Scaling metagenome sequence assembly with probabilistic de Bruijn graphs

Jason Pell[a], Arend Hintze[a], Rosangela Canino-Koning[a], Adina Howe[b], James M. Tiedje[b,c], and C. Titus Brown[a,b,1]

[a]Computer Science and Engineering, Michigan State University, East Lansing, MI 48824; [b]Microbiology and Molecular Genetics, Michigan State University, East Lansing, MI 48824; and [c]Crop and Soil Sciences, Michigan State University, East Lansing, MI 48824

Edited by Dan Gusfield, UC Davis, and accepted by the Editorial Board June 26, 2012 (received for review December 27, 2011)

Deep sequencing has enabled the investigation of a wide range of environmental microbial ecosystems, but the high memory requirements for de novo assembly of short-read shotgun sequencing data from these complex populations are an increasingly large practical barrier. Here we introduce a memory-efficient graph representation with which we can analyze the *k*-mer connectivity of metagenomic samples. The graph representation is based on a probabilistic data structure, a Bloom filter, that allows us to efficiently store assembly graphs in as little as 4 bits per *k*-mer, albeit inexactly. We show that this data structure accurately represents DNA assembly graphs in low memory. We apply this data structure to the problem of partitioning assembly graphs into components as a prelude to assembly, and show that this reduces the overall memory requirements for de novo assembly of metagenomes. On one soil metagenome assembly, this approach achieves a nearly 40-fold decrease in the maximum memory requirements for assembly. This probabilistic graph representation is a significant theoretical advance in storing assembly graphs and also yields immediate leverage on metagenomic assembly.

metagenomics | compression

De novo assembly of shotgun sequencing reads into longer contiguous sequences plays an important role in virtually all genomic research (1). However, current computational methods for sequence assembly do not scale well to the volume of sequencing data now readily available from next-generation sequencing machines (1, 2). In particular, the deep sequencing required to sample complex microbial environments easily results in datasets that surpass the working memory of available computers (3, 4).

Deep sequencing and assembly of short reads is particularly important for the sequencing and analysis of complex microbial ecosystems, which can contain millions of different microbial species (5, 6). These ecosystems mediate important biogeochemical processes but are still poorly understood at a molecular level, in large part because they consist of many microbes that cannot be cultured or studied individually in the lab (5, 7). Ensemble sequencing ("metagenomics") of these complex environments is one of the few ways to render them accessible, and has resulted in substantial early progress in understanding the microbial composition and function of the ocean, human gut, cow rumen, and permafrost soil (3, 4, 8, 9). However, as sequencing capacity grows, the assembly of sequences from these complex samples has become increasingly computationally challenging. Current methods for short-read assembly rely on inexact data reduction in which reads from low-abundance organisms are discarded, biasing analyses towards high-abundance organisms (3, 4, 9).

The predominant assembly formalism applied to short-read sequencing datasets is a de Bruijn graph (10–12). In a de Bruijn graph approach, sequencing reads are decomposed into fixed-length words, or *k*-mers, and used to build a connectivity graph. This graph is then traversed to determine contiguous sequences (12). Because de Bruijn graphs store only *k*-mers, memory usage scales with the number of unique *k*-mers in the dataset rather than the number of reads (12, 13). Thus human genomes can be assembled in less than 512 GB of system memory (14). For more complex samples such as soil metagenomes, which may possess millions or more species, terabytes of memory would be required to store the graph. Moreover, the wide variation in species abundance limits the utility of standard memory-reduction practices such as abundance-based error-correction (15).

In this work, we describe a simple probabilistic representation for storing de Bruijn graphs in memory, based on Bloom filters (16). Bloom filters are fixed-memory probabilistic data structures for storing sparse sets; essentially hash tables without collision detection, set membership queries on Bloom filters can yield false positives but not false negatives. Although, Bloom filters have been used in bioinformatics software tools in the past, they have not been used for storing assembly graphs (17–20). We show that this probabilistic graph representation more efficiently stores de Bruijn graphs than any possible exact representation for a wide range of useful parameters. We also demonstrate that it can be used to store and traverse actual DNA de Bruijn graphs with a 20- to 40-fold decrease in memory usage over two common de Bruijn graph-based assemblers, Velvet and ABySS (21, 22). We relate changes in local and global graph connectivity to the false positive rate of the underlying Bloom filters and show that the graph's global structure is accurate for false positive rates of 15% or lower, corresponding to a lower memory limit of approximately 4 bits per graph node.

We apply this graph representation to reduce the memory needed to assemble a soil metagenome sample, through the use of read partitioning. Partitioning separates a de Bruijn graph into disconnected graph components; these components can be used to subdivide sequencing reads into disconnected subsets that can be assembled separately. This exploits a convenient biological feature of metagenomic samples: They contain many microbes that should not assemble together. Graph partitioning has been used to improve the quality of metagenome and transcriptome assemblies by adapting assembly parameters to local coverage of the graph (23–25). However, to our knowledge, partitioning has not been applied to scaling metagenome assembly. By applying the probabilistic de Bruijn graph representation to the problem of partitioning, we achieve a dramatic decrease of nearly 40-fold in the memory required for assembly of a soil metagenome.

