BIOINFORMATICS

Vol. 27 ISMB 2011, pages i137–i141
doi:10.1093/bioinformatics/btr208

Error correction of high-throughput sequencing datasets with non-uniform coverage

Paul Medvedev[1,*], Eric Scott[2], Boyko Kakaradov[2] and Pavel Pevzner[1]

[1]Department of Computer Science and Engineering and [2]Bioinformatics Program, University of California, San Diego, CA, USA

ABSTRACT

Motivation: The continuing improvements to high-throughput sequencing (HTS) platforms have begun to unfold a myriad of new applications. As a result, error correction of sequencing reads remains an important problem. Though several tools do an excellent job of correcting datasets where the reads are sampled close to uniformly, the problem of correcting reads coming from drastically non-uniform datasets, such as those from single-cell sequencing, remains open.

Results: In this article, we develop the method Hammer for error correction without any uniformity assumptions. Hammer is based on a combination of a Hamming graph and a simple probabilistic model for sequencing errors. It is a simple and adaptable algorithm that improves on other tools on non-uniform single-cell data, while achieving comparable results on normal multi-cell data.

Availability: http://www.cs.toronto.edu/~pashadag.

Contact: pmedvedev@cs.ucsd.edu

1 INTRODUCTION

The continuing improvements to high-throughput sequencing (HTS) platforms have begun to unfold a myriad of new and exciting applications such as transcriptome analysis, metagenomics, single-cell assembly and variation detection. In addition, classical problems such as assembly are becoming feasible for large-scale endeavours, such as the Genome 10K project which seeks to assemble the genomes of 10 000 novel species (Genome 10K Community of Scientists, 2009). All these projects face the same difficulty of handling the base errors that are inevitably prevalent in HTS datasets. Error correction of reads has thus come to the forefront as an essential problem, with a slew of publications in the last 2 years (Ilie *et al.*, 2010; Kelley *et al.*, 2010; Qu *et al.*, 2009; Salmela, 2010; Schroder *et al.*, 2009; Shi *et al.*, 2010; Wijaya *et al.*, 2009; Yang *et al.*, 2010; Zhao *et al.*, 2010).

A common approach of error correction of reads is to determine a threshold and correct k-mers whose multiplicities fall below the threshold (Chaisson and Pevzner, 2008; Kelley *et al.*, 2010; Pevzner *et al.*, 2001; Shi *et al.*, 2010; Zhao *et al.*, 2010). Choosing the correct threshold is crucial since a low threshold will result in too many uncorrected errors, while a high threshold will result in the loss of correct k-mers. The histogram of the multiplicities of k-mers will show a mixture of two distributions—that of the error-free k-mers, and that of the erroneous k-mers. When the coverage is high and uniform, these distributions are centered far apart and can be separated without much loss using a cutoff threshold; such methods therefore achieve excellent results (Kelley *et al.*, 2010).

Though HTS platforms provide relatively uniform coverage in many standard sequencing experiments, in some of the more challenging applications, such as transcriptome sequencing, the coverage remains drastically uneven (Li *et al.*, 2010; Qu *et al.*, 2009). Another prominent emerging example is that of single-cell sequencing, which has enabled the investigation (Kvist *et al.*, 2007; Mussmann *et al.*, 2007) of a diverse range of uncultivated bacteria, from the surface ocean environment (Yooseph *et al.*, 2007) to the human body (Gill *et al.*, 2006). Though these bacteria are not amenable to normal sequencing, recent advances in DNA amplification technology have enabled genome sequencing directly from individual cells without requiring growth in culture (Raghunathan *et al.*, 2005; Rodrigue *et al.*, 2009). The read datasets thus obtained from single-cells suffer from amplification bias, resulting in orders of magnitude difference in coverage (Raghunathan *et al.*, 2005) and the complete absence of coverage in some regions. Previous, single-cell studies have used error correction tools (Margulies *et al.*, 2005; Zerbino and Birney, 2008) that assume near-uniform coverage. However, such approaches are not as effective and result in decreased quality of assemblies when the two multiplicity distributions of erroneous and error-free k-mers are not separable using a simple threshold. Thus, despite the initial technological difficulties, the challenges facing single-cell genomics are increasingly computational rather than experimental (Rodrigue *et al.*, 2009). For applications such as these, it becomes paramount to develop better error-correction algorithms that do not assume uniformity of coverage.

