

**SYSTEMS, METHODS, AND MEDIA FOR *DE NOVO* ASSEMBLY OF WHOLE
GENOME SEQUENCE DATA****CROSS REFERENCE**

[001] This application claims the benefit of U.S. Application Serial No. 62/294,184, filed February 11, 2016, and U.S. Application Serial No. 62/332,914 filed May 6, 2016, both of which are hereby incorporated by reference in their entirety.

BACKGROUND OF THE INVENTION

[002] Genomic sequencing holds great promise for the fields of medicine, forensics and biotechnology. A variety of DNA sequencing methods have been developed based on different sequencing chemistries and machines that carry out genomic sequencing have grown more robust and efficient. Technologically, the speed at which raw genomic data can be acquired exceeds the ability to assemble this raw data into a genome—especially diploid or polyploid genomes. Current sequence assembly methods are complicated, require a large amount of processor capability, and take up large amounts of memory.

SUMMARY OF THE INVENTION

[003] Determining the genome sequence of an individual living organism or tissue is of fundamental importance to biology and medicine. Decades of research have yielded a vast array of laboratory and computational approaches directed at this problem. These vary dramatically in their aggregate experimental burden, including input DNA amount, cost, complexity, and timeline, with greater burden tending to yield a higher quality genome sequence.

[004] At the low end, some methods sequence short fragments of DNA, then align the resulting reads to a haploid reference sequence from the same species, to identify differences with it, thereby partially inferring the sequence of the sample. The methods have been used to generate and analyze over a thousand human samples each, yielding extraordinarily deep information across populations. However, these methods may be intrinsically biased by the reference comparison and in general cannot identify sequences that are novel to the given sample or which represent large-scale changes, nor can they distinguish between changes on parental alleles.

[005] By contrast, data (often from long DNA fragments) may be synthesized in the process known as *de novo* assembly, without exploiting a reference sequence, and which is particularly difficult for large and complex genomes. A core challenge is the correct representation of highly similar sequences. A particularly formidable instance appears in eukaryotes, wherein sexual reproduction contributes maternal and paternal chromosome “copies.” While these copies will be quite similar for long stretches, some regions can differ dramatically, leading not only to small

scale differences but often to gene copy number differences. As homologous chromosomes encode separate gene copies, knowledge of their separate sequences is required to understand phenotype.

[006] Yet even for high-end *de novo* laboratory formulations, the standard of the field has been to computationally weave homologous chromosomes, yielding, for each locus, a single haploid consensus that is generally not present in nature. Better, one would generate a haploid assembly together with a phased catalog of differences between the two originating chromosomes.

[007] The disclosure provided herein bridges the gap between low- and high-end approaches, by creating true diploid, *de novo* assemblies, at very low experimental burden. The disclosed technology is also based on genome partitioning, using an automated microfluidic system. The technology is able to generate the entirety of data for an assembly project from one library. Moreover, the disclosed methodology starts from about one nanogram of high molecular weight DNA, about a million times less than alternative approaches. Advantageously, the cost of the data is in the range of low-end methods based on read alignment and no expertise is required for assembly, because the process is automatic.

[008] The present disclosure provides platforms, systems, media, and methods for the *de novo* assembly of whole-genome sequencing reads into a complete genome using short DNA sequence reads. The methods are compatible with any short-read sequencing technology. The methods described herein are advantageously deployed to “phase” contigs into larger sequence blocks and resolve genomic structural variation such as large indels, duplications, and translocations.

[009] The methods of the disclosure possess many advantages compared to other short-read assembly techniques and long-read techniques, such as single molecule real-time (SMRT) sequencing. Some of the advantages include: reductions in input DNA, reduced requirements for sequence coverage, reduced assembly time, reduced processing requirements, commercial scalability due to the ability to run on commoditized computer resources, and an overall increase in efficiency and cost effectiveness. For example, the methods of this disclosure allow for a nearly 180-fold reduction in processing power and a 21-fold reduction in memory utilization, compared to the FALCON assembler from Pacific Biosciences (PacBio), which utilizes SMRT technology. Additionally, long-read sequencing technologies are limited by the average read length of the technology such as 10-20 kb. The technology described herein has been shown to generate fully phased contigs ranging from 85-105 kb and fully phased sequence blocks at least 5 Mb in length.

[010] In one aspect, disclosed herein is a computer-implemented method of *de novo* genome assembly for nucleic acid sequence data generated from a nucleic acid sample of an organism,

the method comprising: generating, by one or more computers, an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read sequence data derived from a common starting sequence share a common tag; generating, by the one or more computers, a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity; generating, by the one or more computers, a global assembly based on the plurality of local assemblies; cleaning, by the one or more computers, the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and generating, by the one or more computers, a phased genome assembly based on the global assembly by utilizing the tags to separate phased nucleotide sequence; wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence. In certain instances, the genome is diploid. In certain instances, the short-read sequence data is generated from a single sequencing library. In certain instances, the short-read sequence data results in 50x or less coverage of the genome of the organism. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb. The initial assembly can be an initial assembly graph. In certain embodiments, the initial assembly graph is generated by: identifying a plurality of k-mers that have a high probability of being present in the genome of the organism; using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$. The method can further comprise revising, by the one or more computers, the initial assembly graph by: eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and filling in gaps in the initial assembly graph by consulting the original short-read sequence data. K can be between 24 and 96. The plurality of local assemblies can be generated by: using the initial assembly graph as an interim reference; identifying edges of unambiguous sequence; identifying neighboring edges sharing a number of tags above a threshold number of tags; and bringing together edges of unambiguous sequence with the identified neighboring edges. The global assembly can be generated by: identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and bringing together the z-mers in the plurality of local assemblies. Z can be between 100 and 300. The short-read sequence data can be generated from

less than 10 ng of DNA input material. The short-read sequence data can be generated from less than 2 ng of DNA input material. In some embodiments, the assembly can be completed in less than 60 minutes. In these embodiments, the one or more computers can comprise less than 512 GB of storage; in certain embodiments the one or more computers can comprise less than 60 GB of storage. In certain instances, the assembly is completed in less than 20 minutes. In these instances, the one or more computers comprise less than 512 GB of storage. In certain instances, the one or more computers comprise less than 60 GB of storage. In certain instances, the organism is a human being. In certain instances, the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly. In certain instances, the one or more computers occupy one cubic foot or less of physical space.

[011] In another aspect, disclosed herein is a computer-implemented system comprising: a digital processing device comprising: at least one processor, an operating system configured to perform executable instructions, a memory, and a computer program including instructions executable by the digital processing device to create a *de novo* genome assembly application for nucleic acid sequence data generated from a nucleic acid sample of an organism, the application comprising: a first software module generating an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read sequence data derived from a common starting sequence share a common tag; a second software module generating a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity; a third software module generating a global assembly based on the plurality of local assemblies; a fourth software module cleaning the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and a fifth software module generating a phased genome assembly based on the global assembly by utilizing the tags to separate homologous phased nucleotide sequence; wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence. In certain instances, the genome is diploid. In certain instances, the short-read sequence data is generated from a single sequencing library. In certain instances, the short-read sequence data results in 50x or less coverage of the genome of the organism. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb. In certain instances, the initial assembly is an initial assembly graph. In certain instances the software module generates an initial assembly

graph generates the initial assembly graph by identifying a plurality of k-mers that have a high probability of being present in the genome of the organism; using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$. In certain instances, the software module generates an initial assembly graph revises the initial assembly graph by: eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and filling in gaps in the initial assembly graph by consulting the original short-read sequence data. K can be between 24 and 96. The plurality of local assemblies can be generated by: using the initial assembly graph as in interim reference; identifying edges of unambiguous sequence; identifying neighboring edges sharing a number of tags above a threshold number of tags; and bringing together edges of unambiguous sequence with the identified neighboring edges. The global assembly can be generated by: identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and bringing together the z-mers in the plurality of local assemblies. Z can be between 100 and 300. The short-read sequence data can be generated from less than 10 ng of DNA input material. The short-read sequence data can be generated from less than 2 ng of DNA input material. In certain instances, the assembly is completed in less than 60 minutes. In some embodiments, the assembly can be completed in less than 60 minutes. In these embodiments, the one or more computers can comprise less than 512 GB of storage; in certain embodiments the one or more computers can comprise less than 60 GB of storage. In certain instances, the assembly is completed in less than 20 minutes. In these instances, the one or more computers comprise less than 512 GB of storage. In certain instances, the one or more computers comprise less than 60 GB of storage. In certain instances, the organism is a human being. In certain instances, the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly. In certain instances, the digital processing device occupies one cubic foot or less of physical space.

[012] In another aspect, disclosed herein is a non-transitory computer-readable storage media encoded with a computer program including instructions executable by a processing device to create a *de novo* genome assembly application for nucleic acid sequence data generated from a nucleic acid sample of an organism, the application comprising: a first software module generating an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read

sequence data derived from a common starting sequence share a common tag; a second software module generating a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity; a third software module generating a global assembly based on the plurality of local assemblies; a fourth software module cleaning the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and a fifth software module generating a phased genome assembly based on the global assembly by utilizing the tags to separate homologous phased nucleotide sequence; wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence. In certain instances, the genome is diploid. In certain instances, the short-read sequence data is generated from a single sequencing library. In certain instances, the short-read sequence data results in 50x or less coverage of the genome of the organism. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads. In certain instances, the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb. In certain instances, the initial assembly is an initial assembly graph. In certain instances, the software module generating an initial assembly graph generates the initial assembly graph by: identifying a plurality of k-mers that have a high probability of being present in the genome of the organism; using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$. In certain instances, the software module generating an initial assembly graph revises the initial assembly graph by: eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and filling in gaps in the initial assembly graph by consulting the original short-read sequence data. K can be between 24 and 96. The plurality of local assemblies can be generated by: using the initial assembly graph as in interim reference; identifying edges of unambiguous sequence; identifying neighboring edges sharing a number of tags above a threshold number of tags; and bringing together edges of unambiguous sequence with the identified neighboring edges. The global assembly can be generated by: identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and bringing together the z-mers in the plurality of local assemblies. Z can be between 100 and 300. The short-read sequence data can be generated from less than 10 ng of DNA input material. The short-read sequence data can be generated from less than 2 ng of DNA input material. In certain instances, the assembly is completed in less than 60 minutes. In some embodiments, the assembly can be completed in less than 60 minutes. In these embodiments, the one or more computers can comprise less than 512 GB of storage; in certain

embodiments the one or more computers can comprise less than 60 GB of storage. In certain instances, the assembly is completed in less than 20 minutes. In these instances, the one or more computers comprise less than 512 GB of storage. In certain instances, the one or more computers comprise less than 60 GB of storage. In certain instances, the organism is a human being. In certain instances, the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly. In certain instances, the digital processing device occupies one cubic foot or less of physical space.

