the highest quality corrections will always inform future iterations. Adversely, selecting α too close to one will result in slower improvement of correction accuracy, because large α implies that very few corrections are deemed high confidence. Therefore, the increment in information used to update the correction policy in the following iteration will be limited.

2.2.2 Realignment based on high-confidence corrections

We *learn* in successive iterations by re-aligning the updated long reads to the short reads. Note that, for each iteration, the updated context of $\mathcal{L}(i,j)$ could generate entirely different sets of aligned short reads, as well as disparate localized information from previous iterations, leading to the calculation of different sets of normalized weights $\omega^k_{q_{ij}}$ and $\omega^k_{q_{ij}}$. This is why the confidence threshold $\mu^{(\ell)}$ is recomputed based on the statistics of the optimal costs (namely, the percentile measure) and not fixed. The sites on the long read corresponding to low-confidence short reads are left to be changed via the core algorithm in a subsequent iteration while the high confidence changes in prior iterations are effectively fixed.

2.2.3 Termination criteria

We propose two criteria for termination. In the first criterion, if the fractional reduction in the number of unique k-mers (overlapping substrings of length k) between two successive iterations is below a given threshold $\varepsilon \in (0,1)$. That is,

$$\frac{\#\text{ unique } k\text{-mers}(\ell-1) - \#\text{ unique } k\text{-mers}(\ell)}{\#\text{ unique } k\text{-mers}(\ell-1)} < \varepsilon$$

then we terminate after the ℓth iteration. Specific arguments why k-mers are used for termination are provided in Section 3.2.1.

The second criterion involves pre-calculating the number of iterations to achieve at least M' high-confidence corrections. At each iteration, $(1-\alpha)\times 100\%$ of the corrections are retained (by definition of the α-percentile), implying that αM errors remain for the next iteration. Thus, the number of low-confidence corrections remaining after n iterations will be $\alpha^n M$. If the user requires at least M' high-confidence corrections to be made iteratively, one needs to select n that satisfies $M' < (1-\alpha^n)M$. In such a case, algebraic manipulations and taking logarithms yield the following bound on the required number of iterations

$$n > \frac{1}{\log \alpha} \log \left| 1 - \frac{M'}{M} \right|. \tag{2}$$

Note that both logarithm terms are negative. Based on the above, after n corrections, where n satisfies (2), we expect that at least M' high confidence corrections will be made.

3 Results

All experiments in this section were run on Dell PowerEdge R815 servers with AMD Opteron processor 6378, Quad 16 core 2.4 GHz CPU, 32 cores, 512 GB RAM, and 2 x 300 GB 15K RPM SAS drives. We use the Unix `time` command to record the runtime and memory usage of each tool.

3.1 Data acquisition

We test the performance of HECIL on three datasets of varying size: the bacterial genome of *E. coli*, the fungal genome of *S. cerevisiae*, and the malaria vector genome of *A. funestus*. We explore PacBio-sequenced long reads, Illumina-sequenced short reads, and reference genomes of *E. coli* and *S. cerevisiae*, as suggested in the supplementary material of [17]. Long reads of *E. coli* are filtered to exclude reads shorter than 100 bp. The final set contains 33,360 reads that total 98 million bases (Mbp). The corresponding short reads (accession ID: ERR022075) comprise 22,720,100 reads that are 202 bp long. The strain K-12 substr. MG1655 is used for our alignment-based validation of HECIL. To test *S. cerevisiae* data we use 1,758,169 long reads, with 1,402 Mbp and 4,503,422 short reads (accession ID: SRR567755). The reference genome of strain S288C is 12.2 Mbp in size. We obtain long reads for *A. funestus*[1], comprising data from 44 flowcells, ranging between 59,937 and 244,754 reads. Due to the high computational time required by proovread and CoLoRMap to correct the reads of all flowcells, we present a comparative analysis based on a random selection of three: flowcells 1, 4, and 16. Short sequences (accession ID: SRR630594) consists of 37,797,235 reads. The reference genome of strain Fumoz (GenBank assembly accession: GCA_000349085.1) is used for validating corrections.