Results

Bloom Filters Can Store de Bruijn Graphs. Given a set of short DNA sequences, or reads, we first break down each read into a set of

Author contributions: J.P., A. Hintze, and C.T.B. designed research; J.P., A. Hintze, and C.T.B. performed research; J.P., A. Hintze, R.C.-K., A. Howe, and J.M.T. contributed new reagents/analytic tools; J.P., A. Hintze, R.C.-K., and A. Howe analyzed data; and J.P., A. Hintze, and C.T.B. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission. D.G. is a guest editor invited by the Editorial Board.

Freely available online through the PNAS open access option.

Data deposition: The sequence reported in this paper has been deposited in the GenBank database [accession no. SRX128885 (MSB2 soil data)].

[1]To whom correspondence should be addressed. E-mail: ctb@msu.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1121464109/-/DCSupplemental.

overlapping k-mers. We then store each k-mer in a Bloom filter, a probabilistic data structure for storing elements from sparse datasets (see *Methods* for implementation details). Each k-mer serves as a vertex in a graph, with an edge between two vertices N_1 and N_2 if and only if N_1 and N_2 share a (k-1)-mer that is a prefix of N_1 and a postfix of N_2, or vice versa. This edge is not stored explicitly, which can lead to false connections when two reads abut but do not overlap; these false connections manifest as false positives, discussed in detail below.

Thus each k-mer has up to eight edges connecting to eight neighboring k-mers, which can be determined by simply building all possible 1-base extensions and testing for their presence in the Bloom filter. In doing so, we implicitly treat the graph as a simple graph as opposed to a multigraph, which means that there can be no self-loops or parallel edges between vertices/k-mers. By relying on Bloom filters, the size of the data structure is fixed: No extra memory is used as additional data are added.

This graph structure is effectively compressible because one can choose a larger or smaller size for the underlying Bloom filters; for a fixed number of entries, a larger Bloom filter has lower occupancy and produces correspondingly fewer false positives, whereas a smaller Bloom filter has higher occupancy and produces more false positives. In exchange for memory, we can store k-mer nodes more or less accurately: for example, for a false positive rate of 15%, at which one in six random k-mers tested would be falsely considered present, each real k-mer can be stored in under 4 bits of memory (see Table 1). Although there are many false k-mers, they only matter if they connect to a real k-mer.

The false positive rate inherent in Bloom filters thus raises one concern for graph storage: In contrast to an exact graph storage, there is a chance that a k-mer will be adjacent to a false positive k-mer. That is, a k-mer may connect to another k-mer that does not actually exist in the original dataset but nonetheless registers as present, due to the probabilistic nature of the Bloom filter. As the memory per real k-mer is decreased, false positive vertices and edges are gained, so compressing the graph results in a more tightly interconnected graph. If the false positive rate is too high, the graph structure will be dominated by false connectivity—but what rate is "too high"? We study this key question in detail below.

False Positives Cause Local Elaboration of Graph Structure. Erroneous neighbors created by false positives can alter the graph structure. To better understand this effect, we generated a random 1,031 bp circular sequence and visualized the effect of different false positive rates. After storing this single sequence in compressible graphs using $k = 31$ with four different false positive rates ($p_f = 0.01$, 0.05, 0.10, and 0.15), we explored the graph using breadth-first search beginning at the first 31-mer. The graphs in Fig. 1 illustrate how the false positive k-mers connected to the original k-mers (from the 1,031 bp sequence) elaborate with the false positive rate while the overall circular graph structure remains, with no erroneous shortcuts between k-mers that are present in the original sequence. It is visually apparent that even a high false positive rate of 15% does not systematically and erroneously connect distant k-mers.

Table 1. Bits per *k*-mer for various false positive rates

False positive rate	Bits/*k*-mer
0.1%	14.35
1%	9.54
5%	6.22
10%	4.78
15%	3.94
20%	3.34

