DBG2OLC: Efficient Assembly of Large Genomes Using Long Erroneous Reads of the Third Generation Sequencing Technologies

Chengxi Ye1,2, Christopher M. Hill1, Shigang Wu3, Jue Ruan3 & Zhanshan (Sam) Ma2

The highly anticipated transition from next generation sequencing (NGS) to third generation sequencing (3GS) has been difficult primarily due to high error rates and excessive sequencing cost. The high error rates make the assembly of long erroneous reads of large genomes challenging because existing software solutions are often overwhelmed by error correction tasks. Here we report a hybrid assembly approach that simultaneously utilizes NGS and 3GS data to address both issues. We gain advantages from three general and basic design principles: (i) Compact representation of the long reads leads to efficient alignments. (ii) Base-level errors can be skipped; structural errors need to be detected and corrected. (iii) Structurally correct 3GS reads are assembled and polished. In our implementation, preassembled NGS contigs are used to derive the compact representation of the long reads, motivating an algorithmic conversion from a de Bruijn graph to an overlap graph, the two major assembly paradigms. Moreover, since NGS and 3GS data can compensate for each other, our hybrid assembly approach reduces both of their sequencing requirements. Experiments show that our software is able to assemble mammalian-sized genomes orders of magnitude more quickly than existing methods without consuming a lot of memory, while saving about half of the sequencing cost.

The Human Genome Project (HGP), which is perhaps the largest biomedical research project humans have ever undertaken, is responsible for greatly accelerating the advancement of DNA sequencing technologies1. Three generations of DNA sequencing technologies have been developed in the last three decades, and we are at the crossroads of the second and third generations of the sequencing technologies. Third generation sequencing (3GS) technology promises to significantly improve assembly quality and expand its applications in biomedical research and biotechnology development. However, lack of efficient and effective genome assembly algorithms has arguably been the biggest roadblock to the widespread adoption of 3GS technologies. 3GS long reads (averaging up to 5–20 kb per run at this time) usually have high error rates: ~15% with PacBio sequencing2, and as high as ~40% with Oxford Nanopore sequencing3. These high error rates make the assembly of 3GS sequences seem disproportionately complex and expensive compared to the assembly of NGS sequences. As a comparison, the whole genome assembly of a human genome using 3GS data was first reported to have taken half a million CPU hours4 compared to ~24 hours with Illumina NGS sequencing data5. Consequently, in practice, many applications of 3GS technology have been limited to re-sequencing bacteria and other small genomes6. While software for 3GS assembly has made important improvements7–17, especially for high coverage data, the software is still quite slow and not ideally suited for modest coverage data. Another major issue is that the sequencing cost of 3GS...
technology, while decreasing with time, is still at least an order of magnitude more expensive than the popular Illumina NGS sequencing at the time of this work.

While the evolution of genome assembly software solutions has been influenced by multiple factors, the most significant one has been the length of the sequences. Although increasing sequence lengths may simplify the assembly graph, the sequence length also has a critical impact on the computational complexities of genome assembly. Computational biologists have historically formulated the genome assembly problem as a graph traversal problem, i.e., searching for a most likely genome sequence from the overlap graph of the sequence reads in the case of the first generation sequencing technology. The string graph and the best overlap graph are specific forms of the Overlap-Layout-Consensus (OLC) paradigm that are more efficient by simplifying the global overlap graph. The read-based algorithms, aiming to chain the sequencing reads in the most effective way, are computationally expensive because pair-wise alignment of the sequences is required to construct the overlap graph. This issue was tolerable for the relatively low amount of sequences produced from the low-throughput first generation sequencing technologies, but quickly became overwhelming with the enormous amount of short reads produced by high-throughput NGS data. The strategy of chopping the sequencing reads into shorter and overlapping k-grams (so-called k-mers) and building links between the k-mers, was developed in the de Bruijn graph (DBG) framework to simplify NGS assembly. Assembly results are extracted from the linear (unbranched) regions of the k-mer graph in this approach.

The overlap graph model or the OLC-based software packages, such as Celera Assembler, AMOS and ARACHNE, originally used for assembling the first generation sequence data, were also adopted for the NGS assembly before DBG-based approaches became the de facto standard. Newer 3GS technologies, including single-molecule, real-time sequencing (SMRT) and Oxford Nanopore sequencing, produce much longer reads than NGS. The longer reads from 3GS technology were used in the first generation genome assembly, feasible again. Nevertheless, the high error rates of current 3GS technologies render the existing OLC-based assemblers developed for relatively accurate sequences unusable. Similarly, the error-prone long reads make the DBG full of branches and therefore unsuitable for 3GS assembly. Faced with these challenges, the developers of 3GS technology have resorted to using error correction techniques to introduce algorithmic techniques that effectively resolve many of these issues. But first, we present a brief account of the existing genome assembly software technologies to put our contribution in proper context.

