(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 September 2014 (18.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/145503 A2

(51) International Patent Classification:
A01H 5/00 (2006.01) *C12N 15/82* (2006.01)

(21) International Application Number:
PCT/US2014/030288

(22) International Filing Date:
17 March 2014 (17.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/791,948 15 March 2013 (15.03.2013) US

(71) Applicant: **LIEBER INSTITUTE FOR BRAIN DEVELOPMENT** [US/US]; 855 North Wolfe Street, Suite 300, 3rd Floor, Baltimore, Maryland 21205 (US).

(72) Inventors: **GAO, Yuan**; 9884 Foxhill Ct, Ellicott City, Maryland 21042 (US). **LIM, Hun Ki**; 4606 Springwater CT. #A, Owings Mills, Maryland 21117 (US). **SHIN, Joo Heon**; 3 Athenry Ct. #101, Timonium, Maryland 21093 (US).

(74) Agent: **BRIVANLOU, Margaret**; King & Spalding LLP, 1185 Avenue of the Americas, New York, New York 10036 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

WO 2014/145503 A2

(54) **Title**: SEQUENCE ALIGNMENT USING DIVIDE AND CONQUER MAXIMUM OLIGONUCLEOTIDE MAPPING (DCMOM), APPARATUS, SYSTEM AND METHOD RELATED THERETO

(57) **Abstract**: A novel apparatus, system and method for sequence alignment of arbitrary string data, including nucleotide sequences, is provided. A root node is selected from a query sequence and located in the reference sequence. The root node is extended to encompass adjacent elements which are present in both the query and reference sequences, to form an extended root node match. The area to the left and/or right side of the extended root node match is then recursively searched using the same method to identify additional matches. When searching is complete, the joints between the identified nodes are classified according to their characteristics as, e.g., SNPs, deletions, substitutions, insertions, indels, and/or introns. The apparatus may be included in a DNA sequencing machine, or it may be a stand alone machine. Non-biological applications are also provided.

Sequence Alignment Using Divide and Conquer Maximum Oligonucleotide Mapping (DCMOM), Apparatus, System and Method Related Thereto

FIELD OF INVENTION

[0001] The present invention relates to an apparatus, system, and method for the alignment of arbitrary sequences, including nucleotide sequences, to a reference sequence, for example a genome, using a novel divide and conquer technique as well as systems and methods of using the same.

BACKGROUND OF THE INVENTION

[0002] Recent advances in next-generation sequencing ("NGS") technologies provide unprecedented opportunities for the study of various genomes and transcriptomes. NGS, relative to previous technologies, enables a deeper understanding of complex genetic variants, as well as gene expression and splicing events. However, NGS poses many challenges with respect to data analysis. The increased amount of output data (i.e., reads) resulting from NGS has created a need for an improved sequence analysis, and mapping apparatus, system, and methods of using the same, which are faster, more reliable, deterministic, and/or less dependent on heuristic assumptions or configuration parameters compared with those systems already available.

[0003] The extremely large number of reads, or outputs, from NGS makes the alignment of reads a daunting computational issue. It is common to obtain billions of single- or paired-end reads from a single run. Read mapping, as the first step in genome, DNA, transcriptome, or RNA sequence analysis (e.g., "RNA-Seq") has become a central issue and roadblock in this field. The short length of reads, e.g., sequence fragments, poses great challenges to known read alignment or mapping methods; a short read with possible genetic variants including mutations and indels often causes an alignment to be non-uniquely determined, or in some cases, unmatched. Further, ongoing improvements to and new applications of NGS generate variable types of data with differing characteristics that should be considered in the process of alignment. For example, reads can be single-end or paired-end, and reads can be of uniform length or variable length.

[0004] As NGS technology progresses, individual reads may in some cases grow longer, causing error rates to increase. These issues are compounded in RNA-Seq data by the potential for splicing events, which complicate the analysis. Additional challenges posed by NGS reads

include low quality reads, spliced reads, and reads spanning several discontinuous regions or exons on a reference genome. The read mapper is a critical part of the scientific, medical and health care services industries, and one capable of dealing with the challenges introduced by RNA-Seq generated data is desirable. For these reasons, as well as others described herein, an improved read mapper is needed that is able to handle these and other characteristics of NGS reads, for example, other biological applications and information, and also for improving searching and matching methods and apparatus in non-biological fields.

[0005] Maximum Oligonucleotide Mapping ("MOM"; Eaves and Gao, *Bioinformatics* Vol. 25, no. 7 2009, pages 969-970, the entirety of which is hereby incorporated by reference) is a read mapping method with improved sensitivity compared to previous mappers such as MAQ and SOAP, because it is better able to deal with errors at both ends (i.e., both the 3' and 5' ends) of a read by trimming the read in the process of matching; however, MOM is not deterministic and cannot process more complex problems in a read, such as with the introns and exons present in transcriptome analysis.

[0006] Other spliced read mappers make various improvements upon previous-generation mappers, such as BLAST (Mount, David W., Cold Spring Harbor Protocols 2007) and BLAT (Kent, W. James, Genome Res. 2002, 12:656-664). These programs include RUM (Grant, Gregory et al., Bioinformatics (2011) Vol. 27, no. 18, pages 2518-2528), GSNAP (Wu, Thomas D., Bioinformatics (2010) 26 (7): 873-881), TopHat (Trapnell, Cole et al., Bioinformatics (2009) 25 (9) 1105-1111), Mapsplice (Wang, Kai et al., Nucl Acids Res. (2010) 38(18), e178), SpliceMap (Au, Kin Fai et al., Nucleic Acids Res. (2010) 1-9), Bowtie (Langmead, Ben et al., Genome Biology (2009) 10:R25), BWA (Li H. et al. (2009), Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60) (Li H. et al. (2010) Fast and accurate long-read alignment with Burrows-Wheeler Transform. Bioinformatics, Epub), MAQ (Li, Heng et al., Genome Research (2008) Nov; 18(11):1851-8), Mosaik (Stromberg, available at http://bioinformatics.bc.edu/marthlab/Mosaik), Novoalign (available at http://www.novocraft.com), SOAP (Li, Ruiqiang et al., Bioinformatics (2008) 24 (5):713-714), SOAP2 (Li, Ruiqiang et al., Bioinformatics (2009) 25 (15):1966-1967), SOAP3 (Li, Ruiqiang et al., Bioinformatics (2012) 28(6): 878-879), QPALMA (De Bona, Fabio et al., Bioinformatics (2008) Vol. 24 (16):i174-i180), or ZOOM! (Lin H. et al., Bioinformatics (2008) 24(21):2431-7). However, there are many ways in which each of these systems fall short in data mapping

capabilities. These shortcomings have been discussed in several other publications, and include problems such as: slow speed, high dependence on heuristics for analysis, non-parallelizable operations, and false positive or negative results.(See, e.g., Trapnell, Cole et al. *How to map billions of short reads onto genomes*, Nature Biotechnology (2009) Vol. 27 No. 5:455-457, and also *TopHat: discovering splice junctions with RNA-Seq*, Trapnell, Cole et al., Bioinformatics (2009) Vol. 25 no. 9:1105-1111.) All of the aforementioned publications, as well as any other references cited in this disclosure, are hereby incorporated by reference into this application in their entirety.

[0007] In addition to the aforementioned shortcomings, all of the existing methods are effective only in limited applications and under constrained circumstances. For example, most of them do not provide reliable performance on de novo alignment without resorting to prior annotation of gene structures or transcriptomes. Moreover, they have poor or non-existent functionality for reporting plausible mini exons (e.g., exons shorter than 25 nucleotide elements) that are not annotated.

[0008] Because of the aforementioned and other limitations, existing systems for various reasons may miss matches that could be identified, or they identify false matches. Some of the aforementioned systems, for example, require a read to satisfy certain heuristics determined by the user or inherent to the software. These may include, for example, requiring a certain number of consecutive matching elements in the read, only finding a match if it contains no more than a single splice or indel per read, or failing to detect a match if its polymorphisms (e.g., indels, splices, substitutions, and so on) do not exceed an arbitrary point value. Because of these and other limitations, existing systems are unable to, for example, reliably map reads without an annotated reference sequence, or map reads spanning multiple junctions, and are limited in their ability to discover small or mini exons, for example, with a length shorter than the seed length. Furthermore, existing systems may not beneficially use all available information from paired-end reads.

