Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads

Sergey Nurk[1,⋆], Anton Bankevich[1,⋆], Dmitry Antipov[1], Alexey Gurevich[1], Anton Korobeynikov[1,2], Alla Lapidus[1,3], Andrey Prjibelsky[1], Alexey Pyshkin[1], Alexander Sirotkin[1], Yakov Sirotkin[1], Ramunas Stepanauskas[4], Jeffrey McLean[5], Roger Lasken[5], Scott R. Clingenpeel[6], Tanja Woyke[6], Glenn Tesler[7], Max A. Alekseyev[8], and Pavel A. Pevzner[1,9]

[1] Algorithmic Biology Laboratory, St. Petersburg Academic University, Russian Academy of Sciences, St. Petersburg, Russia
[2] Dept. of Mathematics and Mechanics, St. Petersburg State University, St. Petersburg, Russia
[3] Theodosius Dobzhansky Center for Genome Bioinformatics, St. Petersburg State University, St. Petersburg, Russia
[4] Bigelow Laboratory for Ocean Sciences, East Boothbay, ME, USA
[5] J. Craig Venter Institute, La Jolla, California, USA
[6] DOE Joint Genome Institute, Walnut Creek, California, USA
[7] Dept. of Mathematics, University of California, San Diego, La Jolla, CA, USA
[8] Dept. of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA
[9] Dept. of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA

Abstract. Recent advances in single-cell genomics provide an alternative to gene-centric metagenomics studies, enabling whole genome sequencing of uncultivated bacteria. However, single-cell assembly projects are challenging due to (i) the highly non-uniform read coverage, and (ii) a greatly elevated number of chimeric reads and read pairs. While recently developed single-cell assemblers have addressed the former challenge, methods for assembling highly chimeric reads remain poorly explored. We present algorithms for identifying chimeric edges and resolving complex bulges in de Bruijn graphs, which significantly improve single-cell assemblies. We further describe applications of the single-cell assembler SPAdes to a new approach for capturing and sequencing "dark matter of life" that forms small pools of randomly selected single cells (called a *mini-metagenome*) and further sequences all genomes from the mini-metagenome at once. We demonstrate that SPAdes enables sequencing mini-metagenomes and benchmark it against various assemblers. On single-cell bacterial datasets, SPAdes improves on the recently developed E+V-SC and IDBA-UD assemblers specifically designed for single-cell sequencing. For standard (multicell) datasets, SPAdes also improves on A5, ABySS, CLC, EULER-SR, Ray, SOAPdenovo, and Velvet.

⋆ These authors contributed equally.

M. Deng et al. (Eds.): RECOMB 2013, LNBI 7821, pp. 158–170, 2013.

1 Introduction

The standard techniques for Next Generation Sequencing (NGS) require at least a million bacterial cells to sequence a genome. Since most bacteria cannot be cultured in the laboratory [1,2] and thus cannot be sequenced, most bacterial diversity remains below the radar of NGS projects. The "dark matter of life" describes microbes and even entire bacterial phyla that have yet to be cultured and sequenced. For example, only a fraction of the 10,000+ bacterial species in the human microbiome have been sequenced [3,4]. Single-cell sequencing [5,6] has recently emerged as a powerful approach to complement largely gene-centric metagenomic data with whole-genome assemblies of uncultivated organisms.

Currently, *Multiple Displacement Amplification (MDA)*, pioneered by Roger Lasken and colleagues [7], is the dominant approach to whole genome amplification prior to single-cell sequencing. However, assembly of reads from MDA-amplified genomes is challenging because of highly non-uniform read coverage, as well as elevated levels of chimeric reads and read pairs. While recent computational advances (Chitsaz et al., 2011 [8]; Peng et al., 2012 [9], Bankevich et al., 2012 [10]) have opened the possibility of sequencing the genome of any bacterial cell, sequencing the vast majority of bacteria in the human microbiome still remains a distant goal. The bottleneck is that it remains unclear how to isolate and capture low-abundance cells from a complex sample. In particular, while there is great interest in investigating the rare bacterial species in the human microbiome, currently there is no technology for comprehensively surveying the diversity of such a complex sample. Indeed, capturing and sequencing even 100,000 randomly chosen single cells from the human microbiome is unlikely to comprehensively sample the bacterial diversity, since many of the 10,000+ species in the human microbiome are underrepresented [11,12]. Since sequencing 100,000 single cells is prohibitively expensive, the question is how to sample bacterial diversity in a more economical way.

McLean et al., 2012 (submitted) recently developed a new approach for analyzing the "dark matter of life" based on forming random pools of single flow sorted cells and sequencing all cells in the resulting *mini-metagenome* at once. These pools only contain a small number of cells as opposed to metagenomics samples, which often contain billions of cells from different species. Since the artificially formed mini-metagenome has lower complexity than the original metagenome, high quality sequencing of mini-metagenomes (as opposed to metagenomes) becomes feasible.