3.2 Evaluation metrics

3.2.1 *k*-mer-based

We employ the widely-used k-mer counting tool Jellyfish [21] to compute the number of unique k-mers obtained after each correction algorithm is executed. Since errors in long reads are uniformly distributed across their length, large numbers of uncorrected errors often greatly inflate the number of unique k-mers observed. Further, the authors in [22] reported that the set of common k-mers between the highly accurate short reads and the erroneous long reads were crucial in improving the quality of data for downstream analysis. Therefore, a correction algorithm that reduces the number of unique k-mers while increasing the number of valid k-mers is desirable. Fig, 3 gives an illustrative example of this idea based on one *A. funestus* flowcell.

3.2.2 Alignment-based

After each method of correction, we align corrected long reads to its reference genome using BLASR [23]. In addition to computing the number of aligned reads and aligned bases, we evaluate matched bases, that is, the ratio of total number of matched bases and length of sequences in the long reads. We calculate percent identity (PI) by the ratio of matches to alignment length. Since PI is usually above 90%, we also determine the number of aligned reads with PI above 90%.

3.2.3 Assembly-based

One of the most important downstream applications of long reads is *de novo* genome assembly. For this purpose, we use the assembler Canu [24], specifically designed for noisy long reads. We then use QUAST [25] to evaluate the quality of the resulting assembly. We measure total number of contigs, length of the longest contig, and total length, i.e., total number of bases in the assembly. We report the values of N50 (minimum length such that contigs of that length or longer consists half the assembly), and NG50

[1] in process of submission to NCBI

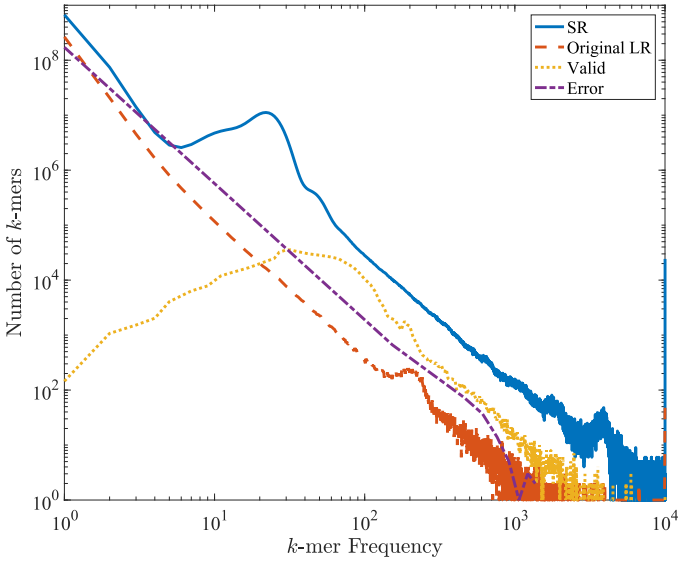

Figure 3: Distribution of k-mer frequency (k=17) in *Anopheles funestus* flowcell #16. The x and y-axes denote k-mer frequency and count of frequency, respectively. The blue line and dashed red line represent k-mers generated from short reads (SR) and original long reads (Original LR), respectively. As discussed in Section 3.2.1, the dotted yellow line indicates that majority of the valid k-mers have high frequency. The purple dot-dashes, representing error k-mers (not found in short reads), mostly consists of unique k-mers.

(minimum length such that contigs of that length or longer consists half the reference assembly). As suggested in [24], we further measure accuracy by aligning the assembled genome to the reference genome using MUMMer's dnadiff tool [26]. In this context, we compute percent of aligned bases (with respect to reference and query) and average identity of 1-to-1 alignment blocks (with respect to reference and query).

3.3 Comparative analysis

We compare the performance of HECIL with cutting-edge hybrid error correction tools such as proovread-2.14.0, LoRDEC-0.6, and CoLoRMap. We use the above-mentioned k-mer-based, alignment-based, and assembly-based metrics to assess the performance of each approach. The comparative results for k-mer-based and alignment-based parameters are presented in Table 1. We report the parameters before correction (original) and after each method of error correction.