[0009] Thus, there is a need for a sequence alignment system that can process unlimited amounts of data, one that is faster, more accurate, and less dependent upon heuristic assumptions in its analyses. Generally, given a very long string (e.g., a genome sequence) and a very large set of reads (e.g., the output from an NGS process or apparatus), it is desirable to be able to map the

reads onto the long string with an unknown number or frequency of read errors, mismatches, short or long indels (i.e., insertions or deletions), and splicing junctions. For various reasons described throughout this application and throughout the referenced publications, there is a need for a deterministic local alignment method, which is unlimited by the number of errors, indels, junctions, or the size thereof, and without fixed or minimum seed lengths, and that can map reads spanning multiple junctions deterministically. Furthermore there is a need for a system that can operate without annotation of the reference sequence, and with fewer or no heuristic assumptions or configuration parameters that affect the quality of the output. Additionally, it is desirable to have a new system that can provide such mappings regardless of the data type (e.g., nucleic acid sequences, DNA, RNA, or any other arbitrary sequence), data length, structural complexity, and within a reasonable amount of time.

[0010] Thus, due to the foregoing limitations with existing systems, there exists a clear need for an improved sequence alignment apparatus, as well as systems and methods of using the same.

SUMMARY OF INVENTION

[0011] Disclosed herein are sequence alignment apparatus, methods, and systems, which remedy the limitations of existing sequence alignment apparatus, methods and systems. Divide and Conquer Maximum Oligonucleotide Mapping ("DCMOM") is a computer-based system, method and apparatus that effectively maps arbitrary sample sequences onto a reference sequence. In certain embodiments, DCMOM may map string data onto a reference string, file, filesystem, database, or any other data. Alternatively, DCMOM may be used to search any quantity of reference data for a query string. In certain embodiments, end-users may supply a DCMOM apparatus or system with a plurality of reads obtained from a sequencer, and DCMOM will output and/or display a mapping between those reads and a reference sequence, such as a genome. However, it will be apparent to one of ordinary skill in the art that the invention disclosed herein is not limited to nucleic acid sequence analysis and is also applicable to other applications, including non-biological applications.

[0012] In certain other embodiments, DCMOM may map RNA-Seq reads onto a reference genome, such as the human genome. In other embodiments, end-users may supply a DCMOM apparatus or system with a biological sample, and the DCMOM apparatus or system will

sequence the sample to create reads, subsequently aligning the reads to a reference sequence, such as a reference genome, and outputting or displaying a result to a user. End-users may program the apparatus or system via a variety of interfaces and methods to adjust the format, type, and manner of input and output data. Output may be delivered to the end-users in a variety of formats and via a variety of delivery mechanisms.

[0013] In one aspect of the invention, an apparatus is provided comprising a computer and software. The apparatus is configured to select a root node subsequence from a query string and determine the location of the root node in a reference sequence. Then, the apparatus identifies the maximally extended subsequence match for that root node, which includes elements adjacent to the root node in the query sequence which match elements in their corresponding locations in the reference sequence. Typically, the root node subsequence will be bounded by a plurality of unmatched elements adjacent to the extended root node, on the right and left sides. The apparatus will subsequently recursively search those portions of the query sequence to the right and/or left side of the extended root node to identify additional matches. Information about the matches is stored into or in a memory, including but not limited to a RAM, a hard drive, magnetic or optical media, or non-volatile memory, or any other device suitable for storing data.

[0014] In another aspect of the invention, a method is provided for sequence alignment using a computer. A user may provide the computer with a query sequence and a reference sequence. A root node subsequence is selected from a query string and its location is determined in the reference sequence. Then, the maximally extended subsequence match for that root node is identified, which typically includes elements adjacent to the root node in the query sequence that match elements in the corresponding locations in the reference sequence. Typically, the root node subsequence will be bounded by a plurality of unmatched elements adjacent to the extended root node, on the right and left sides. The next step, for the right and/or left hand sides, is to recursively search portions of the query sequence to identify additional matches. Information about the matches is stored into a memory.

[0015] In another aspect of the invention, a method for sequence alignment is provided. One step of the method involves the use of a computer adapted to execute a recursive divide and conquer method to map a query sequence to a reference sequence. Another step is to determine match information based on the mapping of the query sequence to the reference sequence.

Typically, the match information may include the location where a plurality of elements in the query sequence are present in the reference sequence.

[0016] Specifically, the invention provides an electronic apparatus comprising a CPU in communication with a memory and a plurality of machine readable storage media, in communication with the CPU, comprising software modules for determining match information corresponding to a first query sequence and a first reference sequence, in which the determining comprises (a) selecting a first root node subsequence, having a first seed length greater than or equal to 9 elements, from the first query sequence, (b) determining a location of the first root node subsequence in the first reference sequence, (c) identifying a maximally extended subsequence match including the first root node subsequence and any sequence elements adjacent to the first root node subsequence that are present in both the first query sequence and the first reference sequence, (d) identifying an unmatched sequence element not present in both the first query sequence and the first reference sequence, adjacent to the maximally extended subsequence match, and (e) repeating steps a-d with a second query sequence and a second reference sequence, wherein the second query sequence comprises at least a portion of the first query sequence other than the maximally extended subsequence match, and wherein the second reference sequence comprises at least a portion of the first reference sequence other than the maximally extended subsequence match, and (f) storing the match information into at least one of the plurality of machine readable storage media.

[0017] The invention provides methods for determining match information corresponding to a first query sequence and a first reference sequence, which methods comprise the steps of (a) selecting, by a CPU, a first root node subsequence, having a first seed length, from the first query sequence; (b) determining, by the CPU, a location of the first root node subsequence in the first reference sequence; (c) identifying, by the CPU, a maximally extended subsequence match including the first root node subsequence and any sequence elements adjacent to the first root node subsequence that are present in both the first query sequence and the first reference sequence; (d) identifying, by the CPU, an unmatched sequence element not present in both the first query sequence and the first reference sequence, adjacent to the maximally extended subsequence match; (e) repeating steps a-d with a second query sequence and a second reference sequence, wherein the second query sequence comprises at least a portion of the first query sequence other than the maximally extended subsequence match, and wherein the second

reference sequence comprises at least a portion of the first reference sequence other than the maximally extended subsequence match, and (f) storing, by the CPU, the match information into a machine readable storage medium.

[0018] In specific embodiments, the first query sequence and/or the reference sequence are nucleotide sequences. In a specific embodiment, the first query sequence is an RNA sequence, particularly an mRNA sequence, and the reference sequence is a DNA sequence. In a preferred embodiment, the first query sequence is a read output of a DNA sequencer. The first query sequence may be a paired-end read or a single end read. In addition, in specific embodiments, the first query sequence is one of a plurality of uniform length reads or one of a plurality of variable length reads.

[0019] In certain embodiments, the second query sequence comprises at least a portion of the first query sequence to the right of the maximally extended subsequence match. In certain embodiments, the second reference sequence comprises at least a portion of the first reference sequence to the right of the maximally extended subsequence match.

[0020] In certain embodiments, the number of elements in the second reference sequence is less than 30,000 elements, less than 25,000 elements, less than 20,000 elements, less than 10,000 elements.

[0021] The methods of the invention may be performed iteratively and/or in parallel to process a plurality of query sequences, e.g., reads from sequencing of nucleic acid, either DNA or RNA, such as RNA-Seq information, from a biological sample. The methods of the invention permit parallel processing of reads such that large numbers of reads may be processed, analyzed, assembled and aligned. The number of query sequences may be anywhere from 1 to billions of query sequences, including more than 10, 100, 1000, 10,000, 1,000,000, 10,000,000, 100,000,000, 1,000,000,000, or 10,000,000,000 or more. In certain embodiments, no more than 1,000,000, 10,000,000, 100,000,000, 1,000,000,000, 10,000,000,000, 100,000,000,000 or 1,000,000,000,000. These query sequences are mapped and assembled to the reference sequence to map, align and assemble the nucleotide sequences within the plurality of query sequences, e.g., the mRNA nucleotide sequences within a transcriptome or genomic sequences onto a reference genome.