BRIEF DESCRIPTION OF THE DRAWINGS

[013] **Fig. 1** shows a non-limiting example of an alignment generated using the methods of this disclosure which preserves information regarding both SNPs and structural variants.

[014] **Fig. 2** shows a non-limiting example of a genomic assembly process.

[015] **Fig. 3** shows a non-limiting example of alternative schematic illustration of a genomic assembly process with megabubbles and microstructure.

[016] **Fig. 4** shows a non-limiting example of a genomic assembly process.

[017] **Fig. 5** shows a non-limiting example of outputs available from the genomic assembly process.

[018] **Fig. 6** shows a non-limiting example of a digital processing device; in this case, a device with one or more CPUs, a memory, a communication interface, and a display.

[019] **Fig. 7** shows a non-limiting example of de novo assembly.

[020] **Fig. 8** shows a non-limiting example of preponderance of errors near long homopolymers.

[021] **Fig. 9** shows a non-limiting example of computing architecture of a *de novo* assembler.

DETAILED DESCRIPTION OF THE INVENTION

Certain definitions

[022] Unless otherwise defined, all technical terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which this invention belongs. As used in this specification and the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise. Any reference to “or” herein is intended to encompass “and/or” unless otherwise stated.

[023] As used herein, a “phased” assembly or sequence refers to creating an assembly where nucleotide data is accurately placed in *cis* at a particular locus such as a chromosome or other subgenomic interval. This could include for example properly resolving a haplotype, diploid genome, polyploid genome, different samples, different cells, different organisms, or even structural variants of a haploid genome such as large indels, translocations, and fusions.

[024] As used herein, “next-generation sequencing” refers to any technology that results in large amounts nucleotide sequence data, generally greater than 1 gigabase in a 24-hour period. No limiting exemplar systems are available from Illumina (San Diego, CA), Life Technologies (Carlsbad, CA), and Pacific Biosystems (Menlo Park, CA).

Advantages of *de novo* assembly

[025] The present disclosure is directed to novel approaches for obtaining and analyzing genomic information from biological samples that allow for improved genetic assembly of the sequence data so derived. In particular, the methods, systems, and media described herein involve the preparation of sequencing libraries that are encoded to preserve long range sequence context of individual, and much shorter stretches of determined sequence information as output by a given sequence system (referred to herein as “reads”). This long-range sequence context allows for the ordering of reads, regardless of length, over much longer sequence contexts than the length of the individual reads, e.g., 2x, 5x, 10x, 100x, 1000x, or even longer contiguous sequence stretches, as well as any range of lengths between these. Such long range contexts can be over contiguous sequence stretches that are on the order of 10 kb, 100 kb, 200 kb, 500 kb, 1 Mb, 2 Mb, 3 Mb, 4 Mb, 5 Mb, or even longer, as well as any range of lengths between these.

[026] By providing such long-range sequence context, one can identify sequence characteristics that can generally be derived from long contiguous stretches of sequence, such as identifying large scale structural variations, determining haplotype information, and the like. All of these benefits are particularly useful in being able to accurately assemble true genome sequences, and moreover, true diploid or polyploid genome sequences, whether operating from a reference sequence or assembling the particular diploid genome, *de novo*. The platforms, systems, media, and methods described herein provide benefits that have been lacking in prior sequencing and *de novo* assembly processes, including use of: (i) very small amounts of input, (ii) lower sequence coverage, (iii) low cost, and (iv) computational efficiency.

[027] The platforms, systems, media, and methods described herein allow assembly of a single sequencing library into a complete genome. A sequencing library is created beginning with a DNA sample that contains at least one entire genome, chromosome, or fragment of DNA. The DNA is then prepared by segmenting the DNA into large fragments typically greater than 1 kilobase, often greater than 10, 50, or 100 kilobases. These segments are then physically partitioned with each partition comprising a DNA fragment and a unique barcode or tag that serves to identify the DNA fragment, and any smaller fragments that are generated from the larger fragment. The barcode can be included on an oligonucleotide. The oligonucleotide can be releasably connected to a solid structure such as a bead or microcapsule. In some embodiments, the oligonucleotide can comprise a random sequence, a sequence complimentary to a target

sequence, a sequence from a primer to attach, or a universal priming site.

[028] Once the large DNA fragments are partitioned with a unique tag, smaller fragments are generated that incorporate the tag, which are then pooled into a library and sequenced. The library can be sequenced by any next-generation sequencing technology. In certain embodiments, the DNA sequence data is generated, by way of non-limiting examples, by pyrosequencing, sequencing by synthesis, sequencing by ligation, ion semiconductor sequencing, or single molecule real time sequencing. In certain embodiments, the DNA sequence data is generated by any technology capable of generating 1 gigabase of nucleotide reads per 24-hour period.

[029] With possession of the sequence data from the reads of the various fragments and their associated barcode sequences, an assembly of a much larger sequence can be performed *de novo*. The sequencing reaction and the assembly step need not be carried out by the same individual or entity. The sequence data can be obtained from a third party. As described herein, this may be accomplished in a manner that preserves computational efficiency while permitting complex assembly processes to be completed. Exemplary methods of segmenting, partitioning, and tagging nucleic acid molecules are described in U.S. Patent Application No. 14/175,935, the full disclosure of which is hereby incorporated by reference in their entirety.

[030] To call variants of all types, in principle one could align the assembly to the reference sequence, and then read off the variants. Aligning an assembly has tremendously greater specificity than aligning reads, thus eliminating most artifacts associated with standard reference-based analyses. **Fig. 1** is an example from the HGP assembly. It shows a SNP, two small insertions and a 554 base insertion on one allele. The other allele shows the same insertion. There are no BLAST hits to GenBank. If you look at NA12878 instead, the same insertion is found, but only on one allele.

Computational efficiency

[031] When assembling genomic context from short-read sequence data, the ultimate assembly is a consensus assembly of samples, cells, or of different haplotypes/chromosomes. As such, creating a true diploid assembly of a genome even from long range sequences has to date been largely impossible. In particular, these assemblies typically may present an average assembly of a diploid genome, rather than specifically identifying haploid variants. In accordance with the methods described herein, one can obtain separate assemblies of different variants of otherwise homologous loci, e.g., from each haplotype, chromosome, cell, or sample.

[032] Conventional assembly processes, due to the nature of shorter sequence reads and the computationally intensive assembly processes, place tremendous stress on computational infrastructures. In some cases, as described herein, processes are employed that have

significantly more efficient computational processes than those conventionally used. In particular, in the processes described herein, an initial assembly graph is prepared. This initial assembly amounts to a “rough sketch” assembly and temporarily ignores areas of unresolved complexity, e.g., regions that may be ambiguous upon first glance, thus, preserving computing power. Once this initial assembly is created, areas of ambiguity may then be further processed by employing the barcoded sequencing data to create an accurate assembly of the areas of ambiguity. This allows k-mers derived from sequencing reads created during the assembly process to be addressable to a certain chromosome, cell, population, haplotype, etc. This allows efficient resolution of ambiguity – by reducing overall complexity in the assembly – conserving computational resources such as RAM, ROM, or processor cycles. In certain embodiments, the methods described herein, can reduce the amount of RAM necessary for assembly to below 512, 256, 128, 64, 32, 16, 8, or 4 gigabytes.

[033] Using barcoded segmented nucleic acids contributes to the efficient assembly of genome sequences by contextualizing k-mers derived from sequence reads. By way of example, in a conventional assembly process not employing barcodes, following cleaning of the global assembly, most loci in the global assembly will represent two or more chromosomal loci (e.g., a maternally inherited and a paternally inherited chromosome). Using the barcodes, these loci are now separated from each other. This process will separate homologous chromosomes and also resolve complex segmental duplications. This is schematically illustrated in **Fig. 2**. As shown, a partially assembled sequence is illustrated with unassembled portion or portions, denoted by the boxes (step I). Neighborhood assembly is used to exploit the barcoded or tagged adjacent sequences to reach into the black box to provide assembly of the sequence therein. Moreover, where that sequence represents differing haplotype or phased sequence information, e.g., as shown by the dual arrows (step II), these barcodes will inform that as well, yielding, e.g., the separate variant assemblies within these regions. From the separate assemblies, one can generate true diploid assemblies at homologous loci (step III). The barcoded reads are placed back on the assembly and loci are identified whose exact sequence is not known with certainty, and so marked. The global assembly graph is now aligned to a reference sequence. First edges are aligned individually. Where these alignments are inconsistent (at points where two edges meet), the inconsistency is resolved by aligning the concatenations of these edges. The reference alignments together with quality markups now imply presence or absence of particular alleles for variants of all types (single base and structural) – and uncertainty where warranted. This is the “correct” version of variant calling attempted by traditional reference-centric approaches. Two or more related samples may be assembled together, yielding a single graph, from which the precise relationship between the samples at each locus may be inferred. This includes the case of tumor

and normal (e.g., from one mixed clinical sample), and the case of *de novo* mutations in a child, seen by sequencing a family, including recombinations. Direct comparisons will reveal differences in loci that are absent from the reference sequence.