As discussed in [17], CoLoRMap performs better than proovread and LoRDEC, when tested on *E. coli* and *S. cerevisiae*. Long reads corrected by the core algorithm of HECIL (iteration 1) generate the lowest number of k-mers, with the exception of the data set *A. funestus* - flowcell 4, where it is still comparable to the best results obtained from proovread. A corresponding increase in valid k-mers indicates higher consensus to the accurate short reads, hence higher accuracy of the corrected long reads. For all data sets, HECIL consistently produces more valid k-mers. It also results in highest number of aligned bases and reads, highest value of percent identity, and reads with at least 90% percent identity to its reference.

We present the results of assembly-based metrics in Table 4. For *E. coli* and yeast, HECIL generates more contiguous assembled long reads, compared to the existing tools, except CoLoRMap where there are ties. The size of the longest contig and the number of bases in the assembled data are highest for our proposed approach. The standard assembly quality parameters like N50 and NG50 have highest values after using HECIL for correction. Furthermore, the assembled genomes of HECIL have higher aligned

bases and 1-to-1 alignment identity. Since we use a subset of mosquito's flowcell data, the proportion of aligned bases in the reference genome with respect to the query genome is low.

For highly heterozygous samples such as insects like mosquitoes [27], low frequency bases in aligned short reads may indicate inherent variation that are not necessarily sequencing errors. Correction algorithms that solely rely on a consensus call or majority vote often discard these heterogenous alleles. The optimization in Step 2 of HECIL is not biased by bases which have high frequency, and hence, is better able to capture variation between similar individuals. This is corroborated by the results obtained from testing HECIL on the highly heterozygous mosquito data set of *Anopheles funestus*.

Although the performance of hybrid correction algorithms largely depend on the set of high coverage short reads, we devise additional experiments to verify that restraining the coverage of short reads does not have a deleterious effect on HECIL. We down-sample short reads by randomly selecting 50%, 25%, and 12% of the data to be used for correction. In *E. coli*, this results in a subset of short reads for correction with an average coverage of 62×, 33×, and 18×, respectively. In Table 2, we present k-mer-based and alignment-based parameters from correcting long reads of *E. coli* with the down-sampled short reads using HECILand assembly-based parameters from the lowest coverage (18×; Table 4). Thus, HECIL shows potential for use in projects that do not have high coverage short read data readily available: this is especially important in larger eukaryotic genomes sequenced predominantly with longer read technology.

In Table 3, we compare the runtimes and maximum memory usage incurred in correcting each data set (see Methods). proovread, LoRDEC, and CoLoRMap were run with 16 threads. The workload of HECIL was split into 16 concurrent tasks, which were run in parallel. Computation time of hybrid error correction methods is mainly dominated by the underlying steps of generating intermediate data, such as mapping short reads to the long reads. Similarly, LoRDEC and CoLoRMap construct a graph data structure, which demands high computational resources. LoRDEC, however, uses the efficient GATB library [28], which lowers the overhead (see Table 3). Although our tool incurs higher computation time than LoRDEC, it is consistently faster (generally almost twice as fast) than the other correction methods and generates overall higher quality corrected long reads without a significant increase in memory consumption.

3.4 Effect of Iterative Learning

We leverage our proposed iterative learning scheme on HECIL to demonstrate its effectiveness in further improving correction accuracy. We select a high-confidence cut-off of 85 percentile, that is, $\alpha = 0.85$. The alignment-based incremental improvements obtained after each iterative correction of HECIL is presented in Fig. 4. For each data set (each column), we observe that the incremental metrics: number of fewer k-mers, number of additional aligned long reads, number of additional aligned bases, and additional percent of matched bases, improve after each iteration, until one of the termination criteria is reached. For the termination criteria, we select ε as 0.02 for the metric of unique k-mers and choose $M'/M = 0.5$, which, using (2) results in at most $n = 5$ iterations. Based on this, we report alignment-based and assembly-based metrics obtained after the fifth iteration of HECIL in Tables 1 and 4, respectively. HECIL in conjunction with iterative learning consistently outperforms all the evaluation metrics. The colored cells in these tables indicate the parameters where the core algorithm of HECIL is comparable but does not outperform the alternatives, and the iterative version of HECIL consistently results in better performance. These results verify the utility of iterative learning-based extension of HECIL, particularly in heterozygous samples like the mosquito data set used in this study.