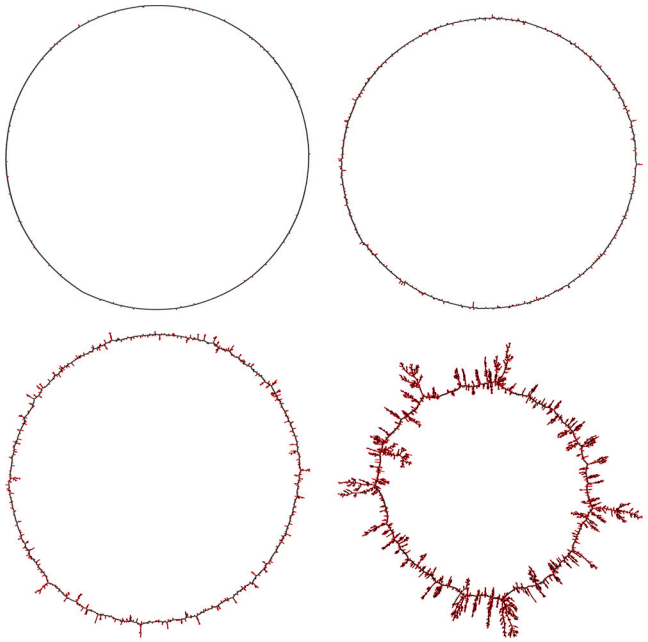

Fig. 1. Graph visualizations demonstrating the decreasing fidelity of graph structure with increasing false positive rate. Erroneous k-mers are colored red and k-mers corresponding to the original generated sequence (1,000 31-mers generated by a 1,031 bp circular chromosome) are black. From top left to bottom right, the false positive rates are 0.01, 0.05, 0.10, and 0.15. Shortcuts "across" the graph are not created.

False Long-Range Connectivity is a Nonlinear Function of the False Positive Rate. To explore the point at which our data structure systematically engenders false long-range connections, we inserted random k-mers into Bloom filters with increasing false positive rates. These k-mers connect to other k-mers to form graph components that increase in size with the false positive rate. We then calculated the average component size in the graph for each false positive rate ($n = 10{,}000$) and used percentile bootstrap to obtain estimates within a 95% confidence interval. Fig. 2 demonstrates that the average component size rapidly increases as a specific threshold is approached, which appears to be at a false positive rate near 0.18 for $k = 31$. Beyond 0.18, the components begin to join together into larger components.

As the false positive rate increases, we observe a sudden transition from many small components to fewer, larger components created by erroneous connections between the "true" components (Fig. 2). In contrast to the linear increase in the local neighborhood structure as the false positive rate increases linearly, the change in global graph structure is abrupt as previously disconnected components join together. This rapid change resembles a geometric phase transition, which for graphs can be discussed in terms of percolation theory (26). We can map our problem to site percolation by considering a probability p that a particular k-mer is present, or "on". (This is in contrast to bond percolation where p represents the probability of a particular edge being present.) As long as the false positive rate is below the percolation threshold p_θ (i.e., in the subcritical phase), we would predict that the graph is not highly connected by false positives.

Percolation thresholds for finite graphs can be estimated by finding where the component size distribution transitions from linear to quadratic in form (27). Using the calculation method described in *Methods*, we found the site percolation threshold for DNA de Bruijn graphs to be $p_\theta = 0.183 \pm 0.001$ for k between 5 and 12. Although we only tested within this limited range of k, the percolation threshold appears to be independent of different k (see Fig. S1). Thus, as long as the false positive rate is below 0.183, we predict that truly disconnected components in

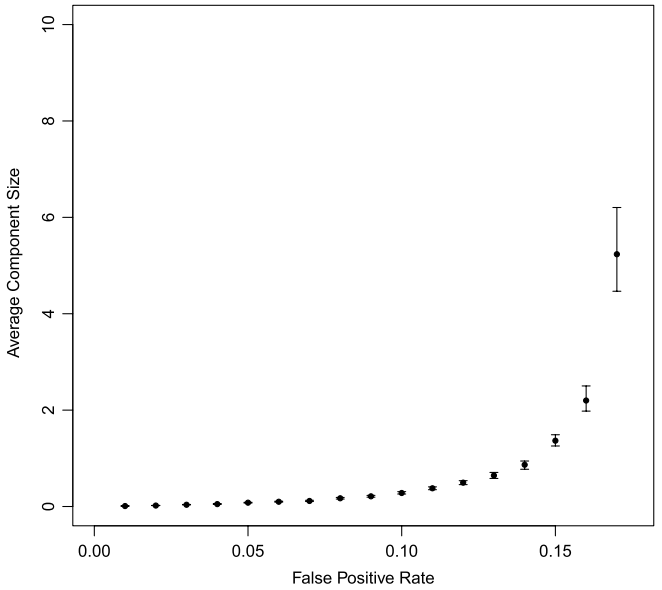

Fig. 2. Average component size versus false positive rate. The average component size sharply increases as the false positive rate approaches the percolation threshold.