[034] In certain implementations, described herein, an initial *de novo* assembly is created using short-read nucleotide sequencing technology. This *de novo* assembly can be based upon short-read sequence data. The short-read sequence data may be from reads of less than 300, 250, 200, 150, 100, 75, or 50 base pairs, including increments therein. The short-read sequence data may be from paired end reads. An initial *de novo* assembly graph may be created in a fashion that results in memory usage being small compared to the genome size. This optimizes the speed of the initial assembly. Memory usage does not increase proportionally with input data amount. The keys to the technologies are:

- (a) Based on frequency and base quality scores, k-mers are identified that have a high probability of being present in the genome –thus the number of these k-mers are dictated by the genome size.
- (b) K-mers are naturally coalesced during generation by bringing together those k-mers sharing a common minimum p-mer ($p < k$) – this reduces memory usage by an order of magnitude.
- (c) Subsequent assembly operations treat the initial assembly as an interim “reference sequence” and thus have very low memory requirements.

Given an edge e in the assembly, representing an unbranched sequence in the sample, its neighboring edges are found – these are the edges sharing a minimum number of barcodes with e . In certain embodiments the edges share at least 1, 2, 3, 4, 5, 6, 7, 8, 9, or 10 barcode sequences. These sequences are then assembled contiguously, yielding local assemblies (neighborhoods). This process also fills in gaps that were missing from the initial assembly. All neighborhoods are then assembled using a very large k-mer value. The resulting assembly is again a graph. The global assembly is now cleaned by removing connections inconsistent with the barcodes.

Assembly method

[035] K-mers that are generated from short sequence reads (read k-mers), approximate the graph that would be obtained by collapsing the sample genome along perfect k-mer matches. Prefiltering can be performed to exclude read k-mers that are likely wrong because, for example, they occur in only one barcode, or have low quality scores or are infrequent in the reads. Then, an initial graph is constructed from the k-mers. The edges of this graph are labeled with DNA sequences, representing unbranched paths (called “unipaths”) in a *De Bruijn* graph. Next, patch

gaps may be performed on this graph. Finally, “hanging ends” may be trimmed from the graph, yielding an initial assembly.

[036] Read k-mers used to assemble the initial assembly can be any length that facilitates assembly, and may vary dependent upon the size of the genome being assembled. The read k-mer can be greater than 1, 10, 20, 30, 40, 40, 60, 70, 80, 90, or 100 base pairs, including increments therein. In certain instances, the read k-mer can be less than 10, 20, 30, 40, 40, 60, 70, 80, 90, or 100, base pairs in length, including increments therein. Generally, a k-mer between 30 and 50 base pairs is ideal for the initial *de novo* assembly. In certain instances, the k-mer may be between 40 and 50 base pairs. The k-mer may be 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, or 50 base pairs in length. Ideally the k-mer is a multiple of four.

[037] After the initial assembly is created a new global assembly (super graph) is constructed. The super graph’s edges are labeled by paths in the initial assembly (and are thus represented as sequences of integers). Formally this graph has the same K value as in the initial assembly, but is resolved to a higher K value. This is achieved by finding paths in the base graph that are closures of read pairs, which therefore have length of approximately 100, 200, 300, 400, or 500 k-mers. These paths are formally aligned along long perfect overlaps to yield the super graph. These second larger k-mers are denoted as z-mers.

[038] After the global assembly the barcodes are used to locate and fill gaps. Gaps can be visualized as divergent “bubbles” due to a branch point in the unipath of the *De Bruijn* graph. Referring to **Fig. 3**, each scaffold has megabubbles, representing fully phased portions of the assembly. Successive megabubbles are not phased relative to one another. In general, each edge shown in the scaffold diagram contains within its “micro-structure,” describing sequences that are not fully determined. These bubbles as shown in **Fig. 3** can be derived from different contexts of the reads from which the k-mers are generated. The different contexts could be different chromosomes, samples or structural variants such as indels, duplications and translocations. The barcodes allow inference of a pool of reads that should cover a gap in the assembly, comprising of all the reads with a certain barcode sequence. After creating a local assembly from this pool, it can be reinserted into the global assembly. This is exemplified in **Fig. 4** where one parental allele is in solid dotted line **401**, while the other is in circle dotted line **402**. Shown is one barcoded molecule landing on all the alleles in solid dots, and one landing on all alleles in circle dots, thus pulling them apart as **411** and **412**, and thus properly phasing the sequence context. In reality, many barcoded molecules collude to execute this phasing operation. In certain embodiments, the methods described herein can create phased blocks of nucleotide sequence comprising over 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or more megabases. These levels of phasing can be achieved at coverage levels of 50x or below, 40x or below, or 30x or below for a 3 gigabase (human) genome.

Output

[039] Referring to **Fig. 5** the algorithm can output information to a user in many ways. Output **501** shows that data can be output in a “raw” style wherein very assembly edge (including microbubble arms and gaps) appears as a separate FASTA record. This is how the software sees the assembly. Output **502** shows that data can be output in a “megabubble style” wherein each megabubble arm corresponds to a single FASTA record, as does each intervening sequence. In some cases, the user can set a threshold to visualize megabubble or intervening sequences only if they occur over a certain stretch of the genome or exceed a certain size threshold. Output **503** shows that data can be output in a “psuedohap style,” a single record per scaffold. Megabubble arms are chosen arbitrarily so many records mix maternal and paternal alleles. Output **504** shows that data can be output in a “psuedohap2 style,” wherein, for each scaffold, two “parallel” pseudo-haplotypes are created and placed in separate FASTA files.

Digital processing device

[040] The methods, systems, and media described herein include at least one digital processing device, or use of the same. The digital processing device includes one or more hardware central processing units (CPUs) or general purpose graphics processing units (GPGPUs) that carry out the device’s functions. The digital processing device further comprises an operating system configured to perform executable instructions. The digital processing device is optionally connected a computer network. By way of example, the digital processing device is optionally connected to the Internet such that it accesses the World Wide Web. By way of further example, the digital processing device is optionally connected to a cloud computing infrastructure. By way of further example, the digital processing device is optionally connected to an intranet. By way of still further example, the digital processing device is optionally connected to a data storage device.

[041] In accordance with the description herein, suitable digital processing devices include, by way of non-limiting examples, commercial server computers and desktop computers known to those of skill in the art. Suitable digital processing devices also include devices custom-built using hardware and techniques known to those of skill in the art.

[042] The digital processing device includes an operating system configured to perform executable instructions. The operating system is, for example, software, including programs and data, which manages the device’s hardware and provides services for execution of applications. Those of skill in the art will recognize that suitable server operating systems include, by way of non-limiting examples, FreeBSD, OpenBSD, NetBSD®, Linux, Apple® Mac OS X Server®, Oracle® Solaris®, Windows Server®, and Novell® NetWare®. Those of skill in the art will recognize that suitable personal computer operating systems include, by way of non-limiting

examples, Microsoft® Windows®, Apple® Mac OS X®, UNIX®, and UNIX-like operating systems such as GNU/Linux®. In some cases, the operating system is provided by cloud computing.

[043] The device includes a storage and/or memory device. The storage and/or memory device is one or more physical apparatuses used to store data or programs on a temporary or permanent basis. In some embodiments, the device is volatile memory and requires power to maintain stored information. In some cases, the device is non-volatile memory and retains stored information when the digital processing device is not powered. The non-volatile memory may comprise flash memory, dynamic random-access memory (DRAM), ferroelectric random access memory (FRAM), phase-change random access memory (PRAM), or the like. In other cases, the device is a storage device including, by way of non-limiting examples, CD-ROMs, DVDs, flash memory devices, magnetic disk drives, magnetic tapes drives, optical disk drives, cloud computing-based storage, and the like. In various cases, the storage and/or memory device is a combination of devices such as those disclosed herein.

[044] The digital processing device optionally includes a display to send visual information to a user. Suitable displays include liquid crystal displays (LCD), thin film transistor liquid crystal displays (TFT-LCD), organic light emitting diode (OLED) displays (including passive-matrix OLED (PMOLED) and active-matrix OLED (AMOLED) displays), plasma displays, video projectors, and head-mounted displays (such as a VR headset) in communication with the digital processing device. Suitable VR headsets include, by way of non-limiting examples, HTC Vive, Oculus Rift, Samsung Gear VR, Microsoft HoloLens, Razer OSVR, FOVE VR, Zeiss VR One, Avegant Glyph, Freefly VR headset, and the like. In various cases, the display is a combination of devices such as those disclosed herein.

[045] The digital processing device optionally includes one or more input devices to receive information from a user. Suitable input devices include keyboards, pointing devices (including, by way of non-limiting examples, a mouse, a trackball, a track pad, a joystick, a game controller, and a stylus), touch screens or a multi-touch screens, microphones to capture voice or other sound input, video cameras or other sensors to capture motion or visual input. In particular cases, the input device is a Kinect, Leap Motion, or the like. In various cases, the input device is a combination of devices such as those disclosed herein.

[046] Referring to **Fig. 6**, in a particular embodiment, an exemplary digital processing device **601** is programmed or otherwise configured to assemble short-read DNA sequences into fully phased complete genomic sequences. The device **601** can regulate various aspects of the sequence assembly methods of the present disclosure, such as, for example, performing initial alignments, quality checking, performing subsequent alignments, resolving ambiguity, and

phasing heterozygous loci. In this embodiment, the digital processing device **601** includes a central processing unit (CPU, also “processor” and “computer processor” herein) **605**, which can be a single core or multi core processor, or a plurality of processors for parallel processing. The digital processing device **601** also includes memory or memory location **610** (e.g., random-access memory, read-only memory, flash memory), electronic storage unit **615** (e.g., hard disk), communication interface **620** (e.g., network adapter) for communicating with one or more other systems, and peripheral devices **525**, such as cache, other memory, data storage and/or electronic display adapters. The memory **610**, storage unit **615**, interface **620** and peripheral devices **625** are in communication with the CPU **605** through a communication bus (solid lines), such as a motherboard. The storage unit **615** can be a data storage unit (or data repository) for storing data. The digital processing device **601** can be operatively coupled to a computer network (“network”) **630** with the aid of the communication interface **620**. The network **630** can be the Internet, an internet and/or extranet, or an intranet and/or extranet that is in communication with the Internet. The network **630** in some cases is a telecommunication and/or data network. The network **630** can include one or more computer servers, which can enable distributed computing, such as cloud computing. The network **630**, in some cases with the aid of the device **601**, can implement a peer-to-peer network, which may enable devices coupled to the device **601** to behave as a client or a server.