[0022] In certain embodiments, the invention provides an apparatus wherein step a further comprises calculating a hash value using a first seed length portion of the first root node subsequence, and wherein step b further comprises generating a hash value for each of a plurality of first seed length portions of the reference sequence. In specific embodiments, the seed length used to generate the hash value for a portion of the second query sequence is shorter than the first seed length used to generate the hash value for a portion of the first query sequence and the seed length may be at least 9 elements, or the seed length is between 9 and 25 elements.

[0023] In specific embodiments, the determining comprises determining intron information by identifying at least 30 elements between two adjacent root nodes mapped to the first reference sequence and storing the intron information onto a machine readable storage medium. In other embodiments, the determining further comprises determining exon information by identifying at least 2 adjacent elements mapped to the first reference sequence as a node and storing the exon information onto a machine readable storage medium. In other embodiments, the determining further comprises determining single nucleotide polymorphism information by identifying one element in the query sequence which are not identical to elements in the reference sequence and storing the single nucleotide polymorphism information onto a machine readable storage medium.

[0024] In specific embodiments, the apparatus or methods of the invention comprise a DNA sequencing device in communication with the CPU, for sequencing genetic information from a sample to obtain the first query sequence. In certain embodiments, the first reference sequence represents at least a portion of the human genome, in specific embodiments the first reference sequence is a chromosome of the human genome. In preferred embodiments, the first reference sequence is a whole genome, preferably a human genome. The first reference sequence may or may not be annotated. In a specific embodiment, the first reference sequence is not annotated.

[0025] In specific embodiments, the determining further comprises additional recursive mapping steps until all elements of the query sequence are mapped to the first reference sequence or determined not to match the first reference sequence. The determining may further comprise merging the first and second root node subsequences by classifying a joint between them, and storing classification information into a machine readable storage medium. In additional embodiments, the determining may further comprise merging a plurality of nodes by

starting with the two leftmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at, then continuing with the two rightmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at.

[0026] In yet further embodiments, in merging the nodes the distance between the location of the first and second root node subsequences on the reference sequence is equal to a number of unmatched elements in the query sequence, and the classification information includes information indicating a substitution. In other embodiments, the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and there is an unmatched element between the nodes, and wherein the classification information includes information indicating an insertion. In yet other embodiments, the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and the two nodes overlap, and wherein the classification information includes information indicating an insertion. In other embodiments, the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion or intron. In yet other embodiments, the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero but less than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion. In yet other embodiments, the distance between the location of the first and second root node subsequences on the reference sequence is greater than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a intron.

[0027] In additional embodiments, the determining further comprises recursively repeating steps a-d with additional query sequences until the remaining unmatched portion of the first query sequence is smaller than 3 elements.

[0028] The invention further provides methods for sequence alignment comprising the step of using a CPU adapted to execute a recursive divide and conquer method to determine match information corresponding to a query sequence and a reference sequence, said match information including the location where a plurality of elements of the query sequence are present in the

reference sequence. Output may be provided in any file format. In specific embodiments, the output is configured as an SAM file and/or a BAM file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0029] The patent or application file contains at least one drawing executed in color. Copies of this patent or patent application publication with color drawing(s) will be provided by the Office upon request and payment of the necessary fee.

[0030] The foregoing and other features and advantages of the invention will be apparent from the following description of embodiments, including those illustrated in the accompanying drawings in which reference designations refer to the same parts throughout the various views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention. It will be appreciated by one of ordinary skill in the art that these exemplary figures do not limit the invention, but rather depict invention in its various embodiments and aspects.

[0031] **FIGS. 1A-E** illustrate the initial mapping tree ("IMT") aspect of certain embodiments of DCMOM, the node merging process, and the final junction tree ("FJT"). **FIG. 1A** depicts two reads. **FIG. 1B** depicts an IMT. **FIG. 1C** illustrates a node merging aspect DCMOM. **FIG. 1D** depicts a final function tree ("FJT"). **FIG. 1E** depicts some possible classifications of a joint between nodes.

[0032] **FIGS. 2A-C** depict a simple example of DCMOM matching a read with multiple exons to the reference sequence. **FIG. 2A** depicts a query sequence ("Read I") and a reference sequence ("Genome"), with three exons. **FIG. 2B** depicts the identification of a root node, and right and left unmapped segments. **FIG. 2C** depicts an additional node and an additional unmapped segment.

[0033] **FIG. 3** depicts the example of **FIG. 2** farther along in the recursive mapping process.

[0034] **FIG. 4** depicts DCMOM classifying a joint in a matched portion of a query sequence as a substitution.

[0035] **FIGS. 5A and B** depict DCMOM classifying two types of insertions. **FIG. 5A** depicts an insertion where the matched sequences of the two nodes are consecutively aligned to the reference genome, but there is one or more mismatched nucleotides between the nodes. **FIG.**

5B depicts an insertion where there are no mismatched nucleotides between the nodes, but the nodes overlap.

[0036] **FIG. 6** depicts DCMOM classifying a joint between two nodes of a matched query sequence as a deletion and spliced junction.

[0037] **FIG. 7** depicts a final junction tree output by DCMOM, created by merging the nodes of an initial mapping tree and classifying the junctions between them.

[0038] **FIG. 8** depicts an example result obtained from DCMOM, indicating 4 reads, each of which span several exons.

[0039] **FIG. 9** depicts DCMOM's detection of a particular short exon (in a gene identified as ABI1) in comparison with the performance of other alignment systems.

[0040] **FIG. 10** depicts DCMOM's detection of insertions corresponding to the location of known variations.

[0041] **FIG. 11** depicts DCMOM's detection of SNPs, deletions, and insertions.

[0042] **FIG. 12** compares the breadth of coverage of several alignment systems, including DCMOM, for varying levels of depth of coverage.

[0043] **FIGS. 13A-E** provide data from comparison of different sequence mapping methodologies. **FIG. 13A** compares the breadth of coverage of several alignment systems, including DCMOM, with each other, for varying levels of depth of coverage. **FIGS. 13B-D** depict magnified subsets of the data graphed in **FIG. 13A**. **FIG. 13E** depicts the breadth of coverage for several alignment systems at zero depth of coverage.

[0044] **FIG. 14** provides a flow chart illustrating, in summary form, the main portions of the operation of an exemplary DCMOM apparatus, system, or method according to the invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0045] DCMOM is a sequence mapping apparatus, system and method that, in certain embodiments may be used to align or map a query sequence onto a reference sequence, such as a genome, where the mapping may span multiple junctions and may contain an unbounded number of mismatches. In some embodiments, DCMOM may be used for the alignment of biological

sequences such as nucleotide sequences. However, DCMOM is not so limited and can be used in non-biological applications as well.

[0046] Sequence alignment (also referred to as sequence assembly or mapping) is a challenging problem because frequently there are millions or billions of elements in the reference sequence, and although the query sequence is typically much shorter, there are often millions or billions of individual query sequences to be mapped. The time savings from slightly optimizing the mapping of each individual read, even if very small individually, may be multiplied for each query sequence to be mapped and ultimately result in large savings of time and increased efficiency. Thus even statistically small improvements may be very substantial and important to the marketability and usability of the mapping system.

[0047] As described in this application, the apparatus, systems and methods of the invention employ one or more computing devices, generally comprising hardware and/or configurable functional modules as described herein, to collect, store, process, generate and deliver data relevant to DCMOM and also, in some embodiments, to present information and an interface to DCMOM to end users. The DCMOM invention may be performed by an apparatus, system or method that practices a plurality of the steps of the DCMOM method, as described herein. Any particular embodiment need not practice all the steps disclosed herein, and need not practice them in the order described here. It will be appreciated that adding, removing or changing the order of steps will not necessarily take an embodiment outside the scope of the invention.

[0048] Briefly, DCMOM, in certain embodiments, finds the location of a smaller datum (i.e., the query sequence) in a larger datum (i.e., the reference sequence). For example, in a biological application, the query sequence may be the output from a sequencer (e.g., a read) and the reference sequence may be a genome (e.g. a mammalian genome). DCMOM typically accomplishes its goal of sequence matching, assembly and alignment by first identifying where a plurality of portions of the query sequence may be found identically in the reference sequence. Identifying where the portions of the query sequence occur in the reference sequence may be a recursive operation, whereby DCMOM first attempts to match the entire query sequence into the reference sequence. If there is not a perfect match, DCMOM breaks up the query sequence into a plurality of subsequences, and attempts to find the locations of those subsequences in the reference sequence, and so on. This process of breaking up the problem to solve into smaller

problems to solve, and then combining those smaller solutions to form a larger solution, is referred to as "recursive," or a "divide-and-conquer paradigm," or both. Once the plurality of subsequences in the query sequence have been mapped to locations in the reference sequence, or found to not match, then DCMOM may classify the joints (e.g., non-matching elements, discontinuities, and the like) between the mapped subsequences in a manner appropriate to the field of matching. For example, when attempting to map a read output from a biological nucleotide sequence to a reference genome, DCMOM may classify the joints as insertions, deletions, substitutions, or the like.