Assembly of mini-metagenome MDA reads is even more challenging than for single-cell MDA reads, and thus requires additional algorithmic developments. Mini-metagenome sequencing can be thought of as sequencing a giant bacterial genome (formed by all genomes within a mini-metagenome) with extremely non-uniform coverage. Moreover, the elevated number of chimeric reads and read pairs (typical for single-cell sequencing) is likely to present an even more difficult challenge in the case of mini-metagenomes, where *intergenomic* chimeric reads (resulting from concatenated fragments from different genomes) can be formed.

This paper addresses computational challenges arising in single-cell and mini-metagenome sequencing. This includes detection of chimeric edges in de Bruijn graphs and analyzing complex bulges. We incorporate these algorithmic developments into the SPAdes assembler [10] and demonstrate that it improves on existing single-cell sequencing tools E+V-SC [8] and IDBA-UD [9]. SPAdes also performs well on standard (multicell) projects. (We refer to a conventional sequencing project using cultivated strains as *multicell* sequencing.) In particular, we show that it improves on A5 [13], ABySS [14], CLC[1], EULER-SR [15], Ray [16], SOAPdenovo [17]), and Velvet [18] in multicell bacterial assemblies. We also benchmark SPAdes on simulated mini-metagenomes obtained by mixing various single-cell read datasets. We demonstrate that SPAdes enables mini-metagenome sequencing, and we investigate the computational limits of mini-metagenome sequencing for assessing low-abundance bacterial species.

2 Identifying Chimeric Edges in de Bruijn Graphs

MDA often results in *chimeric reads* (formed by concatenating fragments from different regions of the genome) and *chimeric read-pairs* (formed by two reads sampled from distant regions of the genome). See [8,19,20] for the extent of chimeric reads and read-pairs in single-cell projects. Chimeric reads result in *chimeric edges* in de Bruijn graphs.

Double-Stranded de Bruijn Graphs. Let DB(Genome, k) be the de Bruijn graph of a circular genome Genome and its reverse complement Genome$'$, where vertices and edges correspond to $(k-1)$-mers and k-mers, respectively. Genome and Genome$'$ each traverse a cycle in this graph; these two cycles form the *genome traversal* of the graph. If a genome has multiple chromosomes or linear chromosomes, the genome traversal of DB(Genome, k) may consist of multiple paths or cycles. The *genomic multiplicity* of an edge is the number of times the traversal passes through this edge. We often work with *condensed graphs* [10], where each edge is assigned a *length* (in k-mers) and the length of a path is the sum of its edge lengths (rather than the number of edges).

Chimeric Edges in de Bruijn Graphs. Let DB(Reads, k) be the de Bruijn graph constructed from a set Reads of reads from Genome and their reverse complements. In the idealized case with full coverage of Genome and no read errors, the graphs DB(Reads, k) and DB(Genome, k) coincide; however, in reality these graphs differ because of coverage gaps and read errors. Edges in DB(Reads, k) may correspond to genome fragments (*correct* edges) as well as arise either from errors in reads or from chimeric reads (*false* edges). While in DB(Genome, k) the genome traversal consists of a pair of cycles, in DB(Reads, k) these cycles may be broken into multiple paths. The genome traversal defines the genomic multiplicities of edges in DB(Reads, k) (or the condensed graph). In particular,

[1] CLC Assembly Cell 3.22.55708 (CLC Bio, `http://www.clcbio.com`).

since false edges are not traversed by the genome traversal in $\mathrm{DB}(\textsc{Reads}, k)$, they have genomic multiplicity zero.

Assemblers use various algorithms to iteratively remove false edges and transform the de Bruijn graph $\mathrm{DB}(\textsc{Reads}, k)$ into a smaller *assembly graph*. We use the notation $\mathrm{DB}^+(\textsc{Reads}, k)$ to denote the current assembly graph at any intermediate stage of assembly, and $\mathrm{DB}^*(\textsc{Reads}, k)$ to denote the final assembly graph.

While most false edges correspond to easily detectable subgraphs, called *tips* and *bulges*, some form *chimeric edges*, which are hard to identify. Chimeric edges arise from chimeric reads, which are abundant in single-cell datasets. While chimeric edges in the de Bruijn graph represent a major obstacle to constructing long contigs, in standard (multicell) assembly datasets, chimeric edges usually have low coverage and thus are easily identified as false and removed by the conventional assemblers. However, this approach does not work for single-cell datasets, where coverage is non-uniform and the level of chimerism is high [8,19]. For such datasets, low coverage does not characterize false edges since many correct edges also have low coverage.