[047] Continuing to refer to **Fig. 6**, the CPU **605** can execute a sequence of machine-readable instructions, which can be embodied in a program or software. The instructions may be stored in a memory location, such as the memory **610**. The instructions can be directed to the CPU **605**, which can subsequently program or otherwise configure the CPU **605** to implement methods of the present disclosure. Examples of operations performed by the CPU **605** can include fetch, decode, execute, and write back. The CPU **605** can be part of a circuit, such as an integrated circuit. One or more other components of the device **601** can be included in the circuit. In some cases, the circuit is an application specific integrated circuit (ASIC) or a field programmable gate array (FPGA).

[048] Continuing to refer to **Fig. 6**, the storage unit **615** can store files, such as drivers, libraries and saved programs. The storage unit **615** can store user data, e.g., user preferences and user programs. The digital processing device **601** in some cases can include one or more additional data storage units that are external, such as located on a remote server that is in communication through an intranet or the Internet.

[049] Continuing to refer to **Fig. 6**, the digital processing device **601** can communicate with one or more remote computer systems through the network **630**. For instance, the device **601** can communicate with a remote computer system of a user. Examples of remote computer systems

include personal computers (e.g., portable PC), slate or tablet PCs (e.g., Apple[®] iPad, Samsung[®] Galaxy Tab), telephones, Smart phones (e.g., Apple[®] iPhone, Android-enabled device, Blackberry[®]), or personal digital assistants.

[050] Methods as described herein can be implemented by way of machine (e.g., computer processor) executable code stored on an electronic storage location of the digital processing device **601**, such as, for example, on the memory **610** or electronic storage unit **615**. The machine executable or machine readable code can be provided in the form of software. During use, the code can be executed by the processor **605**. In some cases, the code can be retrieved from the storage unit **615** and stored on the memory **610** for ready access by the processor **605**. In some situations, the electronic storage unit **615** can be precluded, and machine-executable instructions are stored on memory **610**.

Non-transitory computer readable storage medium

[051] The methods, systems, and media disclosed herein include one or more non-transitory computer readable storage media encoded with a program including instructions executable by the operating system of an optionally networked digital processing device. In some cases, a computer readable storage medium is a tangible component of a digital processing device. In other cases, a computer readable storage medium is optionally removable from a digital processing device. A computer readable storage medium includes, by way of non-limiting examples, CD-ROMs, DVDs, flash memory devices, solid state memory, magnetic disk drives, magnetic tape drives, optical disk drives, cloud computing systems and services, and the like. In some cases, the program and instructions are permanently, substantially permanently, semi-permanently, or non-transitorily encoded on the media.

Computer program

[052] The methods, systems, and media disclosed herein include at least one computer program, or use of the same. A computer program includes a sequence of instructions, executable in the digital processing device's CPU, written to perform a specified task. Computer readable instructions may be implemented as program modules, such as functions, objects, Application Programming Interfaces (APIs), data structures, and the like, that perform particular tasks or implement particular abstract data types. In light of the disclosure provided herein, those of skill in the art will recognize that a computer program may be written in various versions of various languages.

[053] The functionality of the computer readable instructions may be combined or distributed as desired in various environments. In some cases, a computer program comprises one sequence of instructions. In other cases, a computer program comprises a plurality of sequences of instructions. In some cases, a computer program is provided from one location. In other cases, a

computer program is provided from a plurality of locations. In various cases, a computer program includes one or more software modules. In various implementations, a computer program includes, in part or in whole, one or more web applications, one or more mobile applications, one or more standalone applications, one or more web browser plug-ins, extensions, add-ins, or add-ons, or combinations thereof.

Standalone application

[054] In some embodiments, a computer program includes a standalone application, which is a program that is run as an independent computer process, not an add-on to an existing process, e.g., not a plug-in. Those of skill in the art will recognize that standalone applications are often compiled. A compiler is a computer program(s) that transforms source code written in a programming language into binary object code such as assembly language or machine code. Suitable compiled programming languages include, by way of non-limiting examples, C, C++, Objective-C, COBOL, Delphi, Eiffel, Java™, Lisp, Python™, Visual Basic, and VB .NET, or combinations thereof. Compilation is often performed, at least in part, to create an executable program. In some cases, a computer program includes one or more executable compiled applications.

Software modules

[055] The methods, systems, and media disclosed herein include software, server, and/or database modules, or use of the same. In view of the disclosure provided herein, software modules are created by techniques known to those of skill in the art using machines, software, and languages known to the art. The software modules disclosed herein are implemented in a multitude of ways. In various implementations, a software module comprises a file, a section of code, a programming object, a programming structure, or combinations thereof. In other various implementations, a software module comprises a plurality of files, a plurality of sections of code, a plurality of programming objects, a plurality of programming structures, or combinations thereof. By way of non-limiting examples, the one or more software modules comprise a web application, a mobile application, and a standalone application. In some cases, software modules are in one computer program or application. In other cases, software modules are in more than one computer program or application. In some cases, software modules are hosted on one machine. In other cases, software modules are hosted on more than one machine. In particular cases, software modules are hosted on one or more cloud computing platforms and/or services. In some cases, software modules are hosted on one or more machines in one location. In other cases, software modules are hosted on one or more machines in more than one location.

Databases

[056] The methods, systems, and media disclosed herein include one or more databases, or use of the same. In view of the disclosure provided herein, those of skill in the art will recognize that many databases are suitable for storage and retrieval of sequence and graph information. Suitable databases include, by way of non-limiting examples, relational databases, non-relational databases, object oriented databases, object databases, entity-relationship model databases, associative databases, and XML databases. Further non-limiting examples include SQL, PostgreSQL, MySQL, Oracle, DB2, and Sybase. In some cases, a database is internet-based. In further cases, a database is web-based. In still further cases, a database is cloud computing-based. In other cases, a database is based on one or more local computer storage devices.

EXAMPLES

[057] The following illustrative examples are representative of embodiments of the software applications, systems, and methods described herein and are not meant to be limiting in any way.

Example 1 – Diploid de novo sequence assembly using tagged reads generates long stretches of genomic phasing

[058] To thoroughly test the methods of this disclosure on different genomes, this example generated datasets from seven individuals of varied ancestry and sex and three mixed breed dogs as shown in **Table 1**. These datasets demonstrate genomes of sizes 3.2 and 2.5 Gb, respectively. All were created from DNA of size ≥ 80 kb.

Table 1																	
input							continuity				comparison to truth data						run time
											same	parental	reference				
ID	sample	ethnicity	sex	data description	X	F	N50 contig (kb)	N50 phase block (Mb)	N50 scaffold (Mb)	gappiness	N50 perfect stretch (kb)	phasing error %	missing k-mers (%)		discrepant at given distance (%)		wall clock (days)
													haploid	diploid	1 Mb	10 Mb	
A	NA19238	Yoruban	F	one 10X library	56	115	114.6	8.0	18.7	2.1			14.4	10.0	1.3	0.8	2.1
B	NA19240	Yoruban	M	one 10X library	56	125	118.8	9.3	16.4	2.3		0.086	14.9	10.4	1.3	0.7	
C	HG00733	Puerto Rican	F	one 10X library	56	106	124.4	3.4	17.7	2.0		0.089	13.2	9.8	1.0	1.3	2.2
D	HG00512	Chinese	M	one 10X library	56	102	113.2	2.8	15.4	2.2			13.5	10.0	1.4	0.5	1.9
E	NA24385	Ashkenazi	M	one 10X library	56	120	106.7	4.3	15.0	2.6		0.053	14.0	9.6	1.3	0.7	2.1
F	HGP	European	M	one 10X library	56	139	120.2	4.5	18.6	2.5	19.1		12.3	8.8	1.9	0.9	
G	NA12878	European	F	one 10X library	56	92	119.7	2.7	17.5	1.9	18.7	0.018	13.2	9.7	1.0	0.6	2.5
H	NA12878	European	F	unknown number of PacBio	46		1594.2		25.4	4.6			18.0		0.5	2.0	

				libraries plus BioNano Genomics data ^{A2}													
I	NA12878	European	F	6 libraries (fragment, jumping, 10X) ^{A3}	160		12.3		30.1	10.2			19.7		1.1	7.1	
J	NA12878	European	F	9 libraries (fragment, jumping, Fosmid, HiRise) ^{DOVE}	150		43.6		42.8	0.6			14.8		5.6	6.0	
K	NA24385	Ashkenazi	M	7 PacBio libraries	71		4525.2		4.5	0.0			11.8		2.2	17.9	
L	NA24143	Ashkenazi	F	2 PacBio libraries	30		1048.4		1.0	0.0			14.3		15.2	N/A	
M	YH	Chinese	M	~18,000 Fosmid pools and 6 fragment and jumping libraries, Illumina sequenced, plus Complete Genomics data ^{YH}	702		52.5	0.5	23.2	1.5			10.4	1.2	1.6		
N	Cuishle	Goldendoodle	M	one 10X library	56	90	72.2	4.3	11.4								1.4
O	Ollie	mutt	M	one 10X library	56	83	64.6	4.9	12.2								1.5
P	Ruby	mutt	F	one 10X library	56	84	73.0	2.9	12.8								1.3

Table 1 Legend: All statistics were computed after removing scaffolds shorter than 10 kb.

Comparisons to reference use GRCh37 (chr1-22,X,Y), with chrY excluded for female samples.