Researchers began with scaffolding approaches such as AHA, PBjelly and SPACE-LongRead to patch the gaps between high quality assembly regions, i.e., first build a scaffold by aligning reads to the contigs and then use reads that span multiple contigs as links to build a scaffold graph. In ALLPATHS-LG and Celera, long reads are used to find the best path in the de Bruijn graph that bridges the gaps between large contigs. Although these software packages have indeed achieved important advances for 3GS genome assembly, resolving intricate ambiguities is inherently difficult and can lead to structural errors. Furthermore, the underlying graph search algorithms usually have exponential complexity with respect to the search depth and thus, scales poorly; highly repeating regions such as long repeats of simple sequence (SSRs) will lead to large search depths and are not resolvable. In addition, the more powerful read overlap graph structure (of the long reads) was not fully explored in all these approaches. Often these algorithms rely on heuristics such as contig lengths and require iterations. To circumvent these important issues associated with the hybrid approach, a Hierarchical Genome-Assembly Process (HGAP) was developed using a non-hybrid strategy to assemble PacBio SMRT sequencing data, which does not use the NGS short reads. HGAP contains a consensus algorithm that creates long and highly accurate overlapping sequences by correcting errors on the longest reads using shorter reads from the same library. This correction approach was proposed earlier in the hybrid setting and is widely used in assembly pipelines. Nonetheless, this non-hybrid, hierarchical assembly approach requires relatively high sequencing coverage (50x–100x) and substantial error correction time to obtain satisfactory results. It is noteworthy that most of the algorithms we reviewed here were originally designed for bacterial-sized genomes. Though recent advancements in aligning erroneous long reads have also shortened the computational time of 3GS assembly, running these programs on large genomes, especially mammalian-sized genomes, usually imposes a large computational burden (sometimes up to 106 or 107 CPU hours) more suited to large computational clusters and well beyond the capability of a typical workstation.

In this study, we design algorithms to enable efficient assembly of large mammalian-sized genomes. We observe that per-base error correction of each long erroneous read and their pair-wise alignment takes a significantly large proportion of time in existing pipelines, but neither of them is necessary at an initial assembly stage. If all sequencing reads are structurally correct (non-chimeric), one can produce a structurally correct draft genome and improve the base-level accuracy in the final stage, as was originally done in the OLC approach. Taking advantage of this observation, we develop a base-level correction-free assembly pipeline by directly analyzing and exploiting overlap information in the long reads. Unlike previous approaches, we use the NGS assembly to lower the computational burden of aligning 3GS sequences rather than just polishing 3GS data. This allows us to take advantage of the cheap and easily accessible NGS reads, while avoiding the issues associated with existing hybrid approaches mentioned previously. Meanwhile, since NGS and 3GS are independent of each other, the sequencing gaps in one type of data may be covered by the data from the other. The utilization of NGS data also lowers the required sequencing depth of 3GS, and the net result is reduced sequencing cost. Hence, we get the best of both worlds of hybrid and non-hybrid assembly approaches. Specifically, we map the DBG contigs from NGS data to the 3GS long reads to create anchors for the long reads. Each long read is (lossily) compressed into a list of NGS contig
Because the compressed reads are often orders of magnitude shorter than the original reads, finding candidate overlaps between them becomes a simple bookkeeping problem and the approximate alignments and overlaps can be calculated cheaply with the help of the contig identifiers. An overlap graph is constructed by chaining the best overlapped-reads in the compressed domain. The linear unbranched regions of the overlap graph are extracted and uncompressed to construct the draft assembly. Finally, we polish the draft assembly at the base-level with a consensus module to finish the assembly. Overall, compared with the existing approaches, our algorithm offers an efficient algorithmic solution for assembling large genomes with 3GS data in terms of computational resources (time and memory) and required sequencing coverage while also being robust to sequencing errors. Furthermore, our pipeline utilizes the reads overlap information directly and provides an efficient solution to the traditional read threading problem, which is valuable both theoretically and practically even for the NGS assembly.

Methods and Implementations
Our algorithm starts with linear unambiguous regions of a de Bruijn graph (DBG), and ends up with linear unambiguous regions in an overlap graph (used in the Overlap-Layout-Consensus framework). Due to this property, we dub our software DBG2OLC. The whole algorithm consists of the following five procedures, and we implement them as a pipeline in DBG2OLC. Each piece of the pipeline can be carried out efficiently.

1. Construct a de Bruijn graph (DBG) and output contigs from highly accurate NGS short reads.
2. Map the contigs to each long read to anchor the long reads. The long reads are compressed into a list of contig identifiers to reduce the cost of processing the long reads (Fig. 1A).
3. Use multiple sequence alignment to clean the compressed reads, and remove reads with structural errors (or so-called chimeras) (Fig. 2).
4. Construct a best overlap graph using the cleaned compressed long reads (Fig. 1B).
5. Uncompress and chain together the long reads (Fig. 1C), and resort to a consensus algorithm to convert them into DNA sequences (Fig. 1D).

Details for procedure (2–5) are explained below. The explanation of procedure (1) can be found in our previous SparseAssembler for NGS technology and is omitted here.

Availability. The source code and a compiled version of DBG2OLC is available in the following site: https://github.com/yechengxi/DBG2OLC.