Our *chimeric edge identification* procedure is based on the following assumptions for bacterial genomes: (i) since chimeric edges in the condensed de Bruijn graphs are typically short,[2] we assume that edges longer than $n = 250$ have genomic multiplicity at least 1, and (ii) since edges longer than $N = 1500$ (referred to as *long* edges) in the condensed de Bruijn graph tend to have genomic multiplicity 1, we assume that all long edges have genomic multiplicity 1.[3]

Since genomic multiplicities of edges in $\mathrm{DB}^+(\textsc{Reads}, k)$ are unknown, we attempt to bound them. An edge e with genomic multiplicity bounded by $c_{\textsc{lower}}(e)$ from below and by $c_{\textsc{upper}}(e)$ from above has *capacity* $(c_{\textsc{lower}}(e), c_{\textsc{upper}}(e))$. We assign capacities to all edges in the condensed graph of $\mathrm{DB}^+(\textsc{Reads}, k)$ as follows, where the second and third categories are dictated by assumptions (i) and (ii) above:

$$(c_{\textsc{lower}}(e), c_{\textsc{upper}}(e)) = \begin{cases} (0, \infty) & \text{if } \textsc{Length}(e) \leq n; \\ (1, \infty) & \text{if } n < \textsc{Length}(e) \leq N; \\ (1, 1) & \text{if } N < \textsc{Length}(e). \end{cases}$$

To simplify this presentation, we assume that an assembly algorithm successfully removes all (or the vast majority of) bulges and tips, resulting in an intermediate assembly graph $\mathrm{DB}^+(\textsc{Reads}, k)$, but fails to remove chimeric edges. Thus, the search for chimeric edges amounts to finding edges of genomic multiplicity zero.

[2] Out of 117 chimeric edges in the graph $\mathrm{DB}^+(\textsc{Reads}, 55)$ constructed for the single-cell *E. coli* dataset ECOLI-SC (described in Results), 115 have length $\leq n = 250$. Here and in the further statistics $\mathrm{DB}^+(\textsc{Reads}, 55)$ is a graph that we have after doing initial simplifications including removing condensed edges with average coverage below 10 that satisfy some additional length and topology conditions.

[3] This holds for 97% of long edges in $\mathrm{DB}(\textsc{Genome}, 55)$ for the *E. coli* reference genome.

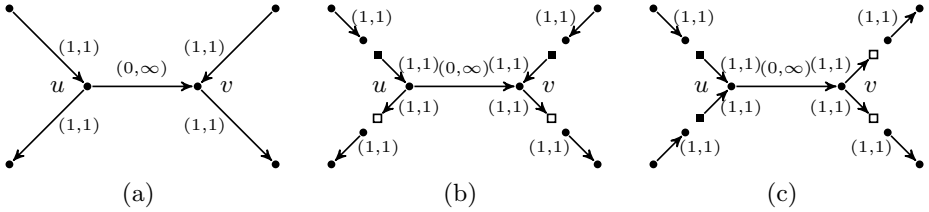

Fig. 1. Example of breaking long edges in an assembly graph. (a) Subgraph of assembly graph where the four diagonal edges are long edges, while the horizontal edge in the center is not long. (b) Result of breaking the four long edges contains a connected component (in the center) with 2 sources (solid square vertices) and 2 sinks (hollow square vertices). The capacities of the edges starting (ending) at the newly formed sources (sinks) are inherited from the capacities of the broken edges. (c) Result of breaking long edges in a subgraph similar to the subgraph in (a) but with different directions on some edges.

Chimeric Edges and Circulations in Networks. A graph with capacity constraints on the edges is referred to as a *network*. Given a vertex v and a function f on edges of a network G, we define $\text{INFLUX}_f(v) = \sum_e f(e)$, where the sum is taken over all incoming edges e of the vertex v. We define $\text{OUTFLUX}_f(v)$ similarly. A function f is called a *circulation* in the network G if $\text{INFLUX}_f(v) = \text{OUTFLUX}_f(v)$ for each vertex v in G, and $c_{\text{LOWER}}(e) \leq f(e) \leq c_{\text{UPPER}}(e)$ for each edge e in G.

The *Circulation Problem* is to find a circulation in a network [21]. Genomic multiplicities define a circulation in the network $\text{DB}^+(\text{READS}, k)$ with capacity constraints. There are usually multiple circulations in this network and we do not know which of them corresponds to the actual genomic multiplicities. However, if an edge e has $f(e) = 0$ in all circulations, then it must be a false edge and in most cases represents a chimeric edge. A polynomial-time algorithm for finding all such edges in the network will be described elsewhere.

We remark that this strategy is based on the assumption that the genome corresponds to a cycle in the graph, which often fails for real data. Since in $\text{DB}^+(\text{READS}, k)$ the genome traversal may be broken into multiple subpaths, a circulation in it may not even exist. To address this complication, we break the network into smaller subnetworks and analyze their subcirculations.