ID: identifier of assembly in this table. **Sample:** type of starting material. HGP is from the living anonymous donor to the Human Genome Project for libraries RPCI 1,3,4,5 (available at <http://bacpac.chori.org/library.php?id=1>), for which 340 Mb of finished sequence are in GenBank. Dogs and HGP were from blood, others are Coriell cell lines. **Ethnicity:** ethnicity, or for dogs, breed origin. **Sex:** sex of sample. **Data description:** capsule description of data type. **X:** estimated coverage of genome by sequence reads. For assemblies of this work, reads were 2x150; 1200M reads were used for each human assembly; 940M reads for each dog assembly; dogs and sample G were sequenced on HiSeq 2500 in rapid run mode; other samples were sequenced on HiSeq X. **F:** inferred length-weighted mean fragment length of DNA in kb. **N50 contig size:** N50 size of FASTA records, after breaking at sequence of 10 or more n or N characters. **N50 scaffold size:** N50 size of FASTA records, excluding Ns. **Gappiness:** fraction of bases that are ambiguous. **N50 perfect stretch:** N50 length in kb of segments on finished sequence from same sample that are perfectly mirrored in assembly (see text). **Phasing error % at 1 Mb:** fraction of phased sites in megabubble branches where phasing did not agree with the majority. **Missing k-mers:** fraction of 100 k-mers in reference that are missing from the

assembly. **Haploid**: haploid version of assembly. **Diploid**: diploid version of assembly.

Discrepant at given distance: of k-mers pairs at the given distance in the assembly, and for which both are uniquely placed on the reference, fraction for which either the reference chromosome, orientation, order, or separation ($\pm 10\%$) are inconsistent. **Wall clock**: run time (days) for assemblies using a single server having 384 GB available memory (booted with “mem=384G”), starting from FASTQ files.

Example 2 – Generation of sequencing data

[059] Nucleic acid preparation and generation of sequencing data has been previously described. Briefly, several million beads were used as input to a given library construction, with each bead containing many copies of a 14-base barcode unique to that bead. A microfluidic device delivers individual beads into approximately one million partitions, along with genomic DNA and reagents. Each partition contained several long fragments (as discussed below), and the system was arranged to create constructs having the barcode, along with ~300 bp of genomic DNA from a fragment sandwiched between Illumina adapters. The barcode was placed at the beginning of the first read in a pair.

[060] Of the nucleic acid that is loaded, approximately 40% appears in the library. For example, if 1.25 ng of material is loaded, distributed across 106 partitions, and has a mean size of 50 kb, the mean number of molecules per partition would be about 10—representing ~0.5 Mb of genome per partition. At 56x coverage, the mean number of read pairs per molecule for a human genome would thus be $(1200\text{M}/2) / (106 \times 10) = 60$, and covering the molecule to depth $(120 \times 150) / (50,000) = 0.36\text{x}$.

[061] For smaller genomes, and at the same fixed level of coverage (56x), the number of read pairs per molecule drops proportionally, which can reduce the power of the data type. For example, for a genome whose size is 1/10th the size of the human genome (320 Mb), the mean number of read pairs per molecule would be about 6, and the distance between read pairs would be about 8 kb, making it hard to anchor barcodes to short initial contigs.

[062] These constructs are then sequenced on an Illumina instrument. Paired reads of length 150 bases each should be generated. This read length was chosen so that data could be sequenced on the HiSeq X instrument, which yields the lowest cost data among Illumina instruments, and which has a maximum read length of 150. Data can also be generated on the HiSeq 2500 in rapid run mode. We recommend that samples be sequenced to 56x, or about 1200M reads for a human genome, however, lower coverage is possible and described later.

Example 3 – Assemblies using nucleic acids of differing size

[063] The performance of the system was tested on DNA of several different sizes, showing that DNA length is an important factor. The data in **Table 2** show assembly performance on data

from four different libraries, constructed from NA12878 DNA of various lengths, and sequenced to 38x coverage. Particularly for DNA < 30 kb, data showed that a DNA of size ~20 kb yielded scaffolds of N50 size 0.6 Mb, whereas DNA of size ~50 kb yielded scaffolds of size N50 12.8 Mb.

Table 2					
DNA length estimates (kb)		Output statistics			
From gel measurement	Length weighted mean, from Supernova	N50 perfect stretch (kb)	N50 contig (kb)	N50 phase block (Mb)	N50 scaffold (Mb)
13	14	2.0	14.2	0.0	0.0
20	21	7.9	61.4	0.1	0.6
30	35	9.0	78.1	0.3	9.6
68	48	8.3	86.9	0.5	12.8

Example 4 – Assemblies of HGP sample at varied coverage

[064] Additionally, sequencing and assembly of the human genome project donor sample was performed at varying coverage. **Table 3** shows that large phase blocks of at least 2.3 megabases can be generated from 38x sequence coverage.

Table 3 - Assemblies of HGP sample at varied coverage				
Coverage (x)	N50 contig (kb)	N50 phase block (Mb)	N50 scaffold (Mb)	N50 perfect stretch (kb)
38	89.8	2.3	13.5	11.8
47	95.0	3.3	14.0	15.1
56	95.2	4.1	13.4	16.9

Example 5 – De novo assembly

[065] Barcoded data provides shallow coverage of each molecule, it is not possible to start the assembly process by separately assembling the reads in each partition (which otherwise would be a natural approach). Instead in this example, the assembly process proceeds by building progressively larger assembly units. Once these units are a few kb long, the probability that a given unit overlapped reads from a given molecule (at the same locus) is high, and it is thus possible to identify many of the barcodes that are incident upon the unit, hence to group the barcodes, and hence assemble groups. This is an analog of separately assembling reads from each partition.

[066] The Supernova algorithm follows this, thus deferring major use of barcodes. To start out, a *De Bruijn* graph algorithm was approached, adapting the method of DISCOVAR to scale to whole genome datasets and exploit barcoded data. K-mers (K=48 in some embodiments) were pre-filtered to remove those present in only one barcode, thus reducing the incidence of false k-mers, i.e., those absent from the sample. The remaining k-mers were formed into an initial

directed graph, in which edges represented unbranched DNA sequences, and abutting edges overlapped by K-1 bases. Operations were then carried out to recover missing k-mers and remove residual false k-mers. At this point the graph (called the base graph) is an approximation to what would be obtained by collapsing the true sample genome sequence along perfect 48-mer repeats.

[067] Next for each read pair, where possible, one path or sometimes more paths were found in the graph that could represent the sequence of the originating insert. These paths were represented as sequences of integers corresponding to the identifiers of edges in the base graph. Whenever there were two paths perfectly overlapping by K=200 bases, the paths formally joined via an equivalence relation. This yielded a new directed graph, called the super graph, whose edges were labeled by sequences of integers, representing paths in the base graph. Each super graph edge may be translated into a DNA sequence. Where super graph edges abutted, their associated sequences overlapped by K-1 bases (K=48 in this example). However the super graph represented an approximation to what would be obtained by collapsing the true sample genome sequence along perfect 200-mer repeats. It was thus far more resolved than the base graph.

[068] The remainder of the assembly process consisted of a series of operations that modifies this graph, with the intention of improving it. To facilitate these operations and to understand the nature of individual assemblies, the graph was decomposed into units called lines. Referring to **Fig. 7**, lines were extended linear regions, punctuated only by “bubbles.” Bubbles are places in the graph where the sequence diverges temporarily along alternate paths that then reconnected. Initially most arise from heterozygous sites in the genome. In **Fig. 7**, each edge represents a DNA sequence. In panel **701**, the portion **702** describes a line in an assembly graph, which is an acyclic graph part bounded on both ends by single edges. The line alternates between five common segments and four bubbles, three of which have two branches. The third bubble **703** is more complicated. The entire graph may be partitioned so that each of its edges lies in a unique line (allowing for degenerate cases, including single edge lines, and circles). The panel **711** shows the line **712** identical to the line **702**, but now each bubble has been replaced by a bubble consisting of all its paths. After this change, each bubble consists only of parallel edges.

[069] One can use lines to scaffold the super graph. This involves determining the relative order and orientation of two lines, then breaking the connections at their ends, then inserting a special “gap” edge between the lines. The end result is a new line, which has a special “bubble” consisting only of a gap edge. Subsequent operations (described later) may remove some of these gaps, replacing them by sequence.

[070] Scaffolding was first carried out using read pairs. If the right end of one line was unambiguously connected by read pairs to the left end of another line, then they could be

connected. Read pairs can reach over short gaps.

[071] To scaffold across larger gaps, barcodes were used. Briefly, if two lines were actually near each other in the genome, then with high probability, multiple molecules (in the partitions) bridged the gap between the two lines. Therefore for any line, one may find candidate lines in its neighborhood by looking for other lines sharing many of the same barcodes. Alternative orders and orientations (O&Os) of these lines were then tested, judiciously restricting tests to small sets of lines to avoid a combinatorial explosion.

[072] For all lines in the assembly, an initial computation was carried out. The computation assigned a linear coordinate system to each line and marks the positions of uniquely placed reads on it, organized by barcode. Now for a given line set S, one can score alternative O&O possibilities, as follows. Each O&O for S thus yielded a sequence of barcoded read positions along a hypothetical merged line. A score was computed for the given O&O, which was a sum, over all its constituent barcodes. For each barcode, first compute the mean separation between successive read placements for that barcode (in the merged line). Then, traversing these placements, in order, found those pairs of consecutive placements that bridged a jump from one constituent line to another, and which may thus represent a misconnection. The separation was divided for this pair by the mean separation for the barcode. If the quotient was smaller than a fixed bound, e.g., 2.0, it was discarded on the theory that it is probably noise. The remaining quotient was added to the score sum.

[073] A given O&O was treated as the “winner” if its score was at least a fixed amount less than competing tested O&O possibilities for the same set of lines. On this basis, lines were scaffolded using barcodes.