[0049] DCMOM is, in some embodiments, a recursive method. A recursive method is one that achieves its goal by the repeated application of a set of instructions. For example, an embodiment of DCMOM may include or define a procedure, wherein one of the steps of the procedure is to invoke the procedure itself. Alternatively, an embodiment of DCMOM may define an object, where a part of the definition of the object is the object itself. In some aspects, an embodiment of DCMOM may include an object, function, or procedure which operates by reference to itself. For example, DCMOM may include a function (e.g., a function for mapping a query sequence to a reference sequence) where in one step in the execution of that function is a reference to the function itself. When the function is executed by an embodiment of DCMOM, it may be referred to as an instance, or as an instantiation, of the function. In this case, the function would be considered instantiated. If one instance of the function executes the function again, the second execution of the function may be referred to as a separate instance. The older instance may be referred to as a "parent instance," and the newer instance may be referred to as a "child instance."

[0050] It will be apparent to one of ordinary skill in the art that there are many ways to implement recursive methods, objects, functions, and the like, and that the description herein is illustrative only and does not limit DCMOM to a particular computer language or style of programming, for example, procedural, functional, or object oriented programming. Likewise, any reference in this description to an "instance" of DCMOM is not limited to a particular programming language, implementation, or style such as procedural, functional or object oriented programming.

[0051] DCMOM is also, in some embodiments, a divide-and-conquer method. Divide and conquer means that a large problem to be solved may be broken up into smaller problems, and the solutions of the smaller problems may be combined to form the solution to the large problem. This typically enables a highly parallel approach to the computation of solutions. In the case of DCMOM, when a child instance is created, the inputs or data upon which that child instance operates may be a subset of the inputs or data upon which the parent instance operated. Likewise, the return value or output of an instance of the DCMOM method may be a partial solution, or a portion of the solution, of the larger problem that the DCMOM method is being applied to.

[0052] DCMOM is also, in some embodiments, a more deterministic method than previous mappers. Deterministic means that a method is not subject to the influence of random or outside events. In comparison to previous methods, DCMOM is less subject to the influence of random or outside events. DCMOM is more deterministic compared to other methods because it requires fewer parameters and heuristics to operate, or even no parameters or heuristics to operate, and thus reduces the influence of those parameters and heuristics on the results or output of the method. It also, in the context of nucleotide sequence analysis, does not require the use of an annotated reference sequence.

[0053] Because of its deterministic, recursive, divide-and-conquer architecture, DCMOM succeeds in detecting matches between query and/or reference sequences undetectable by existing methods, and regardless of the length of the query or reference sequences. For example, in the field of biological nucleotide sequence alignment, particularly, mapping of a transcriptome onto a genome, DCMOM succeeds in detecting novel exons, junctions and variants (such as substitution, insertion, deletion) at the level of a single read. Regardless of the read length, DCMOM can align both RNA-Seq and DNA-Seq data with significant advantages, as described herein, over existing systems. DCMOM can also detect Single Nucleotide Polymorphisms ("SNPs"). Additionally, especially with respect to RNA-Seq data, DCMOM is able to map a read that spans multiple exons. DCMOM is also capable of detecting novel, very small exons (e.g., less than the seed length) as well as possible alternative splicing junctions.

[0054] DCMOM, in some embodiments, utilizes and incorporates an adapted and novel version of the MOM method. A complete description of the MOM method can be found in

Eaves and Gao, 2009, *MOM: maximum oligonucleotide mapping,* Bioinformatics 25:7 2009, p. 969-970, hereby incorporated by reference into this application in its entirety. Briefly, in some embodiments MOM operates under the assumption that errors in the query sequence are more likely near the ends of the sequence, and thus that the longest possible match between a query sequence and a reference sequence may exclude the ends of the query sequence. MOM thus matches a seed portion of the query sequence to the reference sequence, and then extends the alignment in both directions from the seed, either until a predetermined maximum number of mismatches is reached, or until the entire query sequence is matched. Thus extended, the seed plus the additional adjacent matching elements are known as a node or root node. MOM may, in some embodiments, include a parameter whereby a configurable number of mismatches is tolerated in a match. For example, any number of mismatches between about 1 and about 100 mismatches may be configured as acceptable in a match. In some embodiments, between 15 and 40 mismatches, between 10 and 20 mismatches, between 15 and 30 mismatches, or between 20 and 50 mismatches, may be configured as acceptable in a match. The term seed or seed portion refers to a segment, or subsequence, of a sequence. In general, in DCMOM, no mismatches in a seed are tolerated. Extending an alignment means to identify additional elements of the query sequence which match the reference sequence, including - if so configured - a number of tolerated mismatches. Then, MOM trims elements from first one side of the query sequence, and subsequently attempts to extend the match on the other side of the query sequence. MOM then repeats the process by iteratively trimming elements from one or the other sides of the query sequence so as to find the longest possible match.

[0055] Referring to **FIG. 14**, an instance of the DCMOM method typically begins **14.100** by creating a hash table **14.110** from a reference sequence, using a given initial seed length **14.130**, and storing that hash table in a hash table memory **14.150**. Here, a hash table refers to a tabular data structure where some data, typically a sequence of elements, is mathematically hashed to create a new value (i.e., the 'hash' or the 'index'), which is associated with other data, typically the location of that sequence of elements within the reference sequence. In the case of DCMOM, a hash table generally maps a particular sequence of elements to a location in the reference sequence where that sequence of elements may be found. The term "seed length" here refers to the length of the subsequence of elements which is to be hashed for the purposes of creating the hash table. In some embodiments, the location of each possible subsequence of a given length

may be stored in the hash table. The reference sequence may be any kind of data. In some embodiments, the initial reference sequence **14.120** may be a nucleotide sequence of any source or organism, such as, for example, of an animal, mammal, human, non-human mammal, plant, virus, bacteria, yeast, fungus, eukaryote, prokaryote, or other organism. In preferred embodiments, the reference sequence is a genome, but it may also be one or more chromosomes of a genome, a transcriptome, a library of nucleic acids, synthetic nucleic acid, library clone of nucleic acid, or any other source of nucleic acid.

[0056] If the instance of DCMOM is a child instance, the reference sequence may be only a portion of the initial reference sequence, e.g., the portion to the left or right on a contiguous nucleotide sequence, e.g., a chromosome of the reference genome sequence, of an already mapped root node. A root node, as described herein, refers to a seed which has been mapped to a location in the reference sequence, and the mapping optionally expanded to include adjacent matching elements. The reference sequence may be annotated or not, but DCMOM does not require that the reference sequence be annotated. If this instance of DCMOM is the first instance, the initial seed length **14.130** may be received from a configuration parameter. This initial seed length may be any number of elements, for example, 5 elements, 10 elements, 15 elements, 20 elements, 25 elements, 30 elements, 40 elements or 50 elements or more. In specific embodiments, the initial seed length is 15 to 25 elements, 10 to 30 elements, at least 20, 25, 30, 35, 40 or 50 elements. In some embodiments, the seed length may the same length as the initial seed length or, preferably, may be less than the initial seed length on subsequent iterations or recursions of DCMOM, for example in child instances. For example, the seed length may decrease for each deeper recursion of DCMOM. In certain embodiments, the seed length decreases by 1 element for each recursion, 5 elements for each recursion or 10 elements for each recursion, the a minimum seed length of 1 element. In specific embodiments, seed lengths after the initial instance of DCMOM are 3 elements, 5 elements, 8 elements, 10 elements, 12 elements, 15 elements, 20 elements or 25 elements or are 1 element, 2 elements, 3 elements, 5 elements, 10 elements or 15 elements less than the initial seed length.

[0057] The hash table generated in step **14.110** may be implemented using a tabular data structure, or with any of several different types of data structures which would enable one to index or address a body of data. For example, the hash table may be implemented using a dictionary type data structure which maps a hash value to a location on the reference sequence.