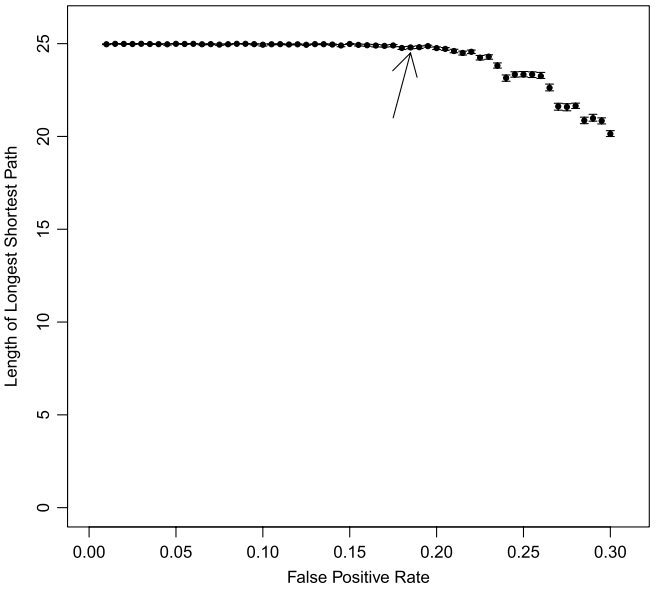

Fig. 3. The diameter of randomly generated 58 bp long circular chromosomes in 8-mer (i.e., a cycle of 50 8-mers) space remains constant for false positive rates up through 18.3%. Only real (nonerror) *k*-mers are considered for starting and ending points.

the graph are unlikely to connect to one another erroneously, that is, due to errors introduced by the probabilistic representation.

Large-Scale Graph Structure is Retained up to the Percolation Threshold. The results from component size analysis and the percolation threshold estimation suggest that global connectivity of the graph is unlikely to change below a false positive rate of 18%. Do we see this invariance in global connectivity in other graph measures?

To assess global connectivity, we employed the diameter metric in graph theory, the length of the "longest shortest" path between any two vertices (28). If shorter paths between real *k*-mers were being systematically generated due to false positives, we would expect the diameter of components to decrease as the false positive rate increased. We randomly generated 58 bp long circular chromosomes (50 bp read with the first 8-mer appended to the end of the string) to construct components containing 50 8-mers and calculated the diameter at different false positive rates. We kept *k* low because we needed to be able to exhaustively explore the graph even beyond the percolation threshold, which is computationally prohibitive to do at higher *k* values. Furthermore, larger circular chromosomes would be more likely to erroneously connect at a fixed *k*, but due to the relatively low number of possible 8-mers, we had to keep the chromosomes small. We only considered paths between two real *k*-mers in the dataset.

At each false positive rate, we ran the simulation 500 times and estimated the mean within a 95% confidence interval using percentile bootstrap. As Fig. 3 shows, erroneous connections between pairs of real *k*-mers are rare below a false positive rate of 20%. For false positive rates above this threshold, spurious connections between real *k*-mers are created, which lowers the diameter. Thus, the larger scale graph structure is retained up through $p = 0.183$, as suggested by the component size analysis and percolation results. This demonstrates that as long as the *k*-mer space is only sparsely occupied by false positives, long "bridges" between distant *k*-mers do not appear spontaneously.

Erroneous *k*-mers From Sequencing Eclipse Graph False Positives. It is important to compare the errors from false positives in the de Bruijn graph representation with errors from real data. In particular, real data from massively parallel sequencers will contain base calling errors. In de Bruijn graph-based assemblers, these sequencing errors add to the graph complexity and make it more difficult to find high-quality paths for generating long, accurate contigs. Because our approach also generates false positives, we wanted to compare the error rate from the Bloom filter graph with experimental errors generated by sequencing (Table 2). We used the *Escherchia coli* K-12 MG1665 genome to compare various graph invariants between an Illumina dataset generated from the same strain (see *Methods*), an exact representation of the genome, and inexact representations with different false positive rates.

For these comparisons, we used a *k* value of 17, for which we can store graphs exactly, i.e., we have no false positives because we can store 4^{17} entries precisely in 2 GB of system memory. This is equivalent to a Bloom filter with one hash table and a 0% false positive rate. We found a total of 50,605 17-mers in the exact representation that were not part of a simple line, i.e., had more than two neighbors (degree > 2). As the false positive rate increased, the number of these 17-mers increased in the expected

Table 2. Effects of loading *E. coli* data at different false positive rates

Graph	Total *k*-mers	False connected *k*-mers	% Real	Deg > 2	Mem (bytes)
E. coli at 0%	4,530,123	0	100	50,605	2.1×10^9
E. coli at 1%	4,814,050	283,927	94.1	313,844	5.4×10^6
E. coli at 5%	6,349,301	1,819,178	71.3	1,339,102	3.5×10^6
E. coli at 15%	31,109,523	26,579,400	14.6	10,522,482	2.2×10^6
Reads at 0%	45,566,033	41,036,029	9.9	7,699,309	2.1×10^9
Reads at 1%	48,182,271	43,652,265	9.4	31,600,208	5.4×10^7
Reads at 5%	62,019,545	57,489,537	7.3	42,642,203	3.6×10^7
Reads at 15%	231,687,063	227,157,037	1.9	113,489,902	2.3×10^7

linear fashion. Furthermore, the number of real 17-mers, those that are not false positives, comprise the majority of the graph. (As above, we only counted false positive k-mers that are transitively connected to at least one real k-mer.)