Recent papers have introduced a powerful idea that has the potential to remove uniformity assumptions (Ilie *et al.*, 2010; Qu *et al.*, 2009; Salmela, 2010; Schroder *et al.*, 2009; Wijaya *et al.*, 2009; Yang *et al.*, 2010). Consider two k-mers x and y that are within a small Hamming distance and present in the read dataset. If our genome does not contain many non-exact repeats, then it is likely that both x and y were generated by the k-mer among them that has higher multiplicity. In this way, we can correct the lower multiplicity k-mer to the higher multiplicity one, without relying on uniformity. This can be further generalized by considering the notion of the *Hamming graph*, whose vertices are the distinct k-mers (annotated with their multiplicities) and edges connect k-mers with a Hamming distance less than or equal to a parameter τ (Yang *et al.*, 2010).

In this article, we develop the method Hammer for error correction without any uniformity assumptions. Hammer is based on a combination of the Hamming graph and a simple probabilistic model. It is a simple and adaptable algorithm that improves on

*To whom correspondence should be addressed.

other tools on non-uniform single-cell *Escherichia coli* data, while achieving comparable results on normal multi-cell data.

2 METHODS

We will first present the simple probabilistic model on which Hammer is based, and then present the algorithm.

2.1 Model

We model sequencing as a process that repeatedly identifies a genomic position at random with an arbitrary (not necessarily uniform) distribution. Then for each position of u, it introduces a mutation with probabilty ϵ, and generates the resulting k-mer for the dataset. We denote $G(u)$ as the set of all k-mers that were generated by u, and we call u a *generating* k-mer.

Let us assume for the moment that we are given a multi-set $C=G(u)$ of all the k-mers that were generated by u, but we do not know the identity of u. We wish to find the k-mer x that maximizes the likelihood of being the generator for C. The likelihood that x generates C is given by

$$L(x)=\prod_{y\in C}(1-\epsilon)^{k-d(x,y)}\epsilon^{d(x,y)}$$

where $d(x,y)$ is the Hamming distance between x and y. It follows that maximizing $L(x)$ is equivalent to minimizing

$$S(x)=\sum_{y\in C}d(x,y) \qquad (1)$$

which we call the *score*.

Define the *consensus* of C as a k-mer that at position i contains the most frequent nucleotide at that position in C, with ties broken arbitrarily. Finding the k-mer x that minimizes $S(x)$ turns out to be easy (Jones and Pevzner, 2004):

Observation 1. *The k-mer x minimizing $S(x)$ is the consensus of C.*

This observation is at the basis of the Hammer algorithm. In reality, we do not know the multi-sets $C=G(u)$; however, we can use the connected components of the Hamming graph to approximate the multi-sets C. We now present the steps of the algorithm.

2.2 Algorithm

Hammer first uses a sort to identify the set K of distinct k-mers in the reads as well as a table $m(x)$ storing the multiplicity of each k-mer. Next, it finds the connected components of the Hamming graph. It is computationally prohibitive to do an all-pairs comparison between the k-mers; however, this is a classical problem from combinatorial pattern matching for which there exist well-studied solutions that rely on spaced seeds (Jones and Pevzner, 2004):

Observation 2. *Let $s^1,\dots,s^{\tau+1}$ be a disjoint partition of $\{1,\dots,k\}$. These represent spaced seeds into k-mers, and we denote the restriction of a k-mer x to the seed s^j as $x(s^j)$. Then, if $d(x,y)\leq\tau$ for some k-mers x and y, there exists j such that $x(s^j)=y(s^j)$.*

Based on this observation, we create $\tau+1$ spaced seeds $s_j=\{j,j+\tau+1,j+2(\tau+1),\dots\}$. Then we make $\tau+1$ copies of the k-mer set K with the j-th copy containing the restriction of K to the indices s^j. Each copy is then lexicographically sorted. In order to account for double-strandedness, we include the reverse complements of every k-mer in our set. Next, we linearly scan through each copy, and identify equivalence blocks, i.e. sets of k-mers that are equal when restricted to s_j. For every equivalence block, we compute the Hamming distances of all pairs of corresponding full-length k-mers. If a pair of k-mers is within a Hamming distance τ, their components are joined. The components are implemented using a union-find data structure (see Cormen *et al.*, 2001 for details).