Reads Compression
We use a simple k-mer index technique to index each DBG contig and map the pre-assembled NGS contigs back to the raw sequencing reads as anchors. The k-mers that appear in multiple contigs are excluded in our analysis to
length with 3GS. The compression usually leads to three factors of reduction in read over five datasets including genomes of different sizes and different sequencing technologies (Table 1, resources identifiers and can be done quickly with low memory. To demonstrate the effectiveness of this strategy, we ran it stage. Similarity detection between these compressed reads becomes a simpler bookkeeping problem with the small 3GS sequencing gaps may be covered by NGS contigs. These sequencing gaps will be bridged in the final gap regions in NGS assembly can be covered by 3GS data. Likewise, low or even no NGS coverage; these important important finding of this work is that per-base accuracy may not be a major roadblock for assembly contiguity. Rather, the chimeras or structural sequencing errors are the major “hot spots” worth putting major effort into. Without cleaning these chimeras, the overlap relations include many falsely generated reads and will lead to a

Figure 2. Reads correction by multiple sequence alignment. The left portion shows removing a false positive anchoring contig (brown) that appears only once in the multiple alignment. The right portion shows detection of a chimeric read by aligning it to multiple reads. A breakpoint is detected as all the reads can be aligned with the left portion of the target read are not consistent with all the reads that can be aligned with the right portion of the target read.

avoid ambiguity. Empirically for PacBio reads, we found that using k = 17 were adequate for all our experiments. For each 3GS long read, we report the matching contig identifiers as an ordered list. A contig identifier is reported if the number of uniquely matching k-mers in that contig is above a threshold, which is adaptively determined based on the contig length. We set this threshold in the range of (0.001 – 0.02)*Contig_Length. This easily tunable threshold parameter allows the user to find a balance between sensitivity and specificity. With low coverage data-sets, this parameter is set lower to achieve better sensitivity; otherwise it is set to higher to enforce better accuracy. In all our experiments, the contigs are generated with our previous SparseAssembler2. After this procedure, each read is converted into an ordered list of contig identifiers. An example of such a list is {Contig_a, Contig_b}, where Contig_a and Contig_b are identifiers of two different contigs. We also record the orientations of these contigs in the mapping. This compact representation is a lossy compression of the original long reads. We term the converted reads as compressed reads in this work. A compressed read is considered to be equivalent to its reverse complement, and the same compressed reads are then collapsed. Since a de Bruijn graph can efficiently partition the genome into chunks of bases as contigs, this lossy compression leads to orders of magnitude reduction in data size. Moreover, the compact representation can span through small regions with low or even no NGS coverage; these important gap regions in NGS assembly can be covered by 3GS data. Likewise, small 3GS sequencing gaps may be covered by NGS contigs. These sequencing gaps will be bridged in the final stage. Similarity detection between these compressed reads becomes a simpler bookkeeping problem with the identifiers and can be done quickly with low memory. To demonstrate the effectiveness of this strategy, we ran it over five datasets including genomes of different sizes and different sequencing technologies (Table 1, resources can be found in the Supplementary Materials). The compression usually leads to three factors of reduction in read length with 3GS.

Ultra-fast Pair-wise Alignments
Most existing algorithms rely on sensitive algorithms27,28 to align reads to other reads or assemblies. In our approach, since the compressed reads are usually much shorter than the original reads, alignments of these compressed reads can be calculated far more efficiently. We adopt a simple bookkeeping strategy and use the contig identifiers to build an inverted-index. Each identifier points to a set of compressed reads that contain this identi- fier. This inverted-index helps us to quickly select the potentially overlapping reads based on shared contig iden-tifiers. Alignments are calculated only with these candidate compressed reads. The alignment score is calculated using the Smith-Waterman algorithm29; the contig identifiers that can be matched are positively scored while the mismatched contig identifiers are penalized. Scores for match/mismatch are calculated based on the involved contig lengths or the number of matching k-mers in the previous step. With the compressed reads, our algorithm can finish pair-wise alignments in a small amount of time.

As discussed previously, state-of-the-art assembly pipelines usually resort to costly base-level error correction algorithms to correct each individual read2,7,8,10,13, which they then feed into an existing assembler. However, an important finding of this work is that per-base accuracy may not be a major roadblock for assembly contiguity. Rather, the chimeras or structural sequencing errors are the major “hot spots” worth putting major effort into. Without cleaning these chimeras, the overlap relations include many falsely generated reads and will lead to a

| Datasets    | Sequencing Technology | Average Raw Read Length | NGS Contig N50 (DBG k-mer size) | Average Compressed Read Length | Compression Ratio |
|-------------|-----------------------|-------------------------|---------------------------------|-------------------------------|------------------|
| S. cervisiae | PacBio               | 4,734                   | 31,233 (k = 51)                 | 7                             | 1:676            |
| A. thaliana  | PacBio               | 5,614                   | 2,264 (k = 51)                  | 8                             | 1:702            |
| H. sapiens   | PacBio               | 14,519                  | 3,115 (k = 51)                  | 11                            | 1:320            |
| E.coli K12   | Oxford Nanopore      | 6,597                   | 3,303 (k = 21)                  | 4                             | 1:1649           |
| E.coli K12   | Illumina Miseq       | 150                     | 3,303 (k = 21)                  | 2                             | 1:75             |

Table 1. The demonstration of the compression ratio on various datasets.
tangled overlap graph. To resolve this issue, we compute multiple sequence alignments (MSA) by aligning each compressed read with all other candidate compressed reads. With MSA we can detect the chimeric reads and the spurious contig identifiers in each read (Fig. 2). Both of these errors are cleaned up. The major side effect of this correction is a slightly increased requirement of the 3GS data coverage so that each compressed read can be confirmed by at least another one. The remaining minor errors (mostly false negatives) in the cleaned compressed reads will be tolerated by the alignment algorithm. In our experiments, we noticed that this algorithm is accurate enough to find high quality overlaps and can be used for constructing draft genomes as assembly backbones.