In contrast, when we examined an exact representation of an Illumina dataset, only 9.9% of the k-mers in the graph truly exist in the reference genome. The number of 17-mers with more than two neighbors in the sequencing reads is higher than for the exact representation of the genome, which demonstrates that sequencing errors add to the complexity of the graph. Overall, the errors demonstrated by sequencers dwarf the errors caused by the inexact graph representation at a reasonable false positive rate.

When we assemble this dataset with the Velvet and ABySS assemblers at $k = 31$, Velvet requires 3.7 GB to assemble the dataset, whereas ABySS requires 1.6 GB; this memory usage is dominated by the graph storage (29). Thus the Bloom filter approach stores graphs 30 or more times more efficiently than either program, even with a low false positive rate of 1%. Although this direct comparison cannot be made fairly—assemblers store the graph as well as k-mer abundances and other information—it does suggest that there are opportunities for decreasing memory usage with the probabilistic graph representation.

Sequences Can Be Accurately Partitioned by Graph Connectivity. Can we use this low-memory graph representation to find and separate components in de Bruijn graphs? The primary concern is that false positive nodes or edges would connect components, but the diameter results suggest that components are unlikely to connect below a 20% false positive rate. To verify this, we analyzed a simulated dataset of 1,000 randomly generated sequences of length 10,000 bp. Using $k = 31$, we partitioned the data across many different false positive rates, using the procedure described in *Methods*. As predicted, the resulting number of partitions did not vary across the false positive rates while $f_p \leq 0.15$ (Fig. S2).

We then applied partitioning to a considerably larger bulk soil metagenome ("MSB2") containing 35 million 75 bp long reads generated from an Illumina GAII sequencer. We calculated the number of unique 31-mers present in the dataset to be 1.35 billion. Then, for each of several false positive rates (see Table 3) we loaded the reads into a graph, eliminated components containing fewer than 200 unique k-mers, and partitioned the reads into separate files based on graph connectivity.

Once we obtained the partition sets, we individually assembled each set of partitions using ABySS, as well as the entire (unpartitioned) dataset, retaining contigs longer than 500 bp. The resulting assemblies were all identical, containing 1,444 contigs with a total assembled sequence of 1.07 megabases. The unpartitioned dataset required 33 GB to assemble with ABySS, whereas the dataset could be partitioned in under 1 GB with a 30-fold decrease in maximum memory usage (Table 3). Moreover, despite this dramatic decrease in the memory required to assemble the dataset, the assembly results are identical.

Discussion

Bloom Filters Can Be Used to Efficiently Store de Bruijn Graphs. The use of Bloom filters to store a de Bruijn graph is straightforward and memory efficient. The expected false positive rate can be tuned based on desired memory usage, yielding a wide range of possible storage efficiencies (Table 1). Because memory usage is k independent in Bloom filters, it is more efficient than the theoretical lower-bound for a lossless exact representation when the number of k-mers inserted in the graph is sparsely populated, which is dependent on k (Fig. 4; see (13) for details on lower-bound memory usage for an exact representation).

Even for low false positive rates such as 1%, this is still an efficient graph representation, with significant improvements in both theoretical memory usage (Fig. 4) and actual memory usage compared to two existing assemblers, Velvet and ABySS (Table 2). We can store k-mers in this data structure with a much smaller set of "erroneous" k-mers than those generated by sequencing errors, and the Bloom filter false positive rates have less of an effect on branching graph structure than do sequencing errors. In addition, the false positives engendered by the Bloom filters are uncorrelated with the original sequence, unlike single-base sequencing errors that resemble the real sequence.

Using a probabilistic graph representation with false positive nodes and edges raises the specter of systematic graph artifacts resulting from the false positives. For partitioning, our primary concern was that false positives would incorrectly connect components, rendering partitioning ineffective. The results from percolation analysis, diameter calculations, and partitioning of simulated and real data demonstrate that below the calculated percolation threshold there is no significant false connectivity. As long as the false positive rate is below 18.3%, long false paths are not spontaneously created and the large scale graph properties do not change. Above this rate, the global graph structure quickly degrades.

Partitioning Works on Real Datasets. Our partitioning results on a real soil metagenome, the MSB2 dataset, demonstrate the utility of partitioning for reducing memory usage. For this specific dataset, we obtained identical results with a 20–40× decrease in memory (Table 3). This is consonant with our results from storing the *E. coli* genome, in which we achieved a 30-fold decrease in memory usage over the exact representation at a false positive rate of 1%. Although increased coverage and variation in dataset complexity will affect actual memory usage for other datasets, these results demonstrate that significant scaling in the memory required for assembly can be achieved in one real case.