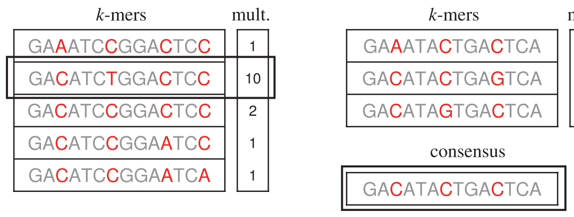

Fig. 1. On the left is an example of a typical cluster with good coverage. There are five k-mers clustered together, with five loci having mis-alignments. We compute the consensus string (taking multiplicities into account), which we find is already in the cluster (boxed in). All the k-mers are then corrected to the consenus. On the right is an example of a common cluster in low coverage regions. The generating k-mer was sequenced three times but each time with a single error. There are three k-mers in the cluster, but the consensus (boxed in) has not been sequenced and therefore is not in the cluster. Nevertheless, we correct all the k-mers to the consensus, allowing Hammer to reconstruct new k-mers that are not present in the original data.

This algorithm finds the connected components of the Hamming graph, which for simplicity we will refer to as *clusters*. It is similar to the algorithm of Reptile (Yang *et al.*, 2010), however, an important difference is that Reptile uses seeds that are consecutive blocks. It is known from sequence alignment (Ma *et al.*, 2002) that using spaced rather than consecutive seeds make it less likely for two k-mers to be equal. Because our index sets are spaced out across the whole range of k, the equivalence blocks should be smaller, allowing for a faster running time in practice.

Finally, we process the clusters one at a time. Let C be the current cluster. If the cluster does not have size one and the consensus string is unique, we correct every k-mer in the cluster to the consensus string (Fig. 1).

We refer to clusters which consist of only a single k-mer as *singletons*. This k-mer is trivially the consensus. It must either be the only element in the generating set of its generating k-mer, or it simply contains more than τ errors than the other members of the set. In another case, there are multiple consensus strings. which typically occurs in small clusters of size two or three. We refer to such clusters as *ambiguous*.

In both of these cases, the Hamming graph does not provide good information regarding the correctness of the k-mers of C. We therefore make a decision solely based on the multiplicity of each k-mer $x\in C$, weighed by the quality values (Kelley *et al.*, 2010). If it is greater than a threshold *singletonCutoff*, we keep x in the dataset; otherwise we remove it. The choice of *singletonCutoff* should be based on minimum coverage—if we expect certain regions to have low coverage, *singletonCutoff* should be around one (we use one in our single-cell experiment).

We also have a parameter *saveCutoff*, which makes sure that k-mers with a weighed multiplicity greater than or equal to *saveCutoff* are never removed, even if they are not the consensus of their cluster. This helps to address a shortcoming of our model, namely, that it supposes that there is only one correct k-mer in a cluster.

2.3 Time/space analysis and parameters

Hammer's running time and memory requirements are asymptotically (and in practice) dominated by the initial sorting of distinct k-mers. The sorting is done by storing the reads in memory and representing each k-mer as a pointer to its location in the reads. When n_r is the number of reads, ℓ is the read length, and $n_k=n_r(\ell-k+1)$ is the number of k-mers, the sort takes $O(n_k\log n_k)$ time and $O(n_k\log n_r)$ memory (here we treat ℓ and k as constants). There are more efficient alternatives for sorting k-mers, including map-reduce (Kelley *et al.*, 2010), hash functions and Graphical Processing Unit (GPU)-based methods (Shi *et al.*, 2010). However, for the purposes of the bacterial genomes we study in this article, we found that a sort sufficed.

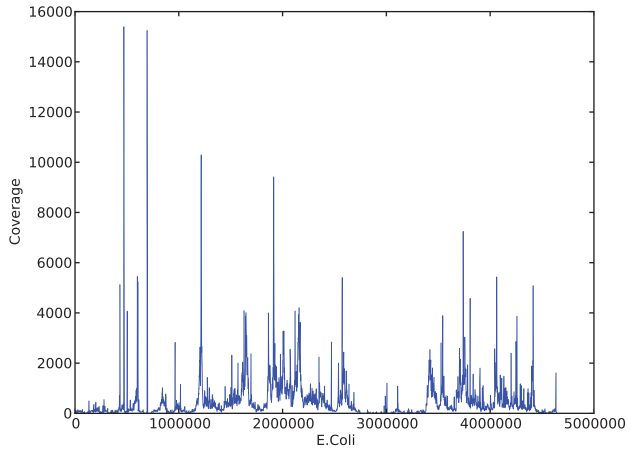

Fig. 2. The coverage of *E.coli* by single-cell reads. There are 210K positions with coverage by less than six reads, while the average coverage is 600.