In some embodiments, DCMOM may also use other varieties of data structures instead of or in addition to a hash table to store mapping information and/or hash information, including but not limited to binary trees, linked lists, pointers, or any other suitable data structure presently known or later discovered which is able to store mapping data. As previously discussed, the hash value may be created from a seed length portion of the reference sequence, and the location on the reference sequence may correspond to the location of the aforementioned hashed, seed-length portion. When the hash table is generated it may, in some embodiments, be stored or cached **14.150** to memory or disk.

[0058] Once a hash table has been created, the next step **14.200** involves searching for the location of a query sequence **14.140** in the hash table. A query sequence may be any arbitrary data, of any arbitrary length. In some embodiments, the query sequence **14.140** may be a "read" output from a nucleic acid sequencer, representing a nucleic acid sequence, DNA, RNA, e.g., mRNA sequence, of any length or description, and the elements of the query sequence may be the nucleotide elements of the read. For example, a read may be a single end read, a paired end read, a uniform length read, or a variable length read. Further, a read may be between about 5 elements and about 200 elements long, between about 10 elements and 150 elements long, or approximately 30 elements long, 50 elements long, 80 elements long, 100 elements long, 120 elements long, 150 elements long, 180 elements long, 200 elements long, or 250 elements long. In specific embodiments, the read is 75 elements long, 100 elements long or 125 elements long. Reads may be at least 100 elements, 200 elements, 500 elements, 750 elements or 1000 elements long or even longer. Alternatively, a read may be much longer, for example up to about 5,000 or about 10,000 elements long. Alternatively, a read may be an entire chromosome or genome, millions or billions of nucleotides in length. A sequencer may be connected to or in communication with the apparatus or system that practices DCMOM, or the sequencer may be within the same apparatus or system. Future sequencing apparatuses may generate reads far in excess of 10,000 elements, up to and including entire genes, chromosomes, or even genomes. No matter the length of the read, embodiments of DCMOM can map such reads. In some embodiments, for a child instance of DCMOM, the query sequence may be a portion of a read, e.g., the portion to the left or right of an already mapped node in the parent instance.

[0059] The query sequence is searched for within the hash table so that a location of the query sequence within the reference sequence may be identified. Searching the hash table or

other data structure may occur by calculating the hash value of a plurality of seed-length portions of the query sequence, preferably starting on the left hand side, but optionally starting on the right hand side or in the middle of the query sequence, and checking to determine if a hash value is found in the data structure. If a hash value is found **14.300** in the data structure, then the seed-length portion may be identified as a root node, and searching stops. DCMOM may be configured to accept only root nodes of a certain minimum length, or DCMOM may accept root nodes of any length. If a minimum root node length is configured, DCMOM may use any minimum root node length. For example, DCMOM may accept only root nodes of at least about 5 elements, 10 elements, 15 elements, or 20 elements, or between about 10 and about 20 elements, or between 10 and 30 elements, or up to about 50 elements, or as few as about 3 elements. If the root node is unable to be extended to at least the aforementioned minimum length, the root node may be rejected and additional seed-length portions of the query sequence will be searched. If no seed length portions of the query sequence are found in the hash table **14.310** or other data structure, then this instance of the method stops and returns an indication that the query sequence could not be found in the reference sequence.

[0060] If a root node was identified in the previous step, the next step is to maximally extend the match to the right and/or left sides of the root node **14.400.** To maximally extend the root node means to compare the elements of the query sequence adjacent to the root node with the elements of the reference sequence adjacent to the location where the root node sequence occurs on the reference sequence. This comparison is performed to identify the largest contiguous sequence of elements, or node, where each element of the query sequence corresponds to and matches to an element of the reference sequence. At the end of this process, three subsequences may be identified from the query sequence – some of which may have length zero: the Left Side Data **14.410,** the Maximally Extended Match **14.420,** and the Right Side Data **14.430.** The Maximally Extended Match **14.420** may encompass all of, or only a part of, the query sequence. Either the Right Side or Left Side Data may have zero elements, for example, if the Maximally Extended Match **14.420** extends all the way to the left side of the query sequence **14.140** then there will be no Left Side Data **14.410.** The Right Side Data **14.430** and/or the Left Side Data **14.410** may be used in a recursive instance of DCMOM as a query sequence. Likewise, the reference sequence may be divided into three subsequences: the Matched Reference Sequence **14.450** corresponding to the area of the reference sequence matched by the Maximally Extended

Match **14.420**, and those portions to the sides of the Matched Reference Sequence, the Left-of-Match Reference Sequence **14.440** and the Right-of-Match Reference Sequence **14.460**. The Left-of-Match and Right-of-Match Reference Sequences may be used in a recursive instance of the method as the reference sequence.

[0061] If the Maximally Extended Match **14.420** comprises **14.500** the entire query sequence **14.140,** then this instance of the DCMOM method is concluded, and it returns an indication **14.510** that the whole query sequence matched the reference sequence, and the location thereof. Alternatively, if the Maximally Extended Match does not comprise the entire query sequence, execution of this instance continues.

[0062] If there is no mismatch **14.600** to the right of the Maximally Extended Match **14.420**, then the Right Side Result **14.630** is null **14.610**, and there will not be a right side child instance. Likewise, if there was no mismatch **14.700** to the left of the Maximally Extended Match **14.420,** then the Left Side Result **14.730** is null **14.720** and there will be no left side child instance.

[0063] However, if there is a mismatch to the right or left or both sides of the Maximally Extended Match **14.420**, then this instance of DCMOM will create one or more child instances of the DCMOM method to attempt to match the elements of the query sequence, including and beyond the mismatch, on the respective side of the query sequence, with the new query sequence input to the child instance being the Right Side Data **14.430** or Left Side Data **14.410**, respectively. In some embodiments, the reference sequence for the child instance may be the initial reference sequence, or it may be the Left-of-Match **14.440** or Right-of-Match **14.460** Reference Sequences.

[0064] If only either the right portion or the left portion of the reference sequence needs to be searched, the search space may thus be reduced by about 50%, on average. For example, if in a particular application of DCMOM the query sequences directional, such that it is known that the Right Side Data will not be mapped to the Left-of-Match Ref. Seq., then the Left-of-Match Ref. Seq. may be excluded from a reference sequence to be searched by a child instance of DCMOM when the query sequence is the Right Side Data.

[0065] As a non-limiting example, if there is a mismatch on the right **14.600** of the Maximally Extended Match **14.420**, then a child instance DCMOM may be instantiated **14.620**, with the Right Side Data **14.430** serving as the query sequence for the new instance, and the

Right-of-Match Reference Sequence **14.460** serving as the reference sequence for the new instance. The seed length for this new DCMOM instance may be the same length as the seed length for the original DCMOM instance, or in some embodiments it may be longer or it may be shorter than the seed length from the original, or its parent, DCMOM instance. For example, the seed length may decrease for each deeper recursion of DCMOM. In certain embodiments, the seed length decreases by 1 element for each recursion, 5 elements for each recursion or 10 elements for each recursion, the a minimum seed length of 1 element. In specific embodiments, seed lengths after the initial instance of DCMOM are 3 elements, 5 elements, 8 elements, 10 elements, 12 elements, 15 elements, 20 elements or 25 elements or are 1 element, 2 elements, 3 elements, 5 elements, 10 elements or 15 elements less than the initial seed length. The result returned by that instance of DCMOM is stored as the Right Side Result **14.630.**

[0066] Similarly, if there is a mismatch on the left **14.700** of the Maximally Extended Match **14.420**, then a new instance of DCMOM will be instantiated, with the Left Side Data **14.410** serving as the query sequence for the new instance, and the Left-of-Match Reference Sequence **14.440** serving as the reference sequence for the new instance, with the result being stored as the Left Side Result **14.730.** It will be appreciated that due to the recursive nature of DCMOM, should a child instance on either or both of the left and right hand side encounter a mismatch in its query sequence, it may create additional child instances of DCMOM, and those child instances may themselves create child instances, and so on. Each of these child instances will complete and return their results to the parent instance, indicating where their input query sequences matched their input reference sequences.