The operation of *breaking an edge* (v, w) in a graph G removes (v, w) from G; adds two new vertices v^* and w^* (called the *sink* and the *source*, respectively); and adds two new edges (v, v^*) and (w^*, w) with the same capacity as the edge (v, w). Given a weighted graph G and a positive integer t, we define G_t as the graph obtained from G by breaking all edges longer than t. To break the de Bruijn graph into subnetworks, we break all long edges (Fig. 1). After this transformation, the graph $\text{DB}^+(\text{READS}, k)$ is typically decomposed into many connected components. For the ECOLI-SC dataset described in the Results section, the graph is decomposed into 114 non-trivial connected components (containing more than one vertex).

A circulation (genome traversal) in $\mathrm{DB}^+(\textsc{Genome}, k)$ defines a *flow* [21] between sources and sinks of every connected component of $\mathrm{DB}^+_N(\textsc{Genome}, k)$ satisfying the capacity constraints. Similarly to $\mathrm{DB}(\textsc{Genome}, k)$, for many components in $\mathrm{DB}^+(\textsc{Reads}, k)$, the genome traversal also defines a flow satisfying the capacity constrains. Thus, the search for chimeric edges in $\mathrm{DB}^+(\textsc{Reads}, k)$, can be performed independently in each component.

Many components have a particularly simple structure with two sources and two sinks (Fig. 1b and Fig. 1c). The only flow that satisfies the capacity constraints in Fig. 1b (resp., Fig. 1c assigns flow 0 (resp., flow 2) to the edge (u, v). Thus (u, v) is classified as chimeric in Fig. 1b and as correct in Fig. 1c.

Unfortunately, the above procedure fails for some connected components (e.g., when the outgoing edge from vertex v in Fig. 1a is missing). Furthermore, such components tend to be large. For example, after assembling reads from a single *E. coli* cell, the procedure fails only for 10% of all components, but these components contain most (52%) vertices of the graph. Below we describe an approach to identify chimeric edges in such components.

Chimeric Edges and Critical Cut-sets. Given a subset U of vertices in the graph, the *cut-set*, denoted $\textsc{cut}(U)$, is the set of all edges (u, v) in the graph such that $u \in U$ and $v \in \overline{U}$ (where $\overline{U}$ denotes the set of vertices of the graph that do not belong to U). We define $c_{\textsc{lower}}(U)$ (resp., $c_{\textsc{upper}}(U)$) as the sum of lower (resp., upper) capacities of all edges from $\textsc{cut}(U)$. A cut-set $\textsc{cut}(U)$ is *balanced* if $c_{\textsc{lower}}(U) \leq c_{\textsc{upper}}(\overline{U})$ and *unbalanced* otherwise. A cut-set $\textsc{cut}(U)$ is *critical* if $c_{\textsc{lower}}(U) = c_{\textsc{upper}}(\overline{U})$. According to Hoffman's Circulation Theorem [21], a circulation exists if and only if every cut-set in the network is balanced.

It is easy to see that for a critical cut-set $\textsc{cut}(U)$, all edges $(u, v) \in \textsc{cut}(U)$ must have genomic multiplicity equal to $c_{\textsc{lower}}(u, v)$, while all edges $(v, u) \in \textsc{cut}(\overline{U})$ must have genomic multiplicity equal to $c_{\textsc{upper}}(v, u)$. Indeed, if the lower capacity of any edge $(u, v) \in \textsc{cut}(U)$ is increased by 1, the cut-set would become unbalanced (as $c_{\textsc{lower}}(U) + 1 > c_{\textsc{upper}}(\overline{U})$), implying that no circulation exists. Similarly, the upper capacity of any edge $(v, u) \in \textsc{cut}(\overline{U})$ cannot be decreased, implying that the genomic multiplicity of (v, u) must be equal to $c_{\textsc{upper}}(v, u)$. In particular, for a critical cut-set $\textsc{cut}(U)$, all *crossing* edges $(u, v) \in \textsc{cut}(U)$ with $c_{\textsc{lower}}(u, v) = 0$ must be chimeric.

SPAdes analyzes only certain types of critical cut-sets that are common in de Bruijn graphs of reads (details are to be described elsewhere).

3 Removing Complex Bulges

Bulges and Bulge Corremoval. Errors in reads often result in two short paths between the same two vertices in the de Bruijn graph, where the two paths have roughly the same length and represent similar sequences. Such pairs of paths may aggregate into larger subgraphs called *bulges*. Assemblers use various *bulge removal* algorithms (and additional steps) to transform the de Bruijn graph $\mathrm{DB}(\textsc{Reads}, k)$ into a smaller *assembly graph* $\mathrm{DB}^*(\textsc{Reads}, k)$. While they remove the vast majority of bulges, they fail to remove some *complex bulges*.