After building the set of distinct k-mers K, we no longer need to deal with all the n_k k-mers, but instead only with the distinct k-mers and their multiplicities. The second step of sorting the $\tau+1$ copies of the k-mers therefore takes time $O(n_k^* \log n_k^*)$, where $n_k^* = |K|$ is the number of distinct k-mers in the dataset.

Finding the connected components requires us to maintain a union find structure of size $O(n_k^*)$. Scanning through the sorted sets requires an all-pair comparison within each equivalence block, which in worst case can take $O((n_k^*)^2)$ time; in practice, though, the size of the blocks is small. Finally, computing the consensuses of the clusters can be done in $O(n_k^*)$ time. Note that the steps of finding the connected components and their consensus are easily parallelizable, with almost no shared memory requirement.

The three parameters of Hammer are: τ, the joining distance for k-mers, *singletonCutoff*, the threshold for singleton and ambiguous clusters, *saveCutoff*, the threshold above which k-mers are never removed, and the k-mer size. The trade-offs with the k-mer size are similar to those in assembly. Too small of a k results in reads being chopped up too much, with important information about the genome being lost. However, setting k too large results in low k-mer coverage and lowers the accuracy of the algorithm. The value of τ should be large enough as to connect most of the generating set of a k-mer into one connected component. Varying the value of *singletonCutoff* or *saveCutoff* represents a direct trade-off between sensitivity and specificity.

3 RESULTS

To test the effectiveness of Hammer on non-uniform datasets, we used reads generated from a single-cell of *E.coli* K-12 strain, using one lane of the Illumina GAII pipeline (Chitsaz,H. *et al.*, 2011, submitted for publication). They total 600× coverage mostly by 100 bp long reads. We mapped the reads to the *E.coli* genome using bowtie (Langmead *et al.*, 2009) and found extremely non-uniform coverage (Fig. 2). There were 94K positions that are not covered by any read, with an additional 116K being covered by less than six reads. In contrast, in an alternate dataset that was generated on a normal multi-cell sample with the same coverage and read length, there were no uncovered regions.

Prior to working with this dataset, we trimmed the first three bases and any trailing bases with a quality value of 2; we then removed any remaining reads with ambiguous bases. We use a value of $k=55$ to run Hammer on this dataset, motivated by the fact that it was found to work best for assemblers on this data (Chitsaz,H. *et al.*, 2011, submitted for publication). The distance τ used in building the Hamming graph was set to two, since we found larger values did not significantly decrease the number of singleton clusters. For the *singletonCutoff*, we used the default value of 1, which, roughly speaking, means that any singleton k-mer that appears at least twice with decent quality values is kept. However, to demonstrate that Hammer can be made super-sensitive if needed, we also ran Hammer with a *singletonCutoff* of 0.5. For the *saveCutoff*, we used a value of 10.

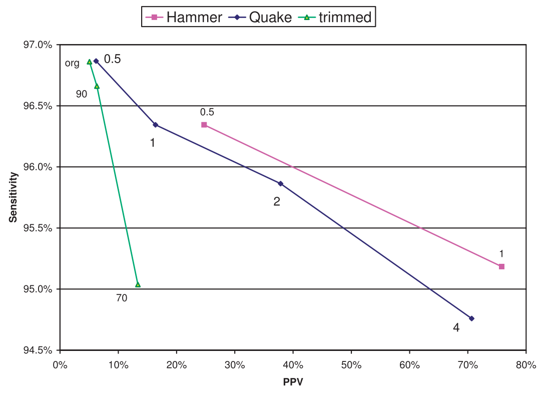

Fig. 3. Sensitivity and positive predictive values for single-cell reads. We show the results of varying the *singletonCutoff* in Hammer (0.5 and 1.0) and the cutoff in Quake (0.3–4.0). We also show the uncorrected dataset and the effect of simply trimming the reads down to 90 or 70 bases.