**Read Overlap Graph**

Compared with most hybrid approaches that used long reads to link together the short read contigs, our approach takes the unorthodox way—we use the short read contigs to help link together the long reads. We construct a best overlap graph\(^2\)\(\) using the above-described alignment algorithm with the compressed reads. In the best overlap graph, each node represents a compressed read. For each node, the best overlapped nodes (one before and the other after) are found based on the overlap score, and the links between these nodes are recorded. The overlap graph is calculated in two rounds (Fig. 1B). In round 1, all the contained nodes (with respect to other nodes) are filtered off. For example, \(|\text{Contig}_a, \text{Contig}_b|\) is removed if \(|\text{Contig}_a, \text{Contig}_b, \text{Contig}_c|\) is present. With this strategy, alignments with repeating and contained nodes are avoided. In round 2, all suffix-prefix overlaps among the remaining nodes are detected with the alignment algorithm. Nodes are chained one to another in both directions and in the best overlapped fashion. Graph simplification is applied to remove tiny tips and merge bubbles in the best overlap graph. Truly unresolvable repeats result in branches in the graph\(^2\) and will be kept as the assembly breakpoints.

Note that constructing the overlap graph with the compressed reads offers us several major benefits. (1) Long read information is sufficiently utilized. (2) The costly long read alignments are accelerated with the easily available NGS contigs. (3) The expensive graph search algorithms (with exponential complexity to the search depth) often used for graph resolving in many existing genome assembly programs are no long needed in our software.

**Consensus**

It is noteworthy that only in this final stage that the compressed reads are converted back to the raw nucleotide reads for polishing purpose. Linear unbranched regions of the best overlap graph encode the unambiguously assembled sequences. Uncompressed long reads that lie in these regions are laid out in the best-overlapped fashion and patched one after another (Fig. 1C). NGS contigs are included when there is a gap in the 3GS data. Reads that are related to each backbone are collected based on the contig identifiers. A consensus module is finally called to align these reads to each backbone and calculate the polished assembly (Fig. 1D). To polish the 3GS assembly that are related to each backbone are collected based on the contig identifiers. A consensus module is finally called to align these reads to each backbone and calculate the polished assembly (Fig. 1D). To polish the 3GS assembly that are related to each backbone are collected based on the contig identifiers. A consensus module is finally called to align these reads to each backbone and calculate the polished assembly (Fig. 1D). To polish the 3GS assembly that are related to each backbone are collected based on the contig identifiers. A consensus module is finally called to align these reads to each backbone and calculate the polished assembly (Fig. 1D).

**Results**

We conducted a comprehensive comparison on a small yeast genome (12 Mbp) dataset to provide a scope of the performance of each software program we compared in this study. Since most other programs do not scale linearly with the data scale and require thousands of hours per-run on genomes larger than 100 Mbp, the readers are encouraged to read through their original publications for the performance results of those programs.

As a side note, the advent of 3GS long reads has raised the bar to a higher level compared with previous sequencing techniques: existing reference genomes usually contain a large number of structural errors and/or variations that can surpass the number of assembly errors using the long reads. In most cases we select assemblies by other assemblers with more coverage (~100x) as references. If high quality reference genome is available, thorough evaluations of our algorithm show that DBG2OLC can provide high quality results with fewer structural errors and comparable per-base accuracy. This has been recently demonstrated in the case study of *D. melanogaster* genome by Chakraborty *et al.*\(^3\) who compared our pre-released software with other premier programs for 3GS data. In this paper, we demonstrate results on some other well-studied species and use existing high coverage assemblies as quality checks. On medium to large genomes, DBG2OLC can produce comparably good results with one to two orders of magnitude less time and memory usages than most existing pipelines. A draft assembly (without polishing) of a 3 Gbp *H. sapiens* can be finished in 3 CPU days with our pipeline, utilizing 30 × 3GS and 50x NGS data. This computational time is roughly comparable to many existing NGS assemblers. The time consumption of each step running different genomes can be found in Table 2.