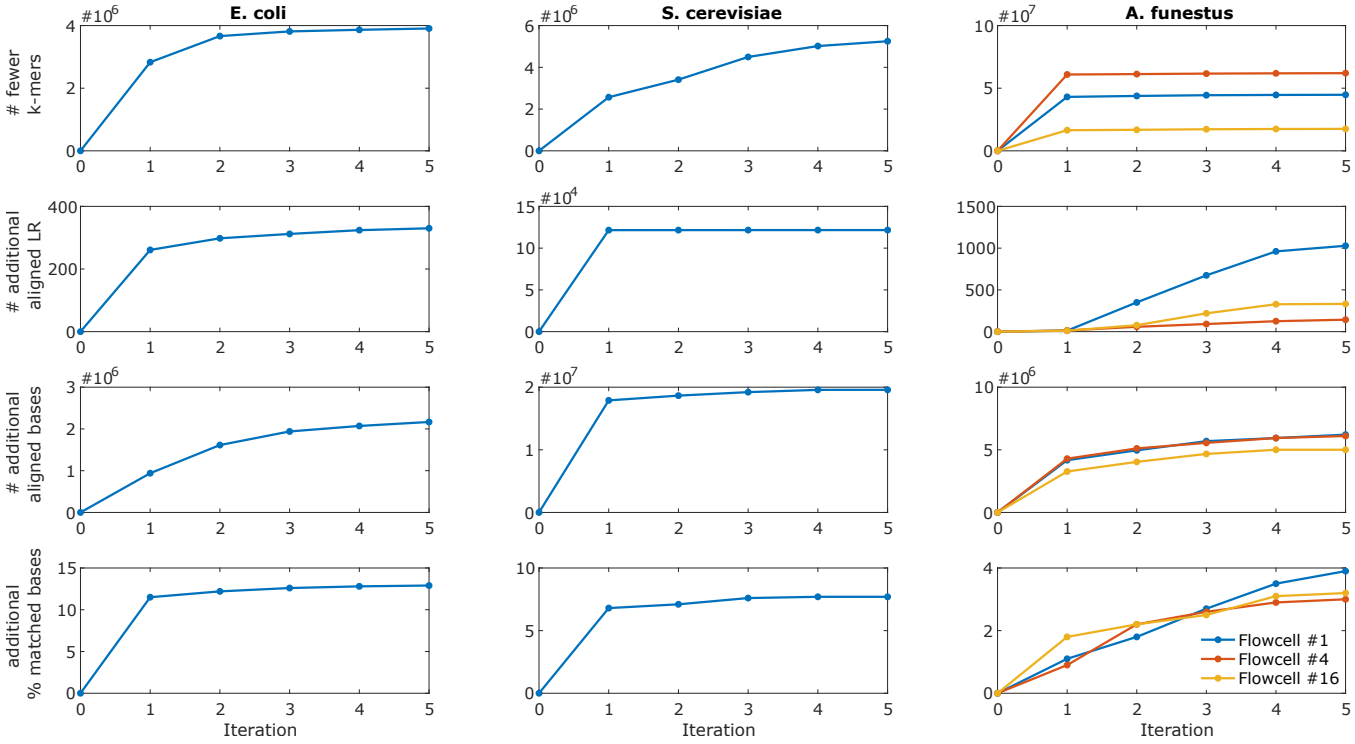

Figure 4: Improvement of alignment-based metrics (# fewer unique k-mers, additional aligned long reads, additional aligned bases, additional percent matched bases) for *E. coli*, *S. cerevisiae*, and *A. funestus* with iterative learning. The 0th iteration denotes the original data set and the 1st iteration indicates corrected data set obtained from running HECIL's core algorithm.