Table 1. Sensitivity and positive predictive values for our experiments

E.coli dataset	Method	Total distinct k-mers	Total correct k-mers	PPV (%)	Sensitivity (%)
single-cell	Uncorrected	87 579 268	4 422 073	5	96.86
	Hammer (cutoff = 1)	5 732 025	4 345 536	76	95.18
	Quake (cutoff = 4)	6 122 140	4 326 126	71	94.76
	Hammer (cutoff = 0.5)	17 790 512	4 398 487	25	96.34
	HiTEC (trimmed)	8 047 103	4 241 039	53	92.90
	Hammer (trimmed)	4 727 996	4 244 005	90	92.96
Normal	Uncorrected	77 421 925	4 564 319	6	99.98
	Hammer	4 787 644	4 545 158	95	99.56
	Quake	4 661 707	4 563 956	98	99.97

The results are shown in Figure 3 and Table 1. To measure the quality of a set of reads, we measure its sensitivity and positive predictive value (PPV) with respect to the source genome. Let K be the set of distinct k-mers present in the reads, and let S be the k-spectrum of *E.coli* (i.e. the set of all distinct k-mers). The sensitivity is measured as $\frac{|K \cap S|}{|S|}$ and reflects the percentage of k-mers from the genome that are present in the reads. The positive predictive value is $\frac{|K \cap S|}{|K|}$ and reflects the fraction of k-mers in the dataset that do not contain errors. Note that we do not measure the quality of the corrected reads against the quality of uncorrected reads directly, but by measuring the quality of each separately against the *E.coli* genome.

3.1 Comparison to other tools

We compared our results against Quake, which is a cutoff threshold-based method that was recently shown to have superior performance

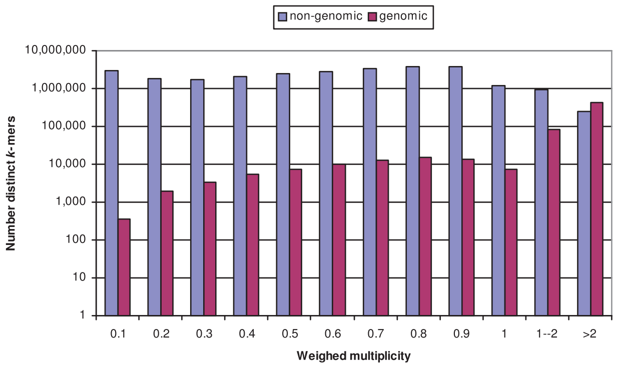

Fig. 4. The histogram showing the number of singletons with the given weighted multiplicity, on a vertical log scale.

Table 2. Cluster classification

Cluster category		Number of clusters	
Consensus accurate?	Num. correct *k*-mers in cluster	Single-cell *E.coli*	Normal *E.coli*
Yes	>1	12 717	13 056
Yes	1	3924 955	4 530 503
Yes	0	14 603	8427
No	>0	643	816
No	0	425 948	241 628

Classification of non-singleton and non-ambiguous clusters for both single-cell and normal data. Each cluster will have a consensus that is either correct or not, and will have a number of *k*-mers that were correct prior to correcion. We show the number of such clusters in Hammer's run on single-cell and on normal *E.coli*.

on normal (multi-cell) sequencing data (Kelley *et al.*, 2010). Quake works by modeling the weighted multiplicity histogram of *k*-mers as a combination of two separate distributions (one from erroneous reads, one from accurate reads), and using that model to determine the best cutoff threshold. It could not find any cutoff on the single-cell data, likely because the distribution model is based on uniformity assumptions that do not hold. However, we could manually try different values, and we plot the results in Figure 3 and show the raw numbers in Table 1. Hammer improves on Quake, retaining about 19K more genomic reads while reducing the number of erroneous reads by ∼390 K.