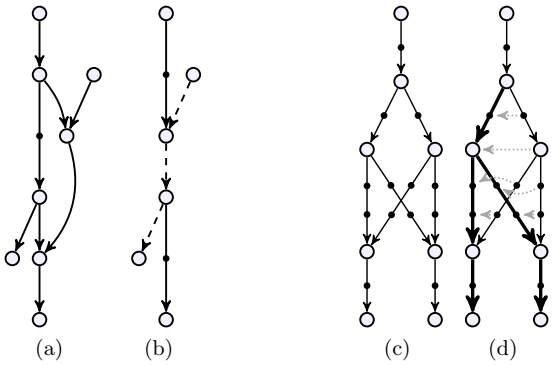

Fig. 2. Illustration of bulge removal algorithm. The vertices of the condensed graph are shown in white. Dotted arrows indicate projection operations (not graph edges). (a–b): Merging paths instead of projecting paths. Merging two paths in (a) results in a graph (b) with an artificial (dashed) path violating condition (ii). (c–d): Blob corremoval. Complex bulge (c) is not removed by the bulge corremoval procedure from [10]. Applying the new "blob corremoval procedure" to blob (c) simplifies it via the projections shown in (d). Thick edges denote the tree to which we project the blob.

In this section, we describe an algorithm for removal of complex bulges that evade the "bulge corremoval" algorithm from [10]. One approach to removing bulges is to map the de Bruijn graph onto a smaller graph. SPAdes tries to find a mapping that satisfies the following conditions:

(i) Every path in the de Bruijn graph maps to a path in the assembly graph.
(ii) For every path ρ in the assembly graph, there exists a path in the de Bruijn graph that maps onto ρ.[4]

Some bulge removal algorithms either do not explicitly map the de Bruijn graph onto the assembly graph or use mappings that may violate conditions (i) and/or (ii). For example, they may find a bulge formed by two paths in the de Bruijn graph and either remove one of the paths or merge these paths into a single one, without considering the impact on other edges incident to these paths. Removing one of the paths may lead to deterioration of assemblies, since important information (along with some correct paths) may be lost. Merging the paths may introduce artificial paths into the assembly graph, violating condition (ii) (see Fig. 2a,b).

SPAdes [10] introduced the *bulge corremoval* procedure, which satisfies conditions (i) and (ii). For each edge (u, v) (with length below a threshold) in the condensed de Bruijn graph, SPAdes searches for a path from u to v of length approximately equal to the length of (u, v). If such an *alternative path* exists, the two paths P and (u, v) form a *simple bulge*. To remove a simple bulge, the edge

[4] In fact, SPAdes creates the assembly graph as a subgraph of the de Bruijn graph so that paths in the assembly graph also represent paths in the de Bruijn graph.

(u, v) is *projected* onto this path and is removed afterwards. Applied iteratively, the bulge corremoval strategy eliminates the vast majority of bulges. However, for some bulges, no edge in the bulge has an alternative path, implying that the algorithm from [10] will not be able to remove such bulges (Fig. 2c). Below we describe an algorithm satisfying conditions (i) and (ii) for removing the majority of the remaining complex bulges.

Blob Corremoval. Let G be a directed acyclic graph (DAG) with vertex set V and edge set E. For vertices v and w in G, we define $v \prec w$ if there exists a directed path from v to w in G. A mapping $f : V \to V, E \to E$ is called a *projection* if (1) for every vertex $v \in V$, we have $f(f(v)) = f(v)$, (2) for every pair of vertices v and w, if $v \prec w$ then $f(v) \prec f(w)$, and (3) for every edge $e = (u, v)$, we have $f(e) = (f(u), f(v))$.

A projection f defines the induced DAG G_f on the vertex set $V_f = f(V)$. We limit our attention to projections of DAGs onto *directed trees* (i.e., projections f such that G_f is a directed tree) and additionally require that every path and its projection have similar lengths. Fig. 2d shows the directed tree and projection of DAG shown in Fig. 2c.

Breaking edges longer than t in the assembly graph $\mathrm{DB}^+(\textsc{Reads}, k)$ results in a graph $\mathrm{DB}^+_t(\textsc{Reads}, k)$ that typically consists of many connected components. A *blob* is a component of $\mathrm{DB}^+_t(\textsc{Reads}, k)$ that is a DAG with a single source and one or more sinks. SPAdes analyzes blobs in $\mathrm{DB}^+(\textsc{Reads}, k)$ and for each blob, attempts to find a directed tree (with root at the source and leaves at the sinks of the blob) such that there exists a projection of the blob onto this tree.

This leads to a *blob corremoval* procedure, which generalizes the bulge corremoval procedure from [10] and satisfies conditions (i), (ii). A generalized notion of blob and efficient algorithm to search for trees and projections will be described elsewhere.