4 Discussion

Third-generation sequencing techniques, particularly Single-Molecule Real-Time (SMRT) sequencing, is revolutionizing modern genomics. The usefulness of current long read data, however, is restricted due to high sequencing error rates. Hence, it is crucial to correct long reads prior to downstream applications like *de novo* genome assembly.

In this paper, we develop a novel approach of hybrid error correction called HECIL, which corrects erroneous long reads based on optimal combinations of base quality and mapping identity of aligned short reads. As seen in Tables 1 and 4, HECIL performs significantly better for an overwhelming majority of evaluation metrics, even with limited amounts of short reads available for correction.

To the best of our knowledge, this is the first time an iterative strategy for improving correction quality via informed realignment has been proposed. The method shows potential using the HECIL core algorithm, but could also be seamlessly integrated with other error correction algorithms that leverage short read alignments.

The current version of HECIL allows decomposition of the workload into independent data-parallel tasks that can be executed simultaneously. A natural extension of the tool is implementing multi-threading to achieve speedup on traditional machines.

Acknowledgements

This work was supported by the Eck Institute for Global Health (EIGH) Ph.D. fellowship to Choudhury, and NIH R21AI123967 to Emrich.

References

[1] D. R. Bentley, "Whole-genome re-sequencing," *Current opinion in genetics & development*, vol. 16, no. 6, pp. 545–552, 2006.

[2] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, M. S. Braverman, Y.-J. Chen, Z. Chen *et al.*, "Genome sequencing in microfabricated high-density picolitre reactors," *Nature*, vol. 437, no. 7057, pp. 376–380, 2005.

[3] K. F. Au, J. G. Underwood, L. Lee, and W. H. Wong, "Improving PacBio long read accuracy by short read alignment," *PloS one*, vol. 7, no. 10, p. e46679, 2012.

[4] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman *et al.*, "Real-time DNA sequencing from single polymerase molecules," *Science*, vol. 323, no. 5910, pp. 133–138, 2009.

[5] J. Korlach, K. P. Bjornson, B. P. Chaudhuri, R. L. Cicero, B. A. Flusberg, J. J. Gray, D. Holden, R. Saxena, J. Wegener, and S. W. Turner, "Real-time DNA sequencing from single polymerase molecules," *Methods in enzymology*, vol. 472, pp. 431–455, 2010.

[6] G. M. Cherf, K. R. Lieberman, H. Rashid, C. E. Lam, K. Karplus, and M. Akeson, "Automated forward and reverse ratcheting of DNA in a nanopore at 5-A precision," *Nature biotechnology*, vol. 30, no. 4, pp. 344–348, 2012.

[7] M. Eisenstein, "Oxford Nanopore announcement sets sequencing sector abuzz," 2012.

[8] E. A. Manrao, I. M. Derrington, A. H. Laszlo, K. W. Langford, M. K. Hopper, N. Gillgren, M. Pavlenok, M. Niederweis, and J. H. Gundlach, "Reading DNA at single-nucleotide resolution with a mutant MspA nanopore and phi29 DNA polymerase," *Nature biotechnology*, vol. 30, no. 4, pp. 349–353, 2012.

[9] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis *et al.*, "Hybrid error correction and de novo assembly of single-molecule sequencing reads," *Nature biotechnology*, vol. 30, no. 7, pp. 693–700, 2012.

[10] E. E. Schadt, S. Turner, and A. Kasarskis, "A window into third-generation sequencing," *Human molecular genetics*, vol. 19, no. R2, pp. R227–R240, 2010.

[11] K. J. Travers, C.-S. Chin, D. R. Rank, J. S. Eid, and S. W. Turner, "A flexible and efficient template format for circular consensus sequencing and snp detection," *Nucleic acids research*, vol. 38, no. 15, pp. e159–e159, 2010.

[12] J. F. Thompson and P. M. Milos, "The properties and applications of single-molecule DNA sequencing," *Genome biology*, vol. 12, no. 2, p. 217, 2011.

[13] S. Goodwin, J. Gurtowski, S. Ethe-Sayers, P. Deshpande, M. C. Schatz, and W. R. McCombie, "Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome," *Genome research*, vol. 25, no. 11, pp. 1750–1756, 2015.