We compared our algorithm results with Celera Assembler (CA, version 8.3rc2), PacBioToCA (in CA8.3rc2}\(^2\), ECTools\(^4\), MHAP (in CA8.3rc2)\(^7\), HGAP (in SMRT Analysis v2.3.0)\(^13\) and Falcon assembler (v0.3.0), which are well recognized as the best-performed genome assemblers for 3GS technologies. Data from PacBio SMRT RS-II

| Species          | Long Read Source | Short Read Assembly (CPU hr) | Compression (CPU hr) | Graph Construction (CPU hr) | Consensus (CPU hr) |
|------------------|------------------|------------------------------|----------------------|-----------------------------|--------------------|
| S. cerevisiae w303 | PacBio           | 0.1                          | 0.03                 | 0.005                       | 2                  |
| A. thaliana ler-0 | PacBio           | 40x                          | 1                    | 0.6                         | 2                  |
| H. sapiens       | PacBio           | 30x                          | 25                   | 37                          | 1600               |
| E. coli K12      | Nanopore         | 30x                          | 0.1                  | 0.02                        | 0.002              |

Table 2. Computation time of each procedure.
| Cov | Assembler | Time (h) | NG50 | Contigs | NGA50 (454) | Identity (454) | Misassemblies (454) | NGA50 (PacBio) | Identity (PacBio) | Misassemblies (PacBio) | Longest | Sum |
|-----|-----------|---------|------|--------|------------|---------------|-------------------|-----------------|-----------------|-------------------|---------|-----|
| 10x | MHAP*     | 36.3    | 554  | 99.68% | 105        | 99.77%        | 6                 | 36,942          | 1,512,911       |                   |         |     |
|     | CA*       | 15.1    | 289  | 97.49% | 134        | 97.46%        | 13                | 448,177         | 12,285,888      |                  |         |     |
|     | PacBioToCA| 173.5   | 898  | 99.88% | 112        | 99.90%        | 6                 | 221,736         | 10,741,663      |                  |         |     |
|     | ECTools   | 24.5    | 169  | 97.76% | 324        | 99.73%        | 29                | 525,820         | 11,785,741      |                  |         |     |
|     | Falcon*   | 1.3     | 675  | 99.23% | 116        | 99.28%        | 4                 | 36,616          | 4,137,485       |                  |         |     |
|     | DBG2OLC   | 1.7     | 168,612 | 99.70% | 408        | 99.81%        | 46                | 1,174,277       | 11,899,604      |                  |         |     |
| 20x | MHAP*     | 17.1    | 364  | 99.70% | 508        | 99.75%        | 24                | 490,764         | 12,356,826      |                  |         |     |
|     | CA*       | 42.4    | 201,649 | 98.83% | 284        | 98.82%        | 21                | 680,599         | 13,052,212      |                  |         |     |
|     | PacBioToCA| 34.2    | 172  | 99.77% | 565        | 99.74%        | 46                | 624,112         | 12,887,799      |                  |         |     |
|     | ECTools   | 3.5     | 93,385 | 99.38% | 345        | 99.42%        | 15                | 281,041         | 10,583,868      |                  |         |     |
|     | Falcon*   | 2.6     | 172,455 | 99.71% | 440        | 99.88%        | 37                | 1,085,773       | 12,476,994      |                  |         |     |
|     | DBG2OLC   | 3.5     | 93,385 | 99.38% | 345        | 99.42%        | 15                | 281,041         | 10,583,868      |                  |         |     |
| 40x | MHAP*     | 36.6    | 65   | 99.91% | 598        | 99.94%        | 24                | 1,090,578       | 12,356,826      |                  |         |     |
|     | CA*       | 115.2   | 114  | 99.66% | 358        | 99.60%        | 11                | 769,189         | 15,171,228      |                  |         |     |
|     | PacBioToCA| 621.7   | 371  | 99.77% | 345        | 99.92%        | 6                 | 742,046         | 11,700,172      |                  |         |     |
|     | ECTools   | 55.8    | 166,945 | 99.79% | 891        | 99.76%        | 7                  | 614,196         | 14,481,947      |                  |         |     |
|     | Falcon*   | 11.2    | 8,578 | 6,908  | 99.90%        | 307                | 307                | 796,762         | 13,052,212      |                  |         |     |
|     | DBG2OLC   | 4.2     | 614,809 | 99.72% | 336        | 99.74%        | 10                | 1,069,920       | 12,116,235      |                  |         |     |
| 80x | MHAP*     | 13.5    | 43   | 99.91% | 526        | 99.95%        | 10                | 1,537,463       | 12,681,860      |                  |         |     |
|     | CA*       | 236.0   | 75   | 99.80% | 319        | 99.90%        | 23                | 1,545,906       | 12,621,393      |                  |         |     |
|     | PacBioToCA| 274.3   | 364  | 99.88% | 45        | 99.91%        | 10                | 233,799         | 11,651,218      |                  |         |     |
|     | ECTools   | 100.9   | 154,348 | 99.71% | 1,470      | 99.76%        | 101               | 881,635         | 15,925,328      |                  |         |     |
|     | Falcon*   | 34.7    | 810,136 | 99.81% | 437        | 99.82%        | 20                | 1,537,463       | 12,681,860      |                  |         |     |
|     | DBG2OLC   | 8.1     | 678,365 | 99.95% | 426        | 99.95%        | 35                | 1,089,897       | 12,209,592      |                  |         |     |

Table 3. Assembly performance comparison on the *S. cerevisiae* genome (genome size: 12 M bp).