The memory requirements for the partitioning process on the MSB2 dataset are dominated by the memory required to store and explore the graph; the higher memory usage observed for

Table 3. Partitioning results on a soil metagenome at $k = 31$

False positive rate (%)	Total memory use (improvement)	Largest partition size in reads
1	1.75 GB (18.8×)	344,426
5	1.20 GB (27.5×)	344,426
10	0.96 GB (34.37×)	344,426
15	0.83 GB (39.75×)	344,426

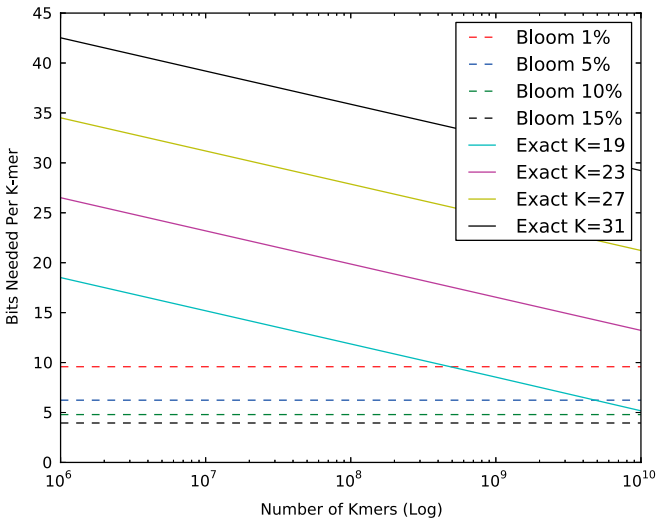

Fig. 4. Comparison between Bloom filters at different false positive rates with the information-theoretic lossless lower bound at different k values. Bloom filters are k independent and are more efficient than any lossless data structure for higher k due to greater sparseness in k-mers inserted compared to all possible k-mers.

partitioning at a false positive rate of 15% is due to the increase in component size from local false positives. Regardless, the memory requirements for downstream assembly of partitions is driven by the size of the largest partition, which here is very small (345,000 reads; Table 3) due to the high diversity of soil and the concordant low coverage. The dominant partition size is remarkably refractory to the graph's false positive rate, increasing by far less than 1% for a 15-fold increase in false positives; this shows that our theoretical and simulated results for component size and diameter apply to the MSB2 dataset as well.

Once partitioned, components can be assembled with parameters chosen for the coverage and sequence heterogeneity present in each partition. Moreover, datasets partitioned at a low k_0 can be exactly assembled with any $k \geq k_0$, because overlaps of $k_0 - 1$ bases include all overlaps of greater length. Because the partitions will generally be much smaller than the total dataset (see Table 3), they can be quickly assembled with many different parameters. This ability to quickly explore many parameters could result in significant improvement in exploratory metagenome assembly, where the "best" assembly parameters are not known and must be determined empirically based on many different assemblies.

Combined with the scaling properties of the graph representation, partitioning with this probabilistic de Bruijn graph representation offers a way to efficiently apply a partitioning strategy to certain assembly problems. Although this work focuses on theoretical properties of the graph representation and analyzes only one real dataset, the results are promising; the next step is to evaluate the approach on many more real datasets.

Concluding Remarks

Developing efficient and accurate approaches to de novo assembly continues to be one of the "grand challenges" of bioinformatics (2). Improved metagenome assembly approaches are particularly important for the investigation of microbial life on Earth, much of which has not been sampled (7, 30). Although our appreciation for the role that microbes play in biogeochemical processes is only growing, we are increasingly limited by our ability to analyze the data. For example, the Earth Microbiome Project is generating petabytes of sequencing data from complex microbial communities, many of which contain entirely novel ensembles of microbes; scaling de novo assembly is a critical requirement of this investigation (31).

The probabilistic de Bruijn graph representation presented here has a number of convenient features for storing and analyzing large assembly graphs. First, it is efficient and compressible: For a given dataset, a wide range of false positive rates can be chosen without impacting the global structure of the graph, allowing graph storage in as little as 4 bits per k-mer. Because a higher false positive rate yields a more elaborate local structure, memory can be traded for traversal time in, e.g., partitioning. Second, it is a fixed-memory data structure, with predictable degradation of both local and global structure as more data are inserted. For datasets where the number of unique k-mers is not known in advance, the occupancy of the Bloom filter can be monitored as data are inserted and directly converted to an expected false positive rate. Third, the memory usage is independent of the k-mer size chosen, making this representation convenient for exploring properties at many different parameters. It also allows the storage and traversal of de Bruijn graphs at multiple k-mer sizes within a single structure, although we have not yet explored these properties. And fourth, it supports memory-efficient partitioning, an approach that exploits underlying biological features of the data to divide the dataset into disconnected subsets.