[074] Once the assembly had been scaffolded, some gaps may be filled in with sequence. For short gaps, read pairs from both sides of the gap reached in and may cover the intervening sequence, from which it may be inferred. For long gaps, first find the barcodes that were incident on sequence proximate to the left and right sides of the gap. Then, find all the reads in these barcodes. This set of reads would include reads that properly were within the gap, and yet be roughly ten times larger than that set (as each droplet contains about ten molecules). The full set of reads was assembled. Reads outside the gap locus tended to be at low coverage and hence not assemble. In this way it was typically possible to fill in the gap with a chunk of graph, and thereby remove the gap from the assembly. The chunk may not be a single sequence. For example at this stage heterozygous sites within the gap would typically be manifested as simple bubbles.

[075] The final step in the assembly process was to phase lines. Referring **Fig. 7**, first for each line, one can find all its simple bubbles, i.e., bubbles having just two branches. Then, a set of

molecules was defined. These were defined by series of reads from the same barcode, incident upon the line, and not having very large gaps (> 100 kb).

[076] A “phasing” is an orientation of each bubble, placing one branch on “top” and the other on the “bottom.” Initially an arbitrary orientation was chosen. Each molecule touched some bubbles, and thus (relative to a given phasing) may be represented as a sequence with entries +1 for top, -1 for bottom, or 0 for silent. A phasing is “good” if each molecule is coherent, containing nearly all 1s or nearly all -1s (plus 0s at silent positions). Accordingly, the score of a phasing was defined to be the sum over all molecules of $\text{Max}(\text{pluses}, \text{minuses}) - \text{Min}(\text{pluses}, \text{minuses})$.

[077] Then, this example carried out iterative perturbations, each of which flipped some bubbles, and kept only those perturbations that increased the phasing score. Three types of perturbations were attempted: (a) Flip bubbles on a given molecule to make it completely coherent; (b) Flip an individual bubble; and (c) Pivot at a given point, flipping all bubbles to its left.

[078] Now came with an initial phasing. Then, one can consider weaknesses in it. First, if flipping a bubble had too small an effect on the score, it was excluded from the phasing operation. For example a bubble might arise at a long homopolymer whose length was fixed in the sample but changed during data generation. Second, if a pivot had too small an effect on the score, the phasing was broken at the pivot point, resulting in multiple phase blocks for a given scaffold. For example this could happen if a sufficiently long block in a given sample was homozygous.

[079] In some applications, Supernova was designed to run on a single Linux server. For a human-sized genome, typical peak memory usage was 300 GB; it is recommended to use a server having ≥ 384 GB RAM. Wall clock run times are shown in **Table 1**. At fixed coverage, memory and run time were roughly linear as a function of genome size.

Example 6 – Supernova output

[080] Again referring **Fig. 3**, a Supernova assembly can capture the biology of a diploid genome. Phase blocks appear as “megabubbles” with each branch representing one parental allele, whereas sequences between megabubbles are nominally homozygous. Successive megabubbles are not phased relative to each other (if they were, they would have been combined). A chain of megabubbles as shown comprise a given scaffold. In addition to large scale features, the Supernova graph encodes smaller features such as gaps and bubbles at long homopolymers whose length is not fully determined by the data. In **Fig. 3**, Supernova assemblies encode diploid genome architecture. Each edge represents a sequence. Megabubble arms represent alternative parental alleles at a given locus, whereas sequences between megabubbles

are homozygous (or appear so to Supernova). Smaller scale features **311** appear as gaps and bubbles.

[081] Again referring to **Fig. 5**, a Supernova assembly can be translated into FASTA in several distinct ways, which might prove useful for different applications. These allow representation of the full “raw” graph **501**, or erase micro features (choosing the most likely branch at small bubbles and replacing gap edges by Ns). There is more than one way to package the result, depending on how megabubble branch points are handled in megabubble style **502**, pseudohap style **503** and pseudohap2 style **504**. Note that erasing micro features entails some loss of information, as in some cases the wrong branch of a bubble is chosen.

[082] Cycles in the graph provide an interesting test case. A cycle means a set of one or more edges that comprise a cyclic part of the graph. These are left intact in the full graph, however in the other forms are replaced by a path through the cycle that traverses each edge at least once, followed by Ns. This unfortunately signifies a gap (which could in principle represent any sequence), whereas the full graph indicates precisely which sequences could be present at the locus.

[083] **Fig. 5** shows several styles. In **501**, the raw style represents every edge in the assembly as a FASTA record (seen as red segments). These include microbubble arms and also gaps (printed as records comprising 100 Ns for gaps bridged by read pairs, or a larger number, the estimated gap size, Supp. Note 6). Unresolved cycles were replaced by a path through the cycle, followed by 10 Ns. Bubbles and gaps generally appear once per 10-20 kb. Raw graph records are roughly two orders of magnitude shorter than megabubble arms. For each edge in the raw graph, there is also an edge written to the FASTA file representing the reverse complement sequence. For the remaining output styles, we flatten each bubble by selecting the branch having highest coverage, merge gaps with adjacent sequences (leaving Ns), and drop reverse complement edges. In the second style **502**, each megabubble arm corresponds to a FASTA record, as does each intervening sequence. The third style **503** is pseudohap style which generates a single record per scaffold. For example in the cartoon for style two, the seven red edges on top (corresponding to seven FASTA records) are combined into a single FASTA record. Megabubble arms are chosen arbitrarily so many records will mix maternal and paternal alleles. The fourth style **504** is like the pseudohap option, except that for each scaffold, two “parallel” pseudohaplotypes are created and placed in separate FASTA files.

Example 7 – Inferred DNA length

[084] For each of the assemblies, one could infer the statistics of DNA molecules that made it into a partition and were thence sequenced, thus reflecting the quality of input material and degradation during the initial steps of library construction. **Table 1** shows the inferred values for

the length-weighted mean (LWM) of these molecules, as field **F**. Dog DNA was in the range 83-90 kb whereas human DNA was in the range 92-139 kb. This difference might conceivably be attributable to differences in base composition, such as at CpG islands. All dog DNA was obtained from fresh blood, as was the longest human DNA sample. The other human samples were obtained from cell lines. The shortest human sample (NA12878) may have been shortest because of repeated handling of the DNA tube to create multiple libraries, as that DNA sample was used as a control for many experiments.

Example 8 –Assessment of human assemblies

[085] This example assessed seven assemblies and six human assemblies, encompassing a wide range of laboratory approaches, from low coverage (30x) PacBio to complex combinations of multiple technologies at much higher coverage (**Table 1**). For each assembly, several statistics were computed, to the extent that those statistics could be computed. Before computing these statistics, first step removed all scaffolds shorter than 10 kb from each assembly, thereby normalizing for differences in the actual cutoffs used in defining the assemblies, which would otherwise significantly affect statistics, including coverage of the genome.

[086] To assess the continuity of the assemblies, the first step computed the N50 contig size. The mean across the seven Supernova assemblies was 117 kb, with little variation. The three assemblies based on PacBio had much larger contigs, whereas contigs from the other assemblies were two-fold or shorter than those from Supernova.

[087] All of the Supernova assemblies were diploid, with N50 phase block size ranging from 2.7 to 10.7 Mb, with variability due presumably to varied ancestry and varied DNA length. Of the six other human assemblies, only the 702x assembly was diploid, and it had an N50 phase block size of 0.5 Mb. The large molecules underlying Linked-Reads enabled the long phase blocks that were difficult to achieve with other technologies.

[088] Scaffolds in the Supernova assemblies ranged from 15 to 19 Mb (N50). While the PacBio-only assemblies had much shorter scaffolds, the four combination assemblies had longer scaffolds, ranging from 23 to 43 Mb. The gaps (fraction of Ns) in these scaffolds also varied greatly, from 0% for the PacBio assemblies, to 2% for the Supernova assemblies, to 10% for assembly I.

[089] Any assessment of assembly continuity would be tempered by an assessment of the accuracy and completeness of those same assemblies. Although one could do this by comparing to a human reference sequence (and we do so later), the ideal would be to exploit ground truth data from the same sample that was assembled. These data would consist of clones that had been independently sequenced and assembled, and which were representative of the genome. One could find only two samples, for which such truth data was available and for which high quality

DNA could be procured to create assemblies. These were the sample from a living Human Genome Project donor, for which 340 Mb of finished clones had been sequenced and assembled during the project, at great expense, and NA12878, which were previously sequenced and assembled 4 Mb of random clones. Although the HGP clones were not truly random, one reason was that they comprised so much of the genome (~10%) that they would be reasonably representative of it.

[090] For a given sample, if the exact sequence was known for each of its chromosomes, one could assess the accuracy of an assembly of that sample by enumerating maximal regions of the genome that are perfectly represented in the assembly. Most such regions would be terminated by errors or gaps in the assembly. (Note that displaying the wrong allele would count as an error.) The N50 size of such perfectly represented regions was called the “N50 perfect stretch.” For diploid genomes, if one has both a diploid assembly (thus attempting to display all chromosomes), and representative finished sequence from the exact same sample (thus providing a sample of those chromosomes), then one can approximate the N50 perfect stretch. Of the samples in **Table 1**, only assemblies F and G satisfy these requirements.

[091] It was found that the N50 perfect stretch in these Supernova assemblies was about 19 kb (**Table 1**). Moreover, examination of the alignments of the finished sequence to the assembly reveals the exact nature of the assembly defects that terminate these perfect stretches. For example **Fig. 8** (and the corresponding alignments for thousands of other clones) shows a preponderance of errors near long homopolymers, which might be attributable to library construction defects, sequencing defects, algorithmic defects, or possibly errors in the finished sequence. In more detail, **Fig. 8** shows that inside a 162 kb region (interesting because it is subsumes a region of Neanderthal origin), there are seven discrepancies between the assembly and the finished sequence for the region (plus two gaps). One of the discrepancies is a single-base mismatch. Because all the assembly reads support the assembly sequence, it seems likely that the finished sequence is wrong in this instance (and not the Supernova assembly). In fact this site was corrected in GRCh38, and thus matches our assembly. The six remaining discrepancies are indels in long homopolymers. When one examines the data at loci like these, one typically sees very low quality reads (usually with quality disintegrating on one side of the homopolymer). Thus it is quite possible that these discrepancies are due to assembly errors.