*Assemblers that use only 3GS data.

The current leading platform of 3GS technology was used to perform the comparative experiments (50x Illumina MiSeq reads were additionally used for PacBioToCA and DBG2OLC, the two hybrid methods). The experiments are run on a server with eight Intel Xeon E7-8857 v2 CPUs (each has 12 cores) and 2 TB memory. For all DBG2OLC experiments in this paper we used SparseAssembler (Ye et al.) to preassemble 50x Illumina short reads into contigs and then to compress the 3GS reads. Similarly, Celera Assembler was used to assemble the short reads into contigs for ECTools. Unassembled short reads were fed into PacBioToCA according to its specification. At the time of this work, 50x Illumina reads cost less, and also can be obtained more easily, than another assembly generated using MHAP with high coverage data as references. DBG2OLC takes advantage of read data.

It is noteworthy that in our current implementation, most of the computation time (~90%) is spent on the consensus step, in which BLASR is called to align all raw reads to the assembly backbone. Since the alignments are multi-threaded, the wall time can be reduced depending on the available threads. The consensus step is relatively independent in genome assembly and is open to any future improvements and accelerations. The overall computational time of the whole pipeline scales near linearly to the data size, which is a highly valuable property to large-scale genome assembly problems. Using 10x–20x coverage of PacBio sequence data, we obtained assembly N50s that are significantly (>10x) better than Illumina data alone (Table 1). The datasets, commands, and parameters can be found in the Supplementary Materials. We used QUAST 3.0 in its default setting to evaluate the assembly results; these are reported as the NGA50, per-base identity rates and misassembly errors. In analyzing 3GS assembly results, the NGA50 is a measure of the average length of high quality region before reaching a poor quality region in the assembly. The identity rates were calculated by summarizing the single base mismatches and insertion/deletion mismatches. Relocations, inversions, and translocations are regarded as misassembly errors. The alignment dot plots can be found in the Supplementary Materials. The nearly perfect diagonal dot plots indicate that DBG2OLC can produce structurally correct assemblies from as low as 10x long read data.

For the yeast dataset we picked an assembly from 454 data (NCBI Accession No.: GCA_000292815.1) and another assembly generated using MHAP with high coverage data as references. DBG2OLC takes advantage of different sequencing types and obtains the most contiguous results using 10x–40x data with comparable levels of accuracy (Table 3). Some non-hybrid assemblers are not able to fully assemble the yeast genome with 10x–20x
PacBio data. It is also worth mentioning a caveat in many current hybrid error correction approaches. These pipelines use NGS contigs to correct the 3GS reads, which seem to have improved the accuracy of each individual 3GS read. However, the errors in NGS contigs may have corrupted the originally correct 3GS reads and lead to consensus errors in the final assembly. For example, we notice the identity rates of the ECTools assembly are higher when aligned to the 454 reference, contrary to all other pipelines. With high enough coverage (and significantly increased sequencing cost), the 3GS self-correction based assembly methods produce better assembly results. Since our pipeline has a major advantage in low coverage data and efficiency, it is expected to scale well to large genomes where low coverage data and computational time becomes major concerns.

We tested DBG2OLC on other medium to large genomes from PacBio sequencers (Table 4). On the *A. thaliana* genome (120 Mbp), the computations with DBG2OLC finish in one hour, with an additional hour spent constructing the initial NGS contigs. The consensus module takes another 10–20 CPU hours to get the final assembly. The peak memory usage is 6 GB. In comparison, existing pipelines can take over one thousand CPU hours to finish constructing the initial NGS contigs. The consensus module takes another 10–20 CPU hours to get the final assembly. The peak memory usage is 6 GB. In comparison, existing pipelines can take over one thousand CPU hours with problems of this scale. On a large 5x4 human (*H. sapiens*) dataset, DBG2OLC is able to produce an assembly with high contiguity starting from 10x PacBio data (NG50 433 kbp) and DBG contigs generated from 50x Illumina reads (Table S1 in Supplementary Materials). To produce a better assembly, the longest 30x of the reads in this dataset (mean length 14.5 kbp) are selected (Table S2 in Supplementary Materials). DBG2OLC occupies 70 GB memory to store the 17-mer index, and takes 37 CPU hours to compress and align the 30x longest PacBio reads. The pair-wise alignment takes only 3 hours and takes less than 6 hours on the full 54x dataset. The final consensus takes roughly 2000 CPU hours. In an initial report by PacBio scientists, the overlapping process took 405,000 CPU hours. Our final assembly quality (NGA50 = 6 Mbp) is comparable to the state-of-the-art results obtained using orders of magnitude more resources. When evaluating this assembly, QUAST 3.0 can take weeks to finish the full evaluation even on our best workstation. We therefore only align our assembly to the longest 500 Mbp assembly generated by the Pacific Biosciences, and report the NGA50 and identity rate in this portion.

DBG2OLC was also tested on an Oxford Nanopore MinION sequencing dataset (Table 4). According to initial studies, this type of data has higher (up to ~40%) error rates compared to PacBio SMRT sequencing. However, we find DBG2OLC still successfully assembled the *E. coli* into one single contig. The polished assembly has an error rate of 0.23%. The dot plot of the alignment of the assembly to the reference can be found in Fig. 13 of the Supplementary Materials.