Our initial motivation for developing this use of Bloom filters was to explore partitioning as an approach to scaling metagenome assembly, but there are many additional uses beyond metagenomics. Here we describe exact partitioning of the graph into components, but inexact partitioning has been successfully applied to mRNAseq assembly (25). Inexact partitioning, as done by the Chrysalis component of the Trinity pipeline, uses heuristics to subdivide the graph for later assembly; the data structure described in this work can be used for this purpose as well. More broadly, a more memory efficient de Bruijn graph representation opens up many additional opportunities. Although de Bruijn graph approaches are currently being used primarily for the purposes of assembly, they are a generally useful formalism for sequence analysis. In particular, they have been extended to efficient multiple-sequence alignment, repeat discovery, and detection of local and structural sequence variation (29, 32–34).

Materials

Genome and Sequence Data. We used the *E. coli* K-12 MG1655 genome (GenBank: U00096.2) and two MG1655 Illumina datasets [Short Read Archive (SRA) accessions SRX000429 and SRX000430] for *E. coli* analyses. The MSB2 soil dataset is available as SRA accession SRA050710.1.

Data Structure Implementation. We implemented a variation on the Bloom filter data structure to store k-mers in memory. In a classic Bloom filter, multiple hash functions map bits into a single hash table to add an object or test for the presence of an object in the set. In our variant, we use multiple prime-number-sized hash tables, each with a different hash function corresponding to the modulus of the DNA bitstring representation with the hash table size; this is a computationally convenient way to construct hash functions for DNA strings. The properties of this implementation are identical to a classical Bloom filter (35).

Estimating False Positive Rate for Erroneous Connectivity. We ran a simulation to find when components in the graph begin to erroneously connect to one another. To calculate the false positive rate p at which this aberrant connectivity occurs, we added random k-mers, sampled from a uniform GC distribution to the data structure and then calculated the occupancy and size of the largest component. From this we sampled the relative size of the largest component and the overall component size distribution for each given occupancy rate. At the occupancy where a "giant component" appears, this component size distribution should be scale-free (27). We then found at what value of p the resulting component size distribution in logarithmic scale can be better fitted in a linear or quadratic fashion using the F-statistic

$$F = \frac{RSS_1 - RSS_2}{p_2 - p_1} \times \frac{n - p_2}{RSS_2},$$

where RSS_i is the residual sum of squares for model i, p_i is the number of parameters for model i, and n is the number of data points. To handle the finite size sampling error, the data were binned using the threshold binning method (36). The critical value for when aberrant connectivity occurred was found by determining the local maxima of the F-values (37).

Graph Partitioning Using a Bloom Filter. We used the Bloom filter data structure containing the k-mers from a dataset to discover components of the graph, i.e., to partition the graph. Here a component is a set of k-mers whose originating reads overlap transitively by at least $k - 1$ base pairs. Reads belonging only to small components can be discovered and eliminated in fixed memory using a simple traversal algorithm that truncates after discovering more than a given number of novel k-mers. For discovering large components we tag the graph at a minimum density by using the underlying reads as a guide. We then exhaustively explore the graph around these tags in order to connect tagged k-mers based on graph connectivity. The underlying reads in each component can then be separated based on their partition.

Assembler Software. We used ABySS v1.3.1 and Velvet v1.1.07 to perform assemblies. The ABySS command was: `mpirun -np 8 ABYSS-P -k31 -o contigs.fa reads.fa`. The Velvet commands were: `velveth assem 31 -fasta -short reads.fa && velvetg assem`. We did not use Velvet for the partitioning analysis because Velvet's error correction algorithm is stochastic and results in dissimilar assemblies for different read order.

Software and Software Availability. We have implemented this compressible graph representation and the associated partitioning algorithm in a software package named khmer. It is written in C++ and Python 2.6 and is available

under the Berkeley Software Distribution open source license at https://github.com/ged-lab/khmer. The graphviz software package was used for graph visualizations. The scripts to generate the figures of this paper are available in the khmer repository.

ACKNOWLEDGMENTS. We thank Chris Adami, Qingpeng Zhang, and Tracy Teal for thoughtful comments and Jim Cole and Jordan Fish for discussion of future applications. In addition, we thank three anonymous reviewers for their comments, which substantially improved the paper. This project was supported by Agriculture and Food Research Initiative Competitive Grant no. 2010-65205-20361 from the United States Department of Agriculture, National Institute of Food and Agriculture and National Science Foundation IOS-0923812, both to C.T.B. The MSB2 soil metagenome was sequenced by the Department of Energy's Joint Genome Institute through the Great Lakes Bioenergy Research Center (DOE BER DE-FC02-07ER64494). A.H. was supported by NSF Postdoctoral Fellowship Award #0905961.