Though Quake was designed with uniformity in mind, an alternate method called HiTEC was recently published, which does not strongly rely on uniformity assumptions (Ilie *et al.*, 2010). HiTEC is based on a similar idea of using clusters to identify correct reads, albeit using suffix trees instead of the Hamming graph. Since HiTEC currently does not support ambiguous bases or variable length reads, in order to run it on our data we had to trim our reads down to 70 bp and remove all reads containing ambiguous bases. We then ran Hammer on the same altered dataset, with the results shown in Table 1. HiTEC sensitivity was nearly the same (a difference of 0.06%, or ∼3 K reads, in favor of Hammer), but its PPV was significantly lower (only 53% compared to 90% by Hammer).

3.2 Analysis of Results

Out of all the correct distinct *k*-mers present in the data, 10% were in singletons. The distribution of their weighted multiplicities is shown in Figure 4. Most of the incorrect *k*-mers have weights <1, however, there is also ∼53K correct *k*-mers that weigh between 0.5 and 1. This difference is apparent when we ran Hammer with a *singletonCutoff* of 0.5 (Table 1), allowing us to increase 1.2% in sensitivity.

We categorize each non-singleton and non-ambiguous cluster according to whether the consensus is correct (i.e. exists in *E.coli*) and the number of correct *k*-mers present in the cluster prior to correction (Table 2). In 90% of the clusters, the consensus is accurate, and in 99% of those cases the consensus was present and was the only correct *k*-mer in the cluster.

The second largest category of clusters (10%) was where no correct *k*-mer was present and the chosen consensus was also incorrect. When the consensus is not present in the cluster, it is usually very small. However, in 3% of cases where the consensus is not present, it turned out to be correct, representing an important gain in sensitivity. These cases allow us to reconstruct *k*-mers that were not present in the original genome. It remains an outstanding problem to find a way to accurately distinguish whether the consensus of a cluster is correct or not, when it is not present in the cluster.

Another category of interest is where the consensus was correct but there was more than one correct *k*-mer in the cluster. These cases occur when the genome has non-identical repeats, and hence there are correct *k*-mers with a small Hamming distance. Because this is a bacteria dataset, there were only 13K such cases, with a large fraction of them retained because of the *saveCutoff*. In eukaryotic genomes, however, this will represent a more serious problem, and thus extending Hammer for eukaryotic genomes is important continuing work.

3.3 Multi-cell normal data

We also tested Hammer on data generated from a normal multi-cell lane of *E.coli* with 600× coverage and 100 bp long reads (accession number ERX002508). We applied the same pre-filtering as for the single-cell data. The coverage of this dataset was close to uniform, with every single base covered by at least one read. As expected, Quake performed extremely well (Table 1), achieving 98% PPV and 99.98% sensitivity. Hammer did nearly as well, with 95% PPV and 99.56% sensitivity. There were still 242K consensus *k*-mers chosen when there were no correct *k*-mers in the cluster at all (Table 1). Again, it will be important to develop a way to detect these clusters without assuming uniformity. Though Hammer's results are comparable, Quake currently remains a better choice when it is known that the dataset is uniform.

3.4 Computational requirements

We used a single Intel Xeon 1.8 GHz machine with four dual-cores and 32 G of RAM. For our dataset, the run took approximately 4 h and needed 20 G of RAM. The longest step was the initial sorting of the *k*-mers, which took about 2 h.

4 DISCUSSION

Despite the success of many error-correction tools, emerging areas such as single-cell sequencing fuel the need to develop better algorithms for situations of non-uniform coverage. Recent approaches have taken steps in the right direction by looking together at all k-mers within a small Hamming neighborhood. Some rely on maximum likelihood estimation (Wijaya *et al.*, 2009), some on complex sets of rules (Yang *et al.*, 2010), and some on greedy local heuristics (Qu *et al.*, 2009). With the notable exception of Qu *et al.* (2009), these algorithms still rely on uniformity of coverage. Moreover, most of these algorithms make decisions about k-mers independent of the decisions made about other k-mers in the same cluster. In this article, we have developed an algorithm that makes a decision about all the k-mers with a cluster in a unified but simple manner, allowing us to handle non-uniform single-cell datasets.