Compared with the state-of-the-art assemblers for 3GS technologies, our proposed method produces assemblies with high contiguity using lower sequencing coverage and memory, and is orders of magnitude faster on large genomes. Its combination of different data types leads to both computation and cost efficiency. These advantages are gained from three general and basic design principles: (i) Compact representation of the long reads leads to efficient alignments. (ii) Base-level errors can be skipped, but structural errors need to be detected and cleaned. (iii) Structurally correct 3GS reads are assembled and polished. DBG2OLC is a specific and simple realization of these principles. Interestingly, this implementation builds a nice connection between the two major assembly frameworks, and even though DBG2OLC is majorly developed for 3GS data, this strategy of compression and converting a de Bruijn graph to an overlap graph is general and can be used for popular NGS data. A preliminary showcase on a purely NGS dataset can be found in the Supplementary Materials. The strategy of compressing long reads and performing the most computationally expensive tasks in the compressed domain strikes a balance between the DBG and OLC frameworks.

### Summary and Discussion

In summary, we have built and validated a new *de novo* assembly pipeline that significantly reduces the computational and sequencing requirements of 3GS assembly. We demonstrate that the erroneous long reads can be directly assembled and can lead to significantly improved assembly without base-level error correction. This strategy, first publicly demonstrated in our pre-released pipeline in 2014, has paved the road for several subsequent development attempts on efficient utilization of 3GS data and promises even more efficient 3GS assemblers. Another major finding in developing DBG2OLC is that 3GS technologies generate chimeric reads, and the problem seems to be severer with the PacBio platform. These structural errors lead to tangles in the assembly graph and greatly hamper the assembly contiguity. The most straightforward way to clean up the chimeric reads resorts to multiple sequence alignment, as implemented in DBG2OLC, which leads to a slightly increased coverage requirement. This limitation will serve as the starting point for future development. We conjecture that near perfect assemblies can be reached with even lower coverage if the chimeras/structural errors can be removed.

### Table 4. DBG2OLC assembly performance comparison on various genomes.

| Genome       | Size   | Coverage | NG50  | Contigs | NGA50 | Identity | Misassemblies | Longest | Sum  |
|--------------|--------|----------|-------|---------|-------|----------|---------------|---------|------|
| *A. thaliana*| 120 Mbp| 10x PacBio| 405,464| 881     | 258,924| 99.77%   | 704           | 1,549,329| 119 Mbp|
|              | 20x PacBio| 2,431,755| 306    | 926,138 | 99.90% | 117      | 6,015,430     | 120 Mbp |
|              | 40x PacBio| 3,601,597| 243    | 1,605,981| 99.93% | 131      | 15,473,059    | 129 Mbp |
| *H. sapiens* | 3.0 Gbp| 10x PacBio| 432,739| 16,689  | 347,104| 99.56%   | —             | 3,507,366| 2.97 Gb|
|              | 20x PacBio| 1,866,756| 9,757  | 1,416,766| 99.82% | —        | 14,597,500   | 3.13 Gb |
|              | 30x longest| 6,085,133| 13,095 | 4,124,714| 99.85% | —        | 23,825,526   | 3.21 Gb |
| *E. coli*    | 4.6 Mbp| 30x Nanopore| 4,680,635| 1       | 1,850,974| 99.77%   | 1             | 4,680,635| 4.7 Mbp|