[14] C.-S. Chin, D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler *et al.*, "Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data," *Nature methods*, vol. 10, no. 6, pp. 563–569, 2013.

[15] L. Salmela and E. Rivals, "LoRDEC: accurate and efficient long read error correction," *Bioinformatics*, p. btu538, 2014.

[16] T. Hackl, R. Hedrich, J. Schultz, and F. Förster, "proovread: large-scale high-accuracy PacBio correction through iterative short read consensus," *Bioinformatics*, vol. 30, no. 21, pp. 3004–3011, 2014.

[17] E. Haghshenas, F. Hach, S. C. Sahinalp, and C. Chauve, "CoLoRMap: Correcting Long Reads by Mapping short reads," *Bioinformatics*, vol. 32, no. 17, pp. i545–i551, 2016.

[18] D. R. Kelley, M. C. Schatz, and S. L. Salzberg, "Quake: quality-aware detection and correction of sequencing errors," *Genome biology*, vol. 11, no. 11, p. R116, 2010.

[19] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997*, 2013.

[20] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," *IEE Proceedings-Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996.

[21] G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," *Bioinformatics*, vol. 27, no. 6, pp. 764–770, 2011.

[22] A. B. Carvalho, E. G. Dupim, and G. Goldstein, "Improved assembly of noisy long reads by k-mer validation," *Genome Research*, vol. 26, no. 12, pp. 1710–1720, 2016.

[23] M. J. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory," *BMC bioinformatics*, vol. 13, no. 1, p. 238, 2012.

[24] S. Koren, B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy, "Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation," *bioRxiv*, p. 071282, 2017.

[25] A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler, "QUAST: quality assessment tool for genome assemblies," *Bioinformatics*, vol. 29, no. 8, pp. 1072–1075, 2013.

[26] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg, "Versatile and open software for comparing large genomes," *Genome biology*, vol. 5, no. 2, p. R12, 2004.

[27] R. R. Love, N. I. Weisenfeld, D. B. Jaffe, N. J. Besansky, and D. E. Neafsey, "Evaluation of DISCOVAR de novo using a mosquito sample for cost-effective short-read genome assembly," *BMC genomics*, vol. 17, no. 1, p. 187, 2016.

[28] GATB, *GATB library*, accessed April 3, 2017. [Online]. Available: http://gatb-core.gforge.inria.fr