[0067] From the foregoing, it will be appreciated that the return value or output of an individual instance of DCMOM may be: that there was no match between the query sequence and the reference sequence **14.310**; that there was a complete match between the query sequence and the reference sequence **14.510**; or that there was a partial match between the query sequence and the reference sequence (see **14.600** and/or **14.700**). In other words, an instance of DCMOM may find either zero or a plurality of nodes in its query sequence which match locations on the reference sequence. DCMOM may also identify mismatches adjacent to or between nodes that do not match locations on the reference sequence. These mismatches are known as joints or edges, and the number of mismatched elements between two nodes, if any, is known as the

distance between those nodes. DCMOM may also identify two nodes whose locations on the reference sequence overlap in certain cases.

[0068] An instance of DCMOM may combine the Maximally Extended Match **14.420**, the Left Side Result **14.730**, and the Right Side Result **14.630** by assembling **14.800** them into the Assembled Result **14.900**. The instance may, in some embodiments, process its individual assembled result **14.940** to merge nodes. Alternatively, the child nodes may return their results without processing, and the results may, in some embodiments, be processed by the parent instance, the first instance, or not at all. Thus, the Assembled Result **14.900** may represent, in some embodiments, an Initial Mapping Tree ("IMT"), if the nodes have not been classified and/or merged. An IMT is the output data from an instance of DCMOM, without complete node merging, represented visually as a tree like structure. Alternatively, the Assembled Result **14.900** may represent a Final Junction Tree ("FJT"), if node merging has already been performed on the Right Side Result **14.630** and/or Left Side Result **14.730.** The FJT is, in certain embodiments, a visual representation of the output data from an instance of DCMOM, similar to an IMT, but where some or all of the nodes have been merged together by classifying the joints between them. The Assembled Result **14.900** may in some embodiments be further processed **14.940** by the current instance of DCMOM, including by additional node merging. In some embodiments, further processing or node merging will not be performed on the Assembled Result **14.900** and it will simply be returned **14.950.**

[0069] An exemplary embodiment of the IMT and FJT aspects of the invention is shown and displayed in **FIG. 1. FIGs. 1A** and **B** depict an initial mapping tree created by DCMOM as it identifies maximal node matches between a read and a reference sequence. **FIG. 1C** shows the merging of two nodes, creating **FIG. 1D** the Final Junction Tree ("FJT"). **FIG. 1E** illustrates some of the different classifications of mismatches in certain embodiments – Single Nucleotide Polymorphisms ("SNPs"), Insertions, and Deletions.

[0070] Returning to **FIG. 14,** the DCMOM method provides a step for merging the nodes **14.940** of the Assembled Result **14.900** to classify the edges (or "joints") between the nodes. Node merging is the process of classifying the joints between the nodes. In some embodiments, node merging may start by classifying the joint between the two left most nodes and move toward the center until it reaches the root node, at which point node merging begins again with

the two right most nodes until it ends at the root node. The types of classification applied to the joints will change depending on the field that DCMOM is used in. For example, in the nucleotide sequencing field, the joints will be classified, for example, as substitutions, insertions, deletions or splice junctions. However, as described herein, DCMOM is not limited to nucleotide sequence and node merging may use different criteria or classifications in other applications.

[0071] Referring to **FIG. 14,** either after node merging has taken place or without merging nodes, an instance of DCMOM preferably returns its result **14.950**. If the instance was a child instance of DCMOM, then the parent instance may use the result as either a Right Side Result **14.630** or a Left Side Result **14.730.** However, if the instance was the first DCMOM instance, the result may then be outputted, stored into memory, printed, node merged, or presented to the user via a video or text display device, and/or through the use of a separate viewing software as described herein. The output data, in certain embodiments, may include an identification of nodes comprising: the elements of the query sequence that match the reference sequence, the locations where elements of the query sequence match the reference sequence, the locations where elements of the query sequence do not match the reference sequence, and/or a classification or description of the joints between nodes. The output data may be provided to a parent instance of DCMOM, a different apparatus, system, or method; or a user interface or other program used to visualize the data. DCMOM may include a module for visually presenting results to the user. In certain embodiments, the viewing software may be *Integrated Genome Viewer, Gene Browser, Gene Workbench,* or any other viewing software. Output from the results of compiling the mapped and assembled query sequences can be in any form known to one in the art, for example an SAM file or a BAM file.

[0072] **FIG. 4** depicts an example of a specific application of the DCMOM node merging process, where the query and reference sequences are nucleotide sequences. Where the distance between a root node and a left node on the reference sequence is one or more elements (in **FIG. 4,** the reference sequence element is an 'A', e.g., a nucleotide or base), and there is the same number of elements between them in the query sequence (in **FIG. 4,** the query sequence element is a 'T'), then the two nodes can be merged into one node, with the joint between them classified as a substitution. If the substitution has one mismatched element, it is known as a Single Nucleotide Polymorphism ("SNP").

[0073] **FIG. 5** depicts another example of a specific application of the DCMOM node merging process, where the query and reference sequences are nucleotide sequences. In **FIG. 5,** DCMOM merges two nodes by categorizing the edge between them as an insertion. In **FIG. 5,** there is no distance between the two nodes on the reference sequence, but there are elements in between them on the query sequence. This is shown in **FIG. 5A,** which depicts a Root Node and a Right Node 1, with zero distance between the nodes on the reference sequence but one element between them in the query sequence (in **FIG. 5A,** a 'T'). **FIG. 5B** depicts an alternative type of insertion where there are no mismatched elements between the nodes, yet the location of the two nodes on the reference sequence overlaps, as depicted by the subsequence TTTT rendered in light blue, bold italics.

[0074] **FIG. 6** depicts another example of a specific application of the DCMOM node merging process, where the query and reference sequences are nucleotide sequences, known as deletion. A deletion occurs where there is a distance between two nodes on the reference sequence, and a shorter distance, mismatches or overlap between the nodes on the query sequence. **FIG. 6** depicts two deletions, which differ from each other by the number of deleted elements. On the left, many elements on the reference sequence (depicted by ellipses) are missing between Left node 1 and Left node 3. On the right, one element (in **FIG. 6**, a 'C') is missing between Left node 3 and Left node 2. Each of these may be classified as a deletion. However, the junction where many elements were missing, or in other words where there was a larger distance between the nodes in the reference sequence, may optionally be classified as a junction. Typically, a deletion is identical to a junction except that where the distance between the joints is greater than a threshold, the joint is classified as a junction rather than a deletion. The arbitrary distance threshold may be any number of elements, and it may be pre-set or set dynamically. For example, the distance threshold could be between about 1 element and about 100 elements. In certain embodiments, the threshold may be between about 9 and about 30 elements. A deletion where the distance between the nodes is below the threshold is considered to be a mere deletion. In some embodiments even though nodes with a substitution, deletion, or insertion are merged, nodes with a junction between them may not be merged, but remain independent in the FJT as shown in **FIG. 7**.

[0075] Through the above described process of node merging, DCMOM may thus combine the results of a plurality of instances of the DCMOM method into a single junction tree,

unconstrained by the number of mismatches, errors in the query sequence, or variations between the query sequence and the reference sequence.

[0076] The method further then provides for mapping numerous query sequences (e.g., reads from sequencing of a nucleic acid sample such as mRNA expressed in a cell or tissue type, i.e., a transcriptome) onto the same reference sequence to assemble the DCMOM output from these each of these plurality of query sequences, i.e, that have been mapped to the reference sequence and undergone the node merging process, into an assembled sequence of the nucleic acids within the sample. For example, in the case of expressed mRNA, the output includes sequences of the mRNA molecules within the sample, including different splice variants (having different combinations of exons and/or removal of different introns) of mRNAs expressed from the same gene. In this way, the entire transcriptome can be assembled and mapped to the reference sequence.

[0077] The DCMOM methods can process, align and assemble large numbers of reads, for example, reads resulting from sequencing of a genome or transcriptome, which results in billions, 10 billions or even 100 billions of individual reads to be processed as query sequences in the DCMOM methods. In view of the efficiencies of the methodology and the ability to process query sequences in parallel, the method of the invention can analyze, assemble and align nucleotide sequences from a biological sample, e.g., a transcriptome or genome, in hours, for example, within 5 hours, 6 hours, 8 hours, 10 hours or 24 hours using computing methodology available for bioinformatics analysis. The DCMOM method may use an annotated or non-annotated reference sequence. An advantage of the method of the invention is that nucleotide sequence can be mapped, assembled and aligned accurately to a non-annotated reference sequence, identifying even small (less than 5, 10, 15, 25 or 30 nucleotides) exons and introns, short (less than or equal to 10, 5, 3, 2 or 1 nucleotide) insertions, deletions or substitutions. By reference to a non-annotated reference sequence, the method of the invention is not restricted to reference sequences previously annotated and the mapping, assembly and alignment is not prejudiced by the annotation, permitting the identification of new splice junctions, insertions, deletions or substitutions. The output of the sequence analysis can be provided in any file format available and known to one skilled in the art, for example, a SAM file or a BAM file.