[092] This comparison also displays two captured gaps in the assembly, one of size 46 bases (as measured by the finished sequence), and captured by read pairs, and the other of size 1765 bases, and not captured by read pairs. The short gap is abutted by low complexity sequence. For the long gap, there is a separate “standalone” contig of size 1225 bases that fits in the gap, and which perfectly matches the finished sequence, and suggesting that an improved version of the

algorithm might at least place this sequence within the gap.

Example 9 – Assembly fidelity

[093] This example considers two approaches to the assessment of human genome assembly fidelity. The first approach was to measure the properties of an assembly of a given sample by comparing to reference sequences obtained from the exact same sample. For the first approach it was necessary to have a true diploid assembly. The second approach was to measure the assembly by comparing to the human reference sequence, understanding that some differences would be due to *bona fide* differences between the originating samples.

[094] Parental sequence data may also be used to assess an assembly. In particular, this could provide a direct readout on the accuracy of phase blocks in a diploid assembly. This has not been done before for human genomes, because for the two extant diploid human assemblies, the parents were not sequenced. This example had four of the Supernova assemblies, (C, E, and G in **Table 1**). The parents had been sequenced, and phased VCFs were available. This example allowed estimation of the phasing accuracy of these assemblies.

[095] To do this, for each megabubble, whenever two positions were found on alternate branches that could be mapped to the same position on GRCh37, which represented different bases (a heterozygous SNP), and which was phased in the VCF, either 0 or 1 was recorded, depending on whether the “top” branch of the megabubble was assigned to the maternal or paternal allele. A sequence of all 0s or all 1s would represent perfect phasing. An evaluation counted all the “votes” (0 or 1), and counted all “wrong votes” (1 if majority = 0, 0 if majority = 1), and summed across all megabubbles of size ≥ 100 kb. The global error rate for phasing of a given assembly would be (wrong votes)/votes, noting that a “long switch” error on even a single megabubble could drive this rate up. This example did not screen out “wrong chromosome” events so that these would also contribute to the error rate (on average 50% of the time).

[096] The observed error rates were (shown in **Table 1**): assembly C (HG00733, Puerto Rican) 0.089% (1368 errors); assembly E (NA24385, Ashkenazi) 0.053% (640 errors); assembly G (NA12878, European) 0.018% = (270 errors). Of the 270 errors for assembly G, 178 were in a single 2 Mb megabubble, and represented a “long switch” error. Similarly of the 640 errors for assembly E, 556 were in two events. For assembly G, 97% of megabubbles were free of detected phasing errors, and 96% for assembly E, whereas this number was only 66% for assembly C, suggesting inaccuracy in the phasing truth data for the Puerto Rican sample. Overall, the data suggests that phasing errors consists of very rare long switch events (perhaps 1-2 per assembly), together with isolated short switch events occurring in a few percent of megabubbles.

[097] Reference sample comparison is described below. Measurements of assembly completeness were highly dependent on the size floor that is used. This example chose an

arbitrary cutoff of 10 kb, ignoring scaffolds shorter than this size. To measure the relative completeness of different assemblies, this example chose count k-mers, because although this approach was imperfect, it was simple and thus relatively straightforward to interpret. Moreover, the method would correctly penalize for regions in an assembly that had a very high error rate. This example used $K = 100$, balancing between two considerations. First it was thought particularly important that the fraction of duplicate k-mers be small, as the analysis would be blind to them. The fraction of duplicate k-mers in GRCh37 is 2.3%. Second, this example did not want to lose too many k-mers to polymorphism. Assuming a polymorphism rate of 1/1000, one would expect about 10% of k-mers would be missing because of sample-to-sample differences.

[098] This example then defined the completeness of a human assembly to be the fraction of the non-duplicate k-mers in GRCh37 that occur in the assembly. Coverage of haploid assemblies is labeled as such (**Table 1**). For Supernova assemblies, this example could either compute their haploid coverage (using output type pseudohap) or their diploid coverage (using output type pseudohap2). This example did not use k-mers in the raw graph although that would have yielded a somewhat higher coverage. For the YH assembly, because there was no direct way to divide the assembly into haplotypes, this example used the entire assembly and reported the coverage statistic as diploid.

[099] This example then assessed misassemblies. To do this, for a given assembly, and for fixed sizes (1 Mb, 10 Mb), the study selected all scaffold segments of the given size in the assembly whose end k-mers occurred exactly once in the reference sequence. This example reported the fraction of such segments for which the end k-mer positions are consistent, meaning specifically that they lie on the same chromosome, in the correct order and orientation, and define a fragment whose length is within 10% of the fixed size. This example excluded cases that bridged a gap of size 100 or more in the reference, as these gap sizes could be inaccurate or polymorphic.

Example 10 – Computational benefits

[0100] The aforementioned examples show embodiments of the technologies disclosed herein. In contrast to the existing technologies, such as those provided by PacBio, the examples herein started with different data types resulting from the underlying barcode technology. As such, the examples considered less noisy data, resulting in lower error rate and higher accuracy.

[0101] The technologies disclosed herein include a preliminary filtering step. The filtering step comprises utilization of base quality scores from the sequencer. Further, the step considers k-mers that occur more than once. The step further comprises utilization of the barcodes, wherein each k-mer must be seen arising from two distinct barcodes. A great advantage of the filtering step was the ability to reduce the amount of starting data by at least two-fold to an order of

magnitude.

[0102] Technologies disclosed herein utilize simple data structures: vectors of vectors.

Vectorized computations allowed for faster computing time. When sparse vectors/matrices arise in some applications, vectorized computations are easier to be manipulated for much reduction in computing time.

[0103] Technologies disclosed herein utilize loops to employ CPUs, rather than GPUs, for massively parallel computing. In some applications GPUs are employed. Certain implementations include both CPUs and GPUs are employed. The merits of parallel computing allow for reduced computing time.

[0104] Technologies disclosed herein utilize lossless random access compression applied to each record of the quality scores and the paths through the graphs (including sequences and edges). The merits of compression allow for less required memory for the analysis and less required storage for holding data or analysis outcomes.

[0105] In contrast to the FALCON assembler from PacBio, these examples reduced computing time by 180-fold and lowered the memory utilization by 21-fold.

Example 11 – Computing Architecture

[0106] **Fig. 9** shows a non-limiting block diagram of a *de novo* assembly system **901** as described herein. The system may comprise at least one CPU **902**, memory **903**, and storage **904**. The computing architecture functions as follows. A set of sequence data **911** is supplied to the *de novo* assembly system **901**. A preliminary filter **921** utilizes base quality scores from a sequencer used to generate the short-read sequence data, and utilizes k-mers for preprocessing. Then, an assembler **922** creates an initial assembly graph. This initial assembly amounts to a “rough sketch” assembly and temporarily ignores areas of unresolved complexity, e.g., regions that may be ambiguous upon first glance, thus, preserving computing power.

[0107] The output of the initial assembler **922** is fed to an ambiguity assembler **923**. Areas of ambiguity may then be further processed by employing the barcoded sequencing data **912** to create an accurate assembly of the areas of ambiguity. The output of the ambiguity processor **923** includes k-mers derived from sequencing reads created during the assembly process to be addressable to a certain chromosome, cell, population, haplotype, etc. An assembler **924** is employed to assemble true diploid, based on steps shown in **Fig. 2**. The barcoded reads are placed back on the assembly and loci are identified whose exact sequence is not known with certainty, and so marked. A final assembler **925** is used to align the global assembly graph with respect to a reference sequence. Edges are aligned individually. When there are inconsistent alignments, the inconsistency is resolved by aligning the concatenations of these edges.

[0108] The system may further comprise a lossless random access compressor **926** that compresses one or more records of the quality scores and paths through the graph.

[0109] In various applications, the filter (**921**), assemblers (**922**, **923**, **924** and **925**), and compressor (**926**) may be implemented in hardware, or software, or a combination thereof. The filter (**921**), assemblers (**922**, **923**, **924** and **925**), and compressor (**926**) can, individually or together, be configured to execute one or more functionalities disclosed herein. Some implementations may interchange the execution order of the filter (**921**), assemblers (**922**, **923**, **924** and **925**), and compressor (**926**), or may integrate two or more them into a single execution module.

[0110] While preferred embodiments of the present invention have been shown and described herein, it will be obvious to those skilled in the art that such embodiments are provided by way of example only. Numerous variations, changes, and substitutions will now occur to those skilled in the art without departing from the invention. It should be understood that various alternatives to the embodiments of the invention described herein may be employed in practicing the invention.

CLAIMS**WHAT IS CLAIMED IS:**

1. A computer-implemented method of *de novo* genome assembly for nucleic acid sequence data generated from a nucleic acid sample of an organism, the method comprising:
 - a) generating, by one or more computers, an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read sequence data derived from a common starting sequence share a common tag;
 - b) generating, by the one or more computers, a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity;
 - c) generating, by the one or more computers, a global assembly based on the plurality of local assemblies;
 - d) cleaning, by the one or more computers, the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and
 - e) generating, by the one or more computers, a phased genome assembly based on the global assembly by utilizing the tags to separate phased nucleotide sequence; wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence.
2. The method of claim 1, wherein the genome is diploid.
3. The method of claim 1, wherein the short-read sequence data is generated from a single library.
4. The method of claim 1, wherein the short-read sequence data results in 50x or less coverage of the genome of the organism.
5. The method of claim 1, wherein the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads.
6. The method of claim 1, wherein the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb.
7. The method of claim 1, wherein the initial assembly is an initial assembly graph.
8. The method of claim 7, wherein the initial assembly graph is generated by:

- a) identifying a plurality of k-mers that have a high probability of being present in the genome of the organism;
 - b) using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and
 - c) bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$.
9. The method of claim 8, further comprising applying, by the one or more computers, a preliminary filter prior to generating the initial assembly, wherein the preliminary filter comprises:
- a) utilization of base quality scores from a sequencer used to generate the short-read sequence data, and
 - b) utilization of k-mers that occur more than once and the tags, such that each k-mer must be seen arising from two distinct tags.
10. The method of claim 9, further comprising applying, by the one or more computers, lossless random access compression to each record of the quality scores and paths through the graph.
11. The method of claim 8, wherein method further comprises revising, by the one or more computers, the initial assembly graph by:
- a) eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and
 - b) filling in gaps in the initial assembly graph by consulting the original short-read sequence data.
12. The method of claim 8, wherein k is between 24 and 96.
13. The method of claim 8, wherein the plurality of local assemblies is generated by:
- a) using the initial assembly graph as in interim reference;
 - b) identifying edges of unambiguous sequence;
 - c) identifying neighboring edges sharing a number of tags above a threshold number of tags; and
 - d) bringing together edges of unambiguous sequence with the identified neighboring edges.
14. The method of claim 13, wherein the global assembly is generated by:
- a) identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and
 - b) bringing together the z-mers in the plurality of local assemblies.