4 Results

Metrics. The N50 (resp., NG50) metric is the maximum contig size such that using blocks of that size or larger gives at least 50% of the assembly length (resp., reference genome length). We use metrics NA50 and NGA50, introduced and justified in [22], instead of the standard N50 or NG50 metrics. To count NA50, contigs are alligned to reference genome. If a contig has a misassembly or has nonaligning sequence such as large gaps or indels, the contig is broken into blocks that do align. Then we compute N50 using these aligned blocks instead of using the original contigs. Similarly, NGA50 is computed as NG50 applied to these adjusted blocks.

In some of our experiments, the fraction of the genome assembled is below 50%, so NGA50 would be 0 for all assemblers, and thus, we use NA50.

Benchmarking. We compared a number of single-cell and conventional assemblers on two *E. coli* paired-end Illumina libraries described in [8]: a single-cell

Table 1. Comparison of assemblers on ECOLI-SC, a single-cell *E. coli* dataset, using QUAST [22]. In each column, the best assembler by that criteria is indicated in bold. Only contigs of length ≥ 500 bp were used. For single cell projects, the total assembly size often exceeds the genome length due to contaminants and other reasons (see [23]). The "GF (%)" (Genome fraction) column filters out these issues. MA: number of misassemblies. Misassemblies are locations on an assembled contig where the left flanking sequence aligns over 1 kb away from the right flanking sequence on the reference. MM: Mismatch (substitution) error rate per 100 kb. IND: number of indels per 100 kb. MM and IND are measured in aligned regions of the contigs.

Assembler	NGA50	# contigs	Longest contig	Total length	MA	MM	IND	GF (%)	# genes
Conventional (multicell) assemblers									
A5	14399	745	101584	4441145	8	11.97	0.19	90.141	3453
ABySS	68534	**179**	178720	4345617	5	2.71	2.66	88.268	3704
CLC	32506	503	113285	4656964	3	4.76	2.87	92.378	3768
EULER-SR	26662	429	140518	4248713	18	9.37	218.72	85.005	3419
Ray	55395	296	210612	4649552	13	2.34	0.87	91.864	3838
SOAPdenovo	18468	569	87533	4098032	7	114.38	11.08	79.861	3038
Velvet	22648	261	132865	3501984	**2**	2.07	1.23	74.254	3098
Single-cell assemblers									
E+V-SC	32051	344	132865	4540286	**2**	1.85	0.70	92.162	3793
IDBA-UD	96947	250	224018	4791744	10	**1.61**	**0.16**	95.661	4046
SPAdes 2.3	**110539**	276	**268756**	4875378	**2**	4.24	0.70	**95.737**	**4057**

library (ECOLI-SC) and a multicell library (ECOLI-MC). They consist of 100 bp paired-end reads with average insert sizes 266 bp for ECOLI-SC and 215 bp for ECOLI-MC. Both *E. coli* datasets have 600× coverage. The *E. coli* K-12 MG1655 reference length is 4639675 bp with 4324 annotated genes.

Tables 1 and 2 present the benchmarking results for various assemblers.[5] Table 1 illustrates that single-cell assemblers significantly improve on the conventional assemblers in *single-cell* projects. Table 2 shows that recently developed single-cell assemblers IDBA-UD and SPAdes also improve on the conventional assemblers in *standard (multicell)* projects by most metrics.

From Genomes to Mini-metagenomes. Below we investigate the performance of SPAdes on artificially simulated mini-metagenomes and demonstrate that it is capable of assembling a significant portion of each genome in a mini-metagenome. In addition to simulations, we also applied this assembly algorithm to a real mini-metagenome dataset; details are in McLean et al., 2012 (submitted).

[5] ABySS 1.3.4, EULER-SR 2.0.1, Ray 2.0.0, Velvet, Velvet-SC, and E+V-SC were run with vertex size 55. A5 and CLC 3.22.55708 were run with default parameters. SOAPdenovo 1.0.4 was run with vertex sizes 27–31. IDBA-UD 1.0.9 was run in its default iterative mode.

Table 2. Comparison of assemblers on ECOLI-MC, a multicell *E. coli* dataset. See the caption of Table 1 for further details.

Assembler	NGA50	# contigs	Longest contig	Total length	MA	MM	IND	GF (%)	# genes
Conventional (multicell) assemblers									
A5	43651	176	181690	4551797	**0**	**0.40**	**0.13**	98.476	4178
ABySS	106155	**96**	221861	4619631	2	2.45	0.52	99.202	4242
CLC	69146	122	221533	4547925	2	0.73	0.15	98.547	4233
EULER-SR	110153	100	221409	4574240	8	2.98	47.15	98.438	4206
Ray	83128	113	221942	4563341	2	2.10	0.20	98.162	4194
SOAPdenovo	62512	141	172567	4519621	1	26.56	5.58	97.405	4134
Velvet	82776	120	242032	4554702	3	0.70	0.20	98.824	4211
Single-cell assemblers									
E+V-SC	54856	171	166115	4539639	**0**	1.21	0.15	98.329	4149
IDBA-UD	111789	107	236470	4562955	1	0.53	0.15	98.834	4215
SPAdes 2.3	**119880**	104	**265405**	**4634928**	2	1.86	0.63	**99.420**	**4250**