[0078] It will be apparent to one of ordinary skill in the art, based on the disclosures contained herein, that the creation of additional instances of the DCMOM method to investigate portions of the reference and query sequences utilizes a novel divide and conquer paradigm with recursion, and enables DCMOM to obtain high performance and parallelism from any number of CPUs or cores to process, assemble and align large numbers of query sequences, for example, those derived from sequencing the mRNA or genomic DNA of a biological sample. It will also be apparent that the recursive nature of the DCMOM method enables DCMOM to map a query sequence to a reference sequence, regardless of the number of mismatches that are found between the query and reference sequence. In some embodiments, recursion may stop when the length of that portion of the query sequence to be used as input falls below an arbitrary threshold, for example, recursion may stop when the remaining unmatched portion of the query sequence (e.g., the Left Side Data **14.410** or the Right Side Data **14.430)** reaches a configured Minimum Query Sequence. The Minimum Query Sequence may be any value. For example, the Minimum Query Sequence may be between about 4 and about 12 elements. Alternatively, it may be about 3 elements or less. Alternatively, there may be no Minimum Query Sequence, or it may be set to a special value (such as zero) which indicates that recursion may continue until the query sequence is only a single element.

[0079] In some embodiments, DCMOM uses a tree data structure, such as a binary tree, that can represent and manipulate hierarchical data with nodes and edges, to store the IMT and/or FJT. In a typical tree data structure, a node is a data structure containing content, and an edge represents the connection or relationship between two nodes. Further, in some embodiments, DCMOM may use a particular type of tree data structure known in the art as a binary tree as either a data structure for storing the IMT and/or FJT, or, in some embodiments, as a main data structure for storing the reads and/or reference sequences. A binary tree is useful in this regard because its structure lends itself to DCMOM's recursive method and it provides an efficient way to process the node merging operations, particularly splicing and variant detection, inherent in the biological analysis applications of DCMOM as described herein. As shown in **FIG. 1,** a read in the DCMOM method may be represented with a tree by dividing the read using the maximum query matching method. Next, nodes in the tree are revisited and merged with each other depending on the relationship between them. Thus, the FJT represents a read spanning one or more exons. There are many tree structure variants that DCMOM could utilize. In some

embodiments, a tree data structure could replace the hash table data structure wherever hash tables are called for in the invention.

[0080] Because DCMOM's divide and conquer strategy is capable of identifying a match even when a query sequence may have multiple mismatches, insertions and deletions, DCMOM is able to map even a short read with multiple exons, indels, and SNPs, as shown in **FIGs. 2** and **3**.

[0081] In some embodiments DCMOM analysis generally comprises three steps: 1) initial mapping, 2) node merging and, 3) output generation. **FIG. 2** depicts, for an exemplary application of DCMOM to biological nucleotide sequences, a query sequence (**Read I**), a reference sequence (**Genome**), and a plurality of the initial mapping steps (**a** and **b**) of the DCMOM method. As shown in **FIG. 2**, **Read I** is an mRNA fragment from three exons (i.e., **Exon1**, **Exon2** and **Exon3**), and excluding two **Introns.** **Read I** includes a substitution, a deletion and an insertion. Here, **Genome** indicates a region in a reference genome corresponding to **Read I**. Further, in **FIG. 2**, **Exon** refers to a portion of the reference sequence that matches a portion of a read and/or is translated into mRNA, as opposed to an **Intron**, which does not match a portion of the read and/or is not translated into mRNA. Again referring to **FIG. 2**, an underlined nucleotide element in red in the **Genome** indicates a deletion, which means the nucleotide is present in the **Genome** but not in **Read I**. A nucleotide in red in both **Read I** and **Genome** indicates a substitution. In **FIG. 2**, an insertion is presented by an element in bold italic blue, and indicates the case where one or more nucleotides of **Read I** are not present at the corresponding location in the **Genome**. (*See, e.g.,* the first element "T" in the Right unmapped segment). At least from this description of **FIG. 2,** it will be apparent to one of ordinary skill in the art that DCMOM is able to match a sequence containing multiple exons.

[0082] Continuing with **FIG. 2,** DCMOM first identifies in **FIG. 2(b)** a **Root node** and maps it to the reference sequence**.** The root node is bounded on the left by the **Left unmapped segment**, and on the right by the **Right unmapped segment.** In some embodiments, DCMOM attempts to map the **Right unmapped segment** to the area of the reference genome to the right of the root node by creating a child instance of DCMOM. This child instance of DCMOM will use the **Right unmapped segment** as the query sequence, and the area of the reference sequence to the right of the root node as the reference sequence. Similarly, for the **Left unmapped**

segment, another child instance of DCMOM method is instantiated, but using the **Left unmapped segment** as the query sequence, and the area of the reference sequence to the left of the root node as the reference sequence. In some embodiments, the length of the reference sequences for the child instances will be about 30,000 elements. As described herein, as instances of DCMOM identify mismatches on in the segment **FIG. 2C**, those instances will create child instances of DCMOM so as to recursively attempt to map the query sequence onto the reference sequence. In this way, DCMOM may identify the entire read as either matched nodes or unmapped joints.

[0083] Continuing with **FIG. 2**, in **FIG. 2C** DCMOM has identified **Left Node I**, and another **Right unmapped segment**. **Left Node I** was not expanded to include the elements of this new **Right unmapped segment** because there is an **Intron** between them on the **Genome**. Thus, another recursive instance of DCMOM is instantiated to map the new **Right unmapped segment**. DCMOM in this way may, in some embodiments, produce a binary tree representing the local alignment of a read to a reference genome.

[0084] **FIG. 3** depicts the example of **FIG. 2**, but at a later stage in the initial mapping process. Each of the nodes in the query sequence (**Read I)** have been mapped to the reference sequence (**Genome)**. **FIG. 3** thus depicts an initial mapping tree (IMT). The output of the top level, parent instance of DCMOM in **FIG. 3** is ready to be node merged.

[0085] **FIG. 7** depicts an example of the output of the DCMOM node merging process. The nodes from **FIG. 3** have been merged to form a final junction tree (FJT) indicating that **Read I** (the query sequence) maps closely to **Genome** (the reference sequence) with three exons, two junctions, a deletion, a substitution, and an insertion.

[0086] There is no inherent limit to the number and type of variants in a single read that DCMOM program can process, though a user may, in some embodiments, configure such limitations into DCMOM apparatus, system or method. In biological applications of DCMOM, this issue is directly related to how many exons (i.e., junctions) a read is able to span. Similarly, it is unnecessary to specify the number of permitted nodes or joints in a read, because the recursive nature of DCMOM enables it to identify an unlimited number of nodes or joints with little computational overhead.

[0087] The ability to process reads spanning multiple junctions and a high tolerance for errors make the apparatus, system and methods of the invention ideal for reconstruction and alignment of RNA-Seq data, especially with longer, lower quality reads. Additionally, computational time is favorable to that of other tools due to the simplicity of the method and the ability to parallelize the computation across multiple machines and/or processor cores within a machine to analyze, map and assemble numerous reads arising from a biological sample, e.g., from RNA-Seq data.

Table 1

Parameter	Description	Default
Read_len	Read Length	100
Seed_len	Seed Length	25
Min_root_len	Minimum match length for root node	30
Min_frag_len	Minimum length of a segment	6
Search_Range	Search range for segment mapping	30,000
Num_threads	Number of threads (cores) available	40
intron_len	Number of elements between two root nodes for an intron	30

[0088] Depending on the particular embodiment of DCMOM, the number of parameters may vary; however, as shown in Table 1, there are several parameters which may be implemented for the DCMOM method itself, and most of them are significantly related to the performance of the method. It will be appreciated that each of the values provided for the parameters herein are exemplary only; other values may be used, including values not specifically described herein, because the invention is not limited by these values, nor is it limited by the length of the query or reference sequences it can map. Where a value or range of potential values is described herein, it is merely a suggested range of values. The precise values for the parameters chosen by the user or implementer of a DCMOM apparatus, system or method will depend on the application to which DCMOM is being applied. The values for the parameters shown in the table above are preferable for certain biological nucleotide applications of DCMOM. In some embodiments, fewer or more parameters may be implemented in a DCMOM apparatus, system or method. In some embodiments, all of the parameters shown in the table above could take any of the corresponding values as shown above.