15. The method of claim 14, wherein z is between 100 and 300.
16. The method of claim 1, wherein the short-read sequence data is generated from less than 10 ng of DNA input material.
17. The method of claim 16, wherein the short-read sequence data is generated from less than 2 ng of DNA input material.
18. The method of claim 1, wherein the assembly is completed in less than 60 minutes.
19. The method of claim 18, wherein the one or more computers comprises less than 512 GB of storage.
20. The method of claim 19, wherein the one or more computers comprises less than 60 GB of storage.
21. The method of claim 1, wherein the assembly is completed in less than 20 minutes.
22. The method of claim 21, wherein the one or more computers comprises less than 512 GB of storage.
23. The method of claim 22, wherein the one or more computers comprises less than 60 GB of storage.
24. The method of claim 1, wherein the organism is a human being.
25. The method of claim 1, wherein the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly.
26. The method of claim 1, wherein the one or more computers occupies one cubic foot or less of physical space.
27. A computer-implemented system comprising: a digital processing device comprising: at least one processor, an operating system configured to perform executable instructions, a memory, and a computer program including instructions executable by the digital processing device to create a *de novo* genome assembly application for nucleic acid sequence data generated from a nucleic acid sample of an organism, the application comprising:
 - a) a first software module generating an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read sequence data derived from a common starting sequence share a common tag;
 - b) a second software module generating a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity;

- c) a third software module generating a global assembly based on the plurality of local assemblies;
- d) a fourth software module cleaning the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and
- e) a fifth software module generating a phased genome assembly based on the global assembly by utilizing the tags to separate homologous phased nucleotide sequence;

wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence.

- 28. The system of claim 27, wherein the genome is diploid.
- 29. The system of claim 27, wherein the short-read sequence data is generated from a single library.
- 30. The system of claim 27, wherein the short-read sequence data results in 50x or less coverage of the genome of the organism.
- 31. The system of claim 27, wherein the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads.
- 32. The system of claim 27, wherein the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb.
- 33. The system of claim 27, wherein the initial assembly is an initial assembly graph.
- 34. The system of claim 33, wherein the software module generating an initial assembly graph generates the initial assembly graph by:
 - a) identifying a plurality of k-mers that have a high probability of being present in the genome of the organism;
 - b) using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and
 - c) bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$.
- 35. The system of claim 33, wherein the application further comprises a software module applying a preliminary filter prior to generating the initial assembly, wherein the preliminary filter comprises:
 - a) utilization of base quality scores from a sequencer used to generate the short-read sequence data, and
 - b) utilization of k-mers that occur more than once and the tags, such that each k-mer must be seen arising from two distinct tags.

36. The system of claim 35, wherein the application further comprises a software module applying lossless random access compression to each record of the quality scores and paths through the graph.
37. The system of claim 34, wherein the software module generating an initial assembly graph revises the initial assembly graph by:
- a) eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and
 - b) filling in gaps in the initial assembly graph by consulting the original short-read sequence data.
38. The system of claim 34, wherein k is between 24 and 96.
39. The system of claim 34, wherein the software module generating a plurality of local assemblies generates the plurality of local assemblies by:
- a) using the initial assembly graph as in interim reference;
 - b) identifying edges of unambiguous sequence;
 - c) identifying neighboring edges sharing a number of tags above a threshold number of tags; and
 - d) bringing together edges of unambiguous sequence with the identified neighboring edges.
40. The system of claim 39, wherein the software module generating a global assembly generates the global assembly by:
- a) identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and
 - b) bringing together the z-mers in the plurality of local assemblies.
41. The system of claim 40, wherein z is between 100 and 300.
42. The system of claim 27, wherein the short-read sequence data is generated from less than 10 ng of DNA input material.
43. The system of claim 42, wherein the short-read sequence data is generated from less than 2 ng of DNA input material.
44. The system of claim 27, wherein the assembly is completed in less than 60 minutes.
45. The system of claim 44, wherein the memory comprises less than 512 GB of storage.
46. The system of claim 45, wherein the memory comprises less than 60 GB of storage.
47. The system of claim 27, wherein the assembly is completed in less than 20 minutes.
48. The system of claim 47, wherein the memory comprises less than 512 GB of storage.
49. The system of claim 48, wherein the memory comprises less than 60 GB of storage.

50. The system of claim 27, wherein the organism is a human being.
51. The system of claim 27, wherein the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly.
52. The system of claim 27, wherein the digital processing device occupies one cubic foot or less of physical space.
53. Non-transitory computer-readable storage media encoded with a computer program including instructions executable by a processing device to create a *de novo* genome assembly application for nucleic acid sequence data generated from a nucleic acid sample of an organism, the application comprising:
- a) a first software module generating an initial assembly based on short-read sequence data, the initial assembly comprising one or more areas of unresolved sequence ambiguity, wherein the short-read sequence data is derived from longer starting sequences from the nucleic acid sequence data and is tagged to preserve long-range sequence context for the organism such that a subset of the short-read sequence data derived from a common starting sequence share a common tag;
 - b) a second software module generating a plurality of local assemblies based on the initial assembly by utilizing the tags to resolve a plurality of areas of sequence ambiguity;
 - c) a third software module generating a global assembly based on the plurality of local assemblies;
 - d) a fourth software module cleaning the global assembly by removing sequence data inconsistent with the long-range sequence context indicated by the tags; and
 - e) a fifth software module generating a phased genome assembly based on the global assembly by utilizing the tags to separate homologous phased nucleotide sequence;
- wherein the phased genome assembly is achieved without alignment to a reference sequence or any independently generated genome sequence.
54. The media of claim 53, wherein the genome is diploid.
55. The media of claim 53, wherein the short-read sequence data is generated from a single library.
56. The media of claim 53, wherein the short-read sequence data results in 50x or less coverage of the genome of the organism.
57. The media of claim 53, wherein the short-read sequence data is tagged to preserve context over a starting sequence 2x – 1000x longer than the reads.

58. The media of claim 53, wherein the short-read sequence data is tagged to preserve context over a starting sequence of 10 kb – 5 Mb.
59. The media of claim 53, wherein the initial assembly is an initial assembly graph.
60. The media of claim 59, wherein the software module generating an initial assembly graph generates the initial assembly graph by:
 - a) identifying a plurality of k-mers that have a high probability of being present in the genome of the organism;
 - b) using the tags to filter the plurality of k-mers based on the number of starting sequences each k-mer occurs in; and
 - c) bringing together k-mers in the plurality of k-mers sharing a common l-mer to form an initial assembly, wherein $l < k$.
61. The media of claim 60, wherein the application further comprises a software module applying a preliminary filter prior to generating the initial assembly, wherein the preliminary filter comprises:
 - a) utilization of base quality scores from a sequencer used to generate the short-read sequence data, and
 - b) utilization of k-mers that occur more than once and the tags, such that each k-mer must be seen arising from two distinct tags.
62. The media of claim 61, wherein the application further comprises a software module applying lossless random access compression to each record of the quality scores and paths through the graph.
63. The media of claim 60, wherein the software module generating an initial assembly graph revises the initial assembly graph by:
 - a) eliminating one or more areas of sequence ambiguity based on a number of reads available for each option within an area of sequence ambiguity; and
 - b) filling in gaps in the initial assembly graph by consulting the original short-read sequence data.
64. The media of claim 60, wherein k is between 24 and 96.
65. The media of claim 60, wherein the software module generating a plurality of local assemblies generates the plurality of local assemblies by:
 - a) using the initial assembly graph as in interim reference;
 - b) identifying edges of unambiguous sequence;
 - c) identifying neighboring edges sharing a number of tags above a threshold number of tags; and

- d) bringing together edges of unambiguous sequence with the identified neighboring edges.
66. The media of claim 65, wherein the software module generating a global assembly generates the global assembly by:
- a) identifying a plurality of z-mers in the plurality of local assemblies that have a high probability of being present in the genome of the organism, wherein $z > k$; and
 - b) bringing together the z-mers in the plurality of local assemblies.
67. The media of claim 66, wherein z is between 100 and 300.
68. The media of claim 53, wherein the short-read sequence data is generated from less than 10 ng of DNA input material.
69. The media of claim 68, wherein the short-read sequence data is generated from less than 2 ng of DNA input material.
70. The media of claim 53, wherein the assembly is completed in less than 60 minutes.
71. The media of claim 70, wherein the processing device comprises less than 512 GB of storage.
72. The media of claim 71, wherein the processing device comprises less than 60 GB of storage.
73. The media of claim 53, wherein the assembly is completed in less than 20 minutes.
74. The media of claim 73, wherein the processing device comprises less than 512 GB of storage.
75. The media of claim 74, wherein the processing device comprises less than 60 GB of storage.
76. The media of claim 53, wherein the organism is a human being.
77. The media of claim 53, wherein the DNA sequence data is whole genome sequence data and the phased genome assembly is a whole genome assembly.
78. The media of claim 53, wherein the processing device occupies one cubic foot or less of physical space.