We applied SPAdes to a simulated mini-metagenome that consists of four bacterial species with known genomes. We mixed together reads (in various proportions), from four different MDA-amplified single-cell bacterial projects at Joint Genome Institute and Bigelow Laboratory. The genomes of these bacteria vary in GC content and genome length: *Prochlorococcus marinus* (31% GC, 1.7 Mb genome length) [24], *Pedobacter heparinus* [25] (42% GC, 5.0 Mb genome length), *Escherichia coli* [26] (51% GC, 4.6 Mb genome length), and *Meiothermus ruber* [27] (63% GC, 3.0 Mb genome length). Note that as simulated datasets, these are highly idealized and do not exactly match what would be found in the environment, but they are useful for modeling the ability of the assembler to deal with different mixtures.

In the first simulation, we randomly selected a fixed fraction of reads from each genome, mixed them together, and assembled the resulting dataset with SPAdes. This simulation was repeated 10 times, varying the fraction as $1/2^m$ with $m = 0, 1, \ldots, 9$ (the same fraction $1/2^m$ applies to all genomes). The assembled contigs were aligned against individual genomes to compute the assembly statistics (in Table 3 we present statistics for *M. ruber* and *P. heparinus*). In particular, even with a relatively small fraction $1/64$ of selected reads, SPAdes assembled 1779 out of 4339 genes for *P. heparinus*, 1366 out of 4324 genes for *E. coli*, and 710 out of 3105 genes for *M. ruber*. This is significantly larger than the number of complete genes captured in a typical metagenomics project. However, for *P. marinus*, only 55 out of 1732 genes were assembled.

In the second simulation, we formed a mini-metagenome using all reads from three species and varied the coverage for the fourth. For the fourth species, we selected either a genome with high GC content (*M. ruber*) or low GC content (*P. heparinus*). Table 4 illustrates that SPAdes recovers a substantial fraction of an underrepresented genome within a mini-metagenome. Even with a

Table 3. SPAdes assembly of a simulated mini-metagenome with equal fractions of each genome

	1/1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
M. ruber:										
Misassemblies	14	12	5	15	13	10	14	8	8	12
NA50 (kb)	44	33	39	24	20	14	9	10	8	3
Longest contig (kb)	113	133	119	114	108	109	93	61	50	39
Genome fraction (%)	76.3	69.5	62.4	56.7	49.1	39.7	29.9	22.2	15.7	10.9
# genes	2160	1950	1777	1533	1219	939	710	521	357	214
P. heparinus:										
Misassemblies	1	2	6	11	11	13	26	27	17	12
NA50 (kb)	185	207	165	96	70	27	12	4	2	1
Longest contig (kb)	946	410	426	307	337	379	225	102	34	32
Genome fraction (%)	97.8	96.6	94.3	88.4	82.5	71.3	57.6	40.4	24.9	12.1
# genes	4148	4038	3855	3524	3133	2401	1779	1125	548	189

Table 4. SPAdes assembly of a mini-metagenome with variable fraction of *M. ruber* and *P. heparinus*

	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512
M. ruber:									
Misassemblies	14	6	8	11	13	14	14	17	11
NA50 (kb)	47	45	27	15	11	10	7	4	2
Longest contig (kb)	137	120	118	106	114	114	62	45	49
Genome fraction (%)	70.0	63.7	58.3	52.4	43.0	34.2	27.1	21.5	16.7
# genes	1973	1797	1585	1327	1022	822	618	464	330
P. heparinus:									
Misassemblies	2	5	8	12	19	23	14	19	15
NA50 (kb)	163	165	131	87	33	11	3	2	1
Longest contig (kb)	426	439	396	339	333	227	89	40	33
Genome fraction (%)	96.6	94.0	88.7	81.6	71.2	56.7	40.6	24.5	12.9
# genes	4043	3815	3527	3116	2488	1752	1113	534	228

small fraction of reads in the underrepresented genome (e.g., 1/256), we recovered a significantly larger number of genes (more than 450 genes for *M. ruber* and *P. heparinus*) as compared to a typical metagenomics project. Tables 3 and 4 demonstrate that the assembly quality of an individual genome depends mainly on the coverage of this genome, rather than on what fraction of the mini-metagenome this genome represents.

5 Discussion

Since 2008, when the first NGS assemblers were released, many excellent assemblers have become available. Since most of them use a de Bruijn graph approach, they often generate rather similar assemblies, at least for bacterial projects. Recent developments in single-cell genomics tested the limits of conventional

assemblers and demonstrated that they all have room for improvement. Our benchmarking illustrates that single-cell assemblers not only enable single-cell sequencing but also improve on conventional assemblers on their own turf.