In Hammer, we proposed the simple choice of the consensus string as a way to correct the k-mers within a cluster. However, this step is easily customizable with whatever choice may best suit the data, and we believe that this is a strength of our approach. In particular, one of Hammer's limitations is that it assumes that there is at most one correct k-mer in a cluster. Though this assumption results in relatively small losses in *E.coli*, and is furthermore offset by the use of the *saveCutoff*, it becomes a more serious problem in highly repetitive eukaryotic genomes. It is possible, however, to develop a more sophisticated approach that allows the identification of multiple correct k-mers within a cluster (Wijaya *et al.*, 2009). Another shortcoming is that Hammer chops the reads up into k-mers—while this is not be a problem for many downstream assemblers who do this anyway, it may be for other tools that require mapping. There are numerous existing techniques for reconstructing reads from the set of corrected k-mers (Chaisson and Pevzner, 2008; Kelley *et al.*, 2010; Yang *et al.*, 2010), and these may be incorporated into Hammer in the future. Newer platforms, such as the PacBio sequencer that has a large amount of indel errors, will also undoubtedly challenge Hammer's current approach; however, we believe that the idea of looking at all the k-mers within a single cluster jointly is a powerful one and will continue to be useful.

ACKNOWLEDGEMENTS

We would like to thank Hamidreza Chitsaz, Daniel Lokshtanov and Glenn Tesler for fruitful discussions and help during this project.

Funding: National Institutes of Health (grant 3P41RR024851-02S1, in part).

Conflict of Interest: none declared.

REFERENCES

Chaisson,M.J. and Pevzner,P.A. (2008) Short read fragment assembly of bacterial genomes. *Genome Res.*, **18**, 324–330.

Cormen,T.H. *et al.* (2001) *Introduction to Algorithms*, 2nd edn. The MIT Press/McGraw-Hill Book Company, pp. 505–509.

Genome 10K Community of Scientists (2009) Genome 10K: a proposal to obtain whole-genome sequence for 10 000 vertebrate species. *J. Heredity*, **100**, 659–674.

Gill,S.R. *et al.* (2006) Metagenomic analysis of the human distal gut microbiome. *Science*, **312**, 1355–1359.

Ilie,L. *et al.* (2010) HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics*, **27**, 295–302.

Jones,N.C. and Pevzner,P.A. (2004) *An Introduction to Bioinformatics Algorithms (Computational Molecular Biology)*. The MIT Press, p. 329.

Kelley,D. *et al.* (2010) Quake: quality-aware detection and correction of sequencing errors. *Genome Biol.*, **11**, R116.

Kvist,T. *et al.* (2007) Specific single-cell isolation and genomic amplification of uncultured microorganisms. *Appl. Microbiol. Biotechnol.*, **74**, 926–935.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Li,J. *et al.* (2010) Modeling non-uniformity in short-read rates in rna-seq data. *Genome Biol.*, **11**, R50.

Ma,B. *et al.* (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.

Margulies,M. *et al.* (2005) Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, **437**, 376–80.

Mussmann,M. *et al.* (2007) Insights into the genome of large sulfur bacteria revealed by analysis of single filaments. *PLoS Biol.*, **5**, e230.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.

Qu,W. *et al.* (2009) Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing. *Genome Res.*, **19**, 1309–1315.

Raghunathan,A., Jr *et al.* (2005) Genomic DNA amplification from a single bacterium. *Appl. Environ. Microbiol.*, **71**, 3342–3347.

Rodrigue,S. *et al.* (2009) Whole genome amplification and de novo assembly of single bacterial cells. *PLoS ONE*, **4**, e6864.

Salmela,L. (2010) Correction of sequencing errors in a mixed set of reads. *Bioinformatics*, **26**, 1284–1290.

Schroder,J. *et al.* (2009) SHREC: a short-read error correction method. *Bioinformatics*, **25**, 2157–2163.

Shi,H. *et al.* (2010) A parallel algorithm for error correction in high-throughput short-read data on cuda-enabled graphics hardware. *J. Comput. Biol.*, **17**, 603–615.

Wijaya,E. *et al.* (2009) Recount: expectation maximization based error correction tool for next generation sequencing data. *Genome Inform. Ser.*, **23**, 189–201.

Yang,X. *et al.* (2010) Reptile: representative tiling for short read error correction. *Bioinformatics*, **26**, 2526–2533.

Yooseph,S. *et al.* (2007) The *sorcerer ii* global ocean sampling expedition: Expanding the universe of protein families. *PLoS Biol.*, **5**, e16.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zhao,X. *et al.* (2010) Edar: an efficient error detection and removal algorithm for next generation sequencing data. *J. Comput. Biol.*, **17**, 1549–1560.