Data	Evaluation Metric	Original	proovread	LoRDEC	CoLoRMap	HECIL (Iter 1)	HECIL (Iter 5)
E. coli	# unique k-mers	81,523,648	78,925,288	80,708,419	80,399,425	**78,693,704**	**77,617,181**
	# valid k-mers	14,531,881	11,463,127	10,240,970	15,026,950	**15,973,826**	**16,413,012**
	# aligned reads	31,071	23,453	30,837	31,271	**31,332**	**31,401**
	# aligned bases	86,642,500	71,320,858	79,365,407	83,344,272	**87,582,014**	**88,809,361**
	% matched bases	76.9	87.9	85.2	87.5	**88.4**	**89.4**
	PI	94.8	**99.7**	99.4	99.2	**99.7**	**99.8**
	# reads with PI > 90%	4,190	1,742	4,832	4,430	**5,408**	**5,493**
S. cerevisiae	# unique k-mers	1,870,396,869	1,871,451,237	1,868,238,946	1,869,232,456	**1,867,828,519**	**1,865,148,289**
	# valid k-mers	36,904,129	32,436,294	30,534,546	37,797,300	**39,452,743**	**40,971,328**
	# aligned reads	224,694	222,976	221,692	223,641	**346,242**	**346,307**
	# aligned bases	1,229,724,663	1,205,706,114	1,171,490,123	1,207,729,568	**1,247,616,674**	**1,249,303,521**
	% matched bases	78.8	83.1	83.4	**85.6**	**85.6**	**86.5**
	PI	93.8	96.3	98.3	98.3	**98.6**	**98.9**
	# reads with PI > 90%	70,346	27,342	70,529	42,136	**72,562**	**72,657**
A. funestus # 1	# unique k-mers	692,831,731	649,989,172	653,931,808	662,366,838	**649,764,906**	**648,091,381**
	# valid k-mers	211,908,809	172,074,427	229,625,736	222,195,325	**242,957,349**	**244,317,225**
	# aligned reads	190,217	94,536	**190,240**	190,166	190,229	**191,245**
	# aligned bases	671,881,278	401,850,047	655,072,426	660,848,583	**676,055,060**	**678,092,137**
	% matched bases	84.0	81.4	83.1	82.1	**85.1**	**87.9**
	PI	94.5	96.8	95.6	97.1	**97.8**	**98.5**
	# reads with PI > 90%	30,487	4,096	36,364	13,220	**49,810**	**50,997**
A. funestus # 4	# unique k-mers	216,327,700	**205,053,236**	205,883,182	206,986,374	205,064,188	**203,997,977**
	# valid k-mers	80,612,612	72,716,589	82,568,831	81,027,437	**83,788,157**	**84,529,123**
	# aligned reads	59,163	32,726	59,165	59,159	**59,177**	**59,306**
	# aligned bases	231,326,514	149,049,154	234,098,182	233,435,402	**235,620,667**	**237,428,249**
	% matched bases	86.3	83.2	87.0	85.6	**87.2**	**89.3**
	PI	94.3	96.9	96.6	97.2	**97.7**	**98.4**
	# reads with PI > 90%	20,253	2,094	22,042	12,526	**22,483**	**22,567**
A. funestus # 16	# unique k-mers	265,998,542	250,267,133	252,291,701	254,293,778	**249,528,780**	**248,471,673**
	# valid k-mers	96,317,177	86,396,798	106,713,483	101,431,900	**109,954,860**	**110,798,014**
	# aligned reads	73,779	43,530	73,757	73,750	**73,790**	**74,111**
	# aligned bases	278,976,792	190,054,632	280,699,552	280,831,201	**282,244,589**	**283,981,841**
	% matched bases	84.3	82.7	85.6	84.5	**86.1**	**87.5**
	PI	94.8	96.9	96.3	97.4	**98.0**	**98.6**
	# reads with PI > 90%	22,111	2,164	22,945	10,418	**23,771**	**23,874**

Table 1: Comparison of alignment-based metrics obtained from testing *E. coli*, *S. cerevisiae*, and *A. funestus* on proovread, LoRDEC, CoLoRMap, and HECIL. For the case of HECIL, metrics are reported before and after using the iterative learning algorithm; specifically, iteration 1 (the core algorithm) and iteration 5 (with four rounds of learning) are shown. The colored cells in the table denote metrics where the core algorithm did not outperform its contemporaries, but with iterative learning, achieved the best performance.

Evaluation Metric	**All SRs**	**50% SRs**	**25% SRs**	**12% SRs**
# k-mers	84,294,561	84,121,231	83,986,262	83,909,706
# unique k-mers	78,693,704	78,292,463	78,097,941	78,008,319
# valid k-mers	15,973,826	15,889,155	15,737,641	15,576,317
# aligned reads	31,332	31,328	31,322	31,318
# aligned bases	87,582,014	87,359,227	87,288,475	87,196,236
% matched bases	88.4	88.4	88.3	88.3
PI	99.7	99.7	99.7	99.6
# reads with PI > 90%	5,408	5,358	5,298	5,223

Table 2: Comparison of alignment-based metrics with downsampled *E. coli* short reads using HECIL's core algorithm.

Data	**Method**	**Runtime**	**Memory**
E. coli	proovread	6:15:37	11.4
	LoRDEC	38:53	6.2
	CoLoRMap	2:48:23	28.9
	HECIL	1:16:55	9.1
S. cerevisiae	proovread	20:54:15	14.5
	LoRDEC	3:43:12	6.1
	CoLoRMap	7:57:49	38.2
	HECIL	5:14:09	11.2
A. funestus (Flowcell # 1)	proovread	76:13:47	8.8
	LoRDEC	35:08:13	3.1
	CoLoRMap	90:50:12	23.4
	HECIL	46:06:47	8.3
A. funestus (Flowcell # 4)	proovread	36:32:25	7.3
	LoRDEC	11:25:05	6.7
	CoLoRMap	32:18:30	20.7
	HECIL	17:38:01	6.9

Table 3: Comparison of runtime (hh:mm:ss) and maximum memory footprint (GB) for correcting long reads. Only the best and worst *A. funestus* single iterations are shown.