1. Pop M (2009) Genome assembly reborn: Recent computational challenges. *Brief Bioinform* 10:354–366.
2. Salzberg S, et al. (2012) GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Res* 22:557–567.
3. Qin J, et al. (2010) A human gut microbial gene catalogue established by metagenomic sequencing. *Nature* 464:59–65.
4. Hess M, et al. (2011) Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science* 331:463–467.
5. Wooley J, Godzik A, Friedberg I (2010) A primer on metagenomics. *PLoS Comput Biol* 6:e1000667.
6. Gans J, Wolinsky M, Dunbar J (2005) Computational improvements reveal great bacterial diversity and high metal toxicity in soil. *Science* 309:1387–1390.
7. Committee on Metagenomics and Functional Applications (2007) *The New Science of Metagenomics: Revealing the Secrets of Our Microbial Planet* (National Research Council (US), National Academy Press, Washington, DC).
8. Venter J, et al. (2004) Environmental genome shotgun sequencing of the Sargasso Sea. *Science* 304:66–74.
9. Mackelprang R, et al. (2011) Metagenomic analysis of a permafrost microbial community reveals a rapid response to thaw. *Nature* 480:368–371.
10. Pevzner P, Tang H, Waterman M (2001) An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci USA* 98:9748–9753.
11. Miller J, Koren S, Sutton G (2010) Assembly algorithms for next-generation sequencing data. *Genomics* 95:315–327.
12. Compeau P, Pevzner P, Tesler G (2011) How to apply de Bruijn graphs to genome assembly. *Nat Biotechnol* 29:987–991.
13. Conway TC, Bromage AJ (2011) Succinct data structures for assembling large genomes. *Bioinformatics* 27:479–486.
14. Gnerre S, et al. (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci USA* 108:1513–1518.
15. Kelley D, Schatz M, Salzberg S (2010) Quake: Quality-aware detection and correction of sequencing errors. *Genome Biol* 11:R116.
16. Bloom B (1970) Space/time tradeoffs in hash coding with allowable errors. *CACM* 13:422–426.
17. Shi H (2010) A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. *J Comput Biol* 17:603–615.
18. Stranneheim H (2010) Classification of DNA sequences using Bloom filters. *Bioinformatics* 26:1595–1600.
19. Malsted P (2011) Efficient counting of *k*-mers in DNA sequences using a bloom filter. *BMC Bioinformatics* 12:333.
20. Liu Y (2011) DecGPU: Distributed error correction on massively parallel graphics processing units using CUDA and MPI. *BMC Bioinformatics* 12:85.
21. Zerbino DR, Birney E (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* 18:821–829.
22. Simpson JT, et al. (2009) ABySS: A parallel assembler for short read sequence data. *Genome Res* 19:1117–1123.
23. Namiki T, Hachiya T, Tanaka H, Sakakibara Y (2011) MetaVelvet: An extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *ACM Conference on Bioinformatics, Computational Biology and Biomedicine*.
24. Peng Y, Leung H, Yiu S, Chin F (2011) Meta-IDBA: A de Novo assembler for metagenomic data. *Bioinformatics* 27:i94–i101.
25. Grabherr M, et al. (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol* 29:644–652.
26. Stauffer D, Aharony A (1994) *Introduction to Percolation Theory* (Taylor and Frances, London).
27. Stauffer D (1979) Scaling theory of percolation clusters. *Phys Rep* 54:1–74.
28. Bondy J, Murty U (2006) *Graph Theory. Graduate Texts in Mathematics* (Springer, New York).
29. Zerbino DR (2009) Genome assembly and comparison using de Bruijn graphs. PhD thesis (Univ of Cambridge, Cambridge, UK).
30. Gilbert J, et al. (2010) Meeting report: The terabase metagenomics workshop and the vision of an earth microbiome project. *Stand Genomic Sci* 3:243–248.
31. Gilbert J, et al. (2010) The Earth microbiome project: Meeting report of the "1 EMP meeting on sample selection and acquisition" at Argonne National Laboratory October 6 2010. *Stand Genomic Sci* 3:249–253.
32. Zhang Y, Waterman M (2003) DNA sequence assembly and multiple sequence alignment by an Eulerian path approach. *Cold Spring Harbor Symposia on Quantitative Biology,* (Cold Spring Harbor Lab Press, Cold Spring Harbor, NY), Vol 68, pp 205–212.
33. Price A, Jones N, Pevzner P (2005) De novo identification of repeat families in large genomes. *Bioinformatics* 21:i351–i358.
34. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G (2012) De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat Genet* 44:226–232.
35. Broder A, Mitzenmacher M (2004) Network applications of bloom filters: A survey. *Internet Math* 1:485–509.
36. Adami C, Chu J (2002) Critical and near-critical branching processes. *Phys Rev E* 66:011907.
37. Wald A (1943) Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Trans Am Math Soc* 54:426–482.