References

1. Rappe, M.S., Giovannoni, S.J.: The uncultured microbial majority. Annu. Rev. Microbiol. 57, 369–394 (2003)
2. Tringe, S.G., Rubin, E.M.: Metagenomics: DNA sequencing of environmental samples. Nat. Rev. Genet. 6(11), 805–814 (2005)
3. Nelson, K.E., Weinstock, G.M., Highlander, S.K., Worley, K.C., Creasy, H.H., et al.: A catalog of reference genomes from the human microbiome. Science 328(5981), 994–999 (2010)
4. Wylie, K.M., Truty, R.M., Sharpton, T.J., Mihindukulasuriya, K.A., Zhou, Y., et al.: Novel bacterial taxa in the human microbiome. PLoS ONE 7(6), e35294 (2012)
5. Stepanauskas, R.: Single cell genomics: an individual look at microbes. Current Opinion in Microbiology 15(5), 613–620 (2012)
6. Lasken, R.S.: Genomic sequencing of uncultured microorganisms from single cells. Nat. Rev. Microbiol. 10(9), 631–640 (2012)
7. Lasken, R.S.: Single-cell genomic sequencing using Multiple Displacement Amplification. Curr. Opin. Microbiol. 10(5), 510–516 (2007)
8. Chitsaz, H., Yee-Greenbaum, J., Tesler, G., Lombardo, M., Dupont, C., et al.: Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. Nat. Biotechnol. 29(10), 915–921 (2011)
9. Peng, Y., Leung, H.C.M., Yiu, S.M., Chin, F.Y.L.: IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics 28(11), 1420–1428 (2012)
10. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.A., Dvorkin, M., et al.: SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. Journal of Computational Biology 19(5), 455–477 (2012)
11. Huttenhower, C., Gevers, D., et al.: Structure, function and diversity of the healthy human microbiome. Nature 486(7402), 207–214 (2012)
12. Li, K., Bihan, M., Yooseph, S., Methe, B.A.: Analyses of the microbial diversity across the human microbiome. PLoS ONE 7(6), e32118 (2012)
13. Tritt, A., Eisen, J.A., Facciotti, M.T., Darling, A.E.: An integrated pipeline for de novo assembly of microbial genomes. PLoS ONE 7(9), e42304 (2012)
14. Simpson, J., et al.: ABySS: a parallel assembler for short read sequence data. Genome Res. 19(6), 1117–1123 (2009)
15. Chaisson, M., Brinza, D., Pevzner, P.: De novo fragment assembly with short mate-paired reads: Does the read length matter? Genome Res. 19(2), 336–346 (2009)
16. Boisvert, S., Laviolette, F., Corbeil, J.: Ray: Simultaneous assembly of reads from a mix of high-throughput sequencing technologies. Journal of Computational Biology 17(11), 1519–1533 (2010)
17. Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., et al.: De novo assembly of human genomes with massively parallel short read sequencing. Genome Res. 20(2), 265–272 (2010)
18. Zerbino, D., Birney, E.: Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 18(5), 821–829 (2008)

19. Lasken, R.S., Stockwell, T.B.: Mechanism of chimera formation during the multiple displacement amplification reaction. BMC Biotechnol. 7, 19 (2007)
20. Woyke, T., Xie, G., Copeland, A., González, J.M., Han, C., Kiss, H., Saw, J.H., Senin, P., Yang, C., Chatterji, S., Cheng, J.F., Eisen, J.A., Sieracki, M.E., Stepanauskas, R.: Assembling the marine metagenome, one cell at a time. PLoS ONE 4(4), e5299 (2009)
21. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press (1962)
22. Gurevich, A., Saveliev, V., Vyahhi, N., Tesler, G.: QUAST: Quality Assessment for Genome Assemblies (2012) (submitted)
23. Woyke, T., Sczyrba, A., Lee, J., Rinke, C., Tighe, D., et al.: Decontamination of MDA reagents for single cell whole genome amplification. PLoS ONE 6(10), e26161 (2011)
24. Dufresne, A., Salanoubat, M., Partensky, F., Artiguenave, F., Axmann, I.M., et al.: Genome sequence of the cyanobacterium Prochlorococcus marinus SS120, a nearly minimal oxyphototrophic genome. Proceedings of the National Academy of Sciences 100(17), 10020–10025 (2003)
25. Han, C., et al.: Complete genome sequence of Pedobacter heparinus type strain (HIM 762-3 T). Standards in Genomic Sciences 1(1) (2009)
26. Blattner, F.R., Plunkett, G., Bloch, C.A., Perna, N.T., Burland, V., et al.: The complete genome sequence of Escherichia coli K-12. Science 277(5331), 1453–1462 (1997)
27. Tindall, B., Sikorski, J., Lucas, S., Goltsman, E., Copeland, A., et al.: Complete genome sequence of Meiothermus ruber type strain (21 T). Standards in Genomic Sciences 3(1) (2010)