**Scientific Reports** | 6:31900 | DOI: 10.1038/srep31900
References

1. Venter, J. C. et al. The sequence of the human genome. Science 291, 1304–1351, doi: 10.1126/science.1058040 (2001).
2. Koren, S. et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nature biotechnology 30, 693–700, doi: 10.1038/nbt.2280 (2012).
3. Laver, T. et al. Assessing the performance of the Oxford Nanopore Technologies MinION. Biomolecular Detection and Quantification 3, 1–8 (2015).
4. Pacific Biosciences of California, I. Data Release: ~54x Long-Read Coverage for PacBio-only De Novo Human Genome Assembly, http://www.pacb.com/blog/data-release-54x-long-read-coverage-for (Published: 2014, Date of access: 17/03/2016).
5. Ye, C., Ma, Z. S., Cannon, C. H., Pop, M. & Yu, D. W. Exploiting sparseness in de novo genome assembly. BMC Bioinformatics 13 Suppl 6, S1, doi: 10.1186/1471-2105-13-S6-S1 (2012).
6. Koren, S. & Phillippy, A. M. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. Current Opinion in Microbiology 23, 110–120, doi: 10.1016/j.mib.2014.11.014 (2015).
7. Berlin, K. et al. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature biotechnology 33, 623–630 (2015).
8. Salmela, L. & Rivals, E. LoRDEC: accurate and efficient long read error correction. Bioinformatics doi: 10.1093/bioinformatics/btu338 (2014).
9. Lee, H. et al. Error correction and assembly complexity of single molecule sequencing reads. BioRxiv 006395, doi: 10.1101/006395 (2014).
10. Hackl, T., Hedrich, R., Schultz, J. & Forster, F. Proovread: large-scale high-accuracy PacBio correction through iterative short read consensus. Bioinformatics doi: 10.1093/bioinformatics/btu392 (2014).
11. Boetzer, M. & Pirovano, W. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. BMC Bioinformatics 15, 211, doi: 10.1186/1471-2105-15-211 (2014).
12. Deshpande, V., Fung, E. K., Pham, S. & Bafna, V. In Algorithms in Bioinformatics Vol. 8126 Lecture Notes in Computer Science (eds Aaron Darling & Jens Stoye) Ch. 27, 349–363 (Springer Berlin Heidelberg, 2013).
13. Chin, C. S. et al. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. Nature methods 10, 563–569, doi: 10.1038/nmeth.2474 (2013).
14. Ribeiro, F. J. et al. Finished bacterial genomes from shotgun sequence data. Genome research 22, 2270–2277, doi: 10.1101/ gr.141515.112 (2012).
15. English, A. C. et al. Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. PloS one 7, e47768, doi: 10.1371/journal.pone.0047768 (2012).
16. Bashir, A. et al. A hybrid approach for the automated finishing of bacterial genomes. Nature biotechnology 30, 701–707, doi: 10.1038/ nbt.2288 (2012).
17. Au, K. E., Underwood, J. G., Lee, L. & Wong, W. H. Improving PacBio long read accuracy by short read alignment. PloS one 7, e46679, doi: 10.1371/journal.pone.0046679 (2012).
18. Nagarajan, N. & Pop, M. Sequence assembly demystified. Nature reviews. Genetics 14, 157–167, doi: 10.1038/nrg3367 (2013).
19. Myers, E. W. The fragment assembly string graph. Bioinformatics 21 Suppl 2, i79–85, doi: 10.1093/bioinformatics/bti1114 (2005).
20. Pevzner, P. A., Tang, H. & Waterman, M. S. An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences 98, 9748–9753, doi: 10.1073/pnas.010735398 (2001).
21. Miller, J. R. et al. Aggressive assembly of pyrosequencing reads with mates. Bioinformatics 24, 2818–2824, doi: 10.1093/ bioinformatics/btn548 (2008).
22. Simpson, J. T. & Durbin, R. Efficient de novo assembly of large genomes using compressed data structures. Genome research 22, 549–556, doi: 10.1101/gr.126953.111 (2012).
23. Treangen, T. J., Sommer, D. D., Angly, F. E., Koren, S. & Pop, M. Next generation sequence assembly with AMOS. Current Protocols in Bioinformatics, doi: 10.1002/0471250953.b11108s33 (2011).
24. Batzoglou, S. et al. ARACHNE: a whole-genome shotgun assembler. Genome research 12, 177–189, doi: 10.1101/gr.208902 (2002).
25. Myers, G. Efficient local alignment discovery amongst noisy long reads. Algorithms in Bioinformatics 52–67, doi: 10.1007/978-3-662-44753-6 (2014).
26. Chaisson, M. J., Brinza, D. & Pevzner, P. A. De novo fragment assembly with short mate-paired reads: Does the read length matter? Genome research 19, 336–346, doi: 10.1101/gr.79053.108 (2009).
27. Kurtz, S. et al. Versatile and open software for comparing large genomes. Genome Biology 5, R12 (2004).
28. Chaisson, M. & Tesler, G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. BMC Bioinformatics 15, 238 (2012).
29. Smith, T. F. & Waterman, M. S. Identification of Common Molecular Subsequences. J Mol Biol 147, 195–197, doi: 10.1016/0022-8286(81)90087-5 (1981).
30. Ye, C. & Ma, Z. S. Sparc: a sparsity-based consensus algorithm for long erroneous sequencing reads. PeerJ 4, e2016 (2016).
31. Chakraborty, M., Baldwin-Brown, J. G., Long, A. D. & Emerson, J. J. A practical guide to de novo genome assembly using long reads. bioRxiv 029306 (2015).
32. Gurevich, A., Saveliev, V., Vyahhi, N. & Tesler, G. QUAST: quality assessment tool for genome assemblies. Bioinformatics 29, 1072–1075, doi: 10.1093/bioinformatics/btt086 (2013).

Acknowledgements
We appreciate Prof. Mihai Pop, Prof. James Yorke, Dr. Alexey Zimin and their groups at the University of Maryland for supports and helpful discussions. We thank Dr. Sergey Koren and Daniel Liang for helping us to improve our manuscript. This research received funding from the following sources: National Science Foundation of China (Grants No. 61175071, 71473243), the Exceptional Scientists Program and Top Overseas Scholars Program of Yunnan Province, and Yunling Industrial Innovation Grant.

Author Contributions
C.Y. and Z.M. conceived and designed the study; C.Y. and C.M.H. wrote and tested the software; C.Y. and Z.M. wrote the paper; C.M.H. and J.R. participated in performance evaluation and discussion; S.W. helped in running test cases.

Additional Information
Supplementary information accompanies this paper at http://www.nature.com/srep

Competing financial interests: The authors declare no competing financial interests.