Data	Evaluation Metric	Original	proovread	LoRDEC	CoLoRMap	HECIL (Iter 1)	HECIL (Iter 5)
E. coli (D-SR)	# Contigs	182	29	28	24	**20**	—
	Largest contig	69,266	567,484	885,819	813,262	**1,204,631**	—
	Total length	3,508,197	4,235,031	4,068,085	4,036,161	**4,596,013**	—
	N50	24,663	189,712	179,638	184,367	**232,826**	—
	NG50	17,847	212,621	190,621	210,913	**267,311**	—
	Aligned base (%) - Ref / Query	83 / 84	87 / 89	92 / 93	48 / 92	**97 / 100**	—
	Average Identity (1-1) - Ref / Query	88 / 88	93 / 93	97 / 97	97 / 97	**99 / 99**	—
E. coli	# Contigs	182	26	24	**19**	**19**	**17**
	Largest contig	69,266	605,792	920,903	1,089,140	**1,223,474**	**1,481,824**
	Total length	3,508,197	4,629,719	4,623,137	4,624,793	**4,838,971**	**5,106,276**
	N50	24,663	231,774	226,456	239,066	**256,830**	**288,192**
	NG50	17,847	231,774	226,456	239,066	**294,635**	**344,848**
	Aligned base (%) - Ref / Query	82 / 87	92 / 92	98 / 98	54 / 94	**99 / 99**	**99 / 99**
	Average Identity (1-1) - Ref / Query	91 / 91	95 / 95	96 / 96	97 / 97	**98 / 98**	**99 / 99**
S. cerevisiae	# Contigs	26	32	28	**24**	**24**	**23**
	Largest contig	1,543,990	1,537,979	1,552,711	1,555,857	**1,558,190**	**1,713,201**
	Total length	12,341,981	12,485,995	**12,497,078**	12,315,869	12,435,702	**12,731,203**
	N50	777,602	777,713	818,962	932,935	**1,018,591**	**1,308,313**
	NG50	777,602	777,713	818,962	932,935	**1,538,190**	**2,005,346**
	Aligned base (%) - Ref / Query	95 / 90	91 / 91	95 / 95	78 / 97	**99 / 99**	**99 / 99**
	Average Identity (1-1) - Ref / Query	92 / 92	93 / 93	97 / 97	98 / 98	**99 / 99**	**99 / 99**
A. funestus	# Contigs	998	712	788	847	**633**	**543**
	Largest contig	71,070	36,306	75,298	72,306	**84,490**	**94,937**
	Total length	25,405,949	8,371,287	26,745,092	26,802,126	**28,954,268**	**32,371,298**
	N50	13,038	14,802	15,118	14,555	**16,409**	**19,014**
	NG50	71,070	45,637	77,294	76,306	**84,490**	**91,303**
	Aligned base (%) - Ref / Query	20 / 87	23 / 93	27 / 96	20 / 95	**31 / 99**	**37 / 99**
	Average Identity (1-1) - Ref / Query	83 / 83	87 / 87	95 / 95	92 / 92	**98 / 98**	**99 / 99**

Table 4: Comparison of assembly-based metrics obtained from testing *E. coli*: with downsampled short reads (D-SR) having 18x coverage (lowest coverage) and original short reads, *S. cerevisiae*, *A. funestus* (merged flowcells) on proovread, LoRDEC, CoLoRMap, and HECIL. For the case of HECIL, metrics are reported before and after using the iterative learning algorithm; specifically, iteration 1 (the core algorithm) and iteration 5 (with four rounds of learning) are shown. The colored cells in the table denote metrics where the core algorithm did not outperform its contemporaries, but with iterative learning, achieved best results.