[0089] With respect to seed length parameter (in the example of Table 1, "Seed_len"), the longer the seed becomes, the faster the computation for identifying the location of that seed becomes. However, lengthening the seed length increases the chance that a read will not contain a seed match. The seed_len specifies the initial seed length and this could be any value with no maximum limit. For example, in some embodiments, the seed_len could be any value between about 1 and about 30,000. In other embodiments, it could be anywhere between about 30,000 and about 100,000. In some embodiments, and preferably for biological nucleotide sequence alignment, the seed_len may be as little as about 2 or as large as about 30. In an embodiment, the seed_len is about 18 to about 25. In an embodiment, the seed_len is about 15 to about 20. In certain embodiments the seed_len is 5, 10, 15, 20, 25, 30, 40 or 50 or more. In specific embodiments, the seed_len is 15 to 25, 10 to 30, or at least 20, 25, 30, 35, 40 or 50.

[0090] The read_len, in some embodiments, specifies the read length, or the length of the query sequencs. This could be any value, or it could be unspecified. For example, the read_len could be any value between about 1 and about 50,000. In some embodiments, read_len could be about 75, 100, or 125 or between about 100 and about 1000.

[0091] The min_root_len parameter indicates a minimum size of a node which a DCMOM instance will identify as a root node. The min_root_len can, in some embodiments, be any value between about 1 and about 30,000. In some embodiments, the min_root_len will be between about 10 to about 50 elements. In specific embodiments, the min_root_len will be about 5, 10, 15, or 20, or between about 10 and about 20, or between 10 and 30, or up to about 50, or as few as about 3. In some embodiments, at least one of the exons that the read spans must be longer than the min_root_len.

[0092] The min_frag_len parameter defines the minimum size of node that DCMOM will identify as a non-root node and can, in some embodiments, be any value between about 1 and about 30,000. It also determines the size of a shortest exon that the method will report. In some embodiments, the min_frag_len parameter could also be between about 3 and about 10 and, in certain embodiments may be, 4, 5, 6, 7, 8, 9, 12, 15, 20 or 30.

[0093] The search_range parameter determines the range beyond a node boundary in which DCMOM will search for a match on the reference genome. Search_range can, in some embodiments, be any value between about 1 and about 2,000,000. The search_range parameter

thus defines the length of the longest intron which DCMOM will be able to identify between two exons. In some embodiments, the Search Range parameter may be between about 3,000 and about 50,000.

[0094] The intron_len defines the minimum number of elements that must be found between two adjacent mapped nodes for the joint between the two nodes to be classified as an intron. The intron_len may, in some embodiments, be between about 2 and about 10,000 elements or nodes. In some embodiments, intron_len may be the same as min_frag_len. Alternatively, intron_len could be between about 3 and about 32 and, in certain embodiments, in certain embodiments may be, 4, 5, 6, 7, 8, 9, 12, 15, 20 or 30.

[0095] Lastly, the num_threads parameter defines how many computational threads of the DCMOM method may be running simultaneously on a DCMOM apparatus or system. Num_threads may take any value supported by the underlying operating system and/or hardware platform(s) that DCMOM runs on. The num_threads may, in some embodiments, be between about 1 and about 5,000. Alternatively, NumThreads may be between about 2 and about 40.

Hardware Systems

[0096] Unless specifically stated otherwise herein, it will be appreciated that throughout the description of the DCMOM apparatus, systems and/or methods, discussions utilizing terms such as "processing" or "computing" or "collecting" or "analyzing" or "calculating" or "determining" or "displaying" or "presenting" or "storing" or "selecting" or "identifying" "storing" or "software" or "module" or "subroutine" or "program" or "instance" or the like, can refer to the actions or processes of a data processing system, or similar electronic device, that manipulates and transforms data represented as physical (electronic, magnetic, nuclear or quantum) quantities within the system's registers and memories into other data similarly represented as physical quantities within the system's memories or registers or other such information storage, transmission or display devices.

[0097] The embodiments of DCMOM can relate to an apparatus, system or method for performing one or more of the functions described herein. An apparatus or system may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Any other data, including data generated by the program, data input to the program, or data output from the

program, may also be stored in the computer. When such data is stored in a computer, it may be stored in a memory. A memory as used herein refers to one or more of machine (e.g., computer) readable storage media, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs and magnetic-optical disks, read only memories (ROMs), random access memories (RAMs) erasable programmable ROMs (EPROMs), solid state drives (SSDs), electrically erasable programmable ROMs (EEPROMs), flash memory, magnetic or optical cards, portable data storage devices, or any type of media suitable for storing electronic instructions or data, and may be coupled to a bus or in wireless communication with other elements.

[0098] Some embodiments of DCMOM described herein may be described as software executed on at least one computer, though it is understood that embodiments can be configured in other ways and retain functionality. The embodiments can be implemented on known devices such as a server, a personal computer, a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit element(s), and ASIC or other integrated circuit, a digital signal processor, a hard-wired electronic or logic circuit such as a discrete element circuit, a server, a tablet computer, a wireless handheld device, a cell phone or smartphone, a netbook, an electronic flight bag, or the like. Specific devices which might be used as a computing component of the system include iPad, Android, Kindle or other tablet computers; iPhone, Android, Blackberry, or other cell phones or smart-phones; or laptop computers such as those commonly manufactured by Apple, Lenovo, Dell or HP; or server or mainframe computers such as those manufactured by IBM or others. In general, any device or devices capable of implementing the one or more of processes described herein can be used to implement the systems and techniques according to this invention.

[0099] It will be appreciated that the various components of the DCMOM invention can be located at distant portions of a distributed network and/or the internet, or within a dedicated secure, unsecured and/or encrypted system. Thus, it should be appreciated that the components of the apparatus or system can be combined into one or more devices or co-located on a particular node of a distributed network, such as a telecommunications network. As will be appreciated from the description, and for reasons of computational efficiency, the components of the system can be arranged at any location within a distributed network without affecting the operation of the system. Moreover, the components could be embedded in a dedicated machine.

[00100] Furthermore, it should be appreciated that any of the various links connecting the various elements of DCMOM can be wired or wireless links, or any combination thereof, or any other known or later developed element(s) that is capable of supplying and/or communicating data to and from the connected elements. For example, the links or networks might be ethernet, 802.11 Wi-Fi, Bluetooth, GSM, GPRS, EDGE, 3G, 4G, LTE, satellite network links, fiber optic links, HAM radio, peer-to-peer, mesh network, or any other type of data communications network. It shall be understood that the invention may dynamically update its data and outputs depending on the incoming data and information received from these links or networks. The terms determine, calculate and compute, and variations thereof, as used herein are used interchangeably and include any type of methodology, process, mathematical operation, or technique.

[00101] DCMOM, as described and claimed herein is not to be limited in scope by the specific embodiments disclosed since these embodiments are intended as illustrations of several aspects of the invention. Any equivalent embodiments are intended to be within the scope of this invention. Indeed, various modifications of the invention in addition to those shown and described herein will become apparent to those skilled in the art from the foregoing description. Such modifications are also intended to fall within the scope of the appended claims.

Configurable Functional Modules

[00102] In certain embodiments, DCMOM may consist of one or more configurable modules in communication with each other. These modules may be implemented as instructions stored on a processor readable storage medium which are executed by a computing node. As discussed above, these modules may be co-located on a single computing node, or they may be located in physically distinct locations. Each of the modules may be coupled via a data communications network so as to communicate with any or all of the other modules in the system as desired. Generally, unless specifically stated to the contrary in the following discussion, each module is at least in communication with one other module. In the discussion of any module, it will be appreciated that any description of a connection or communication between that module and other modules does not limit the invention; in fact each module can be configured to communicate with any other module or modules as desired.

[00103] In certain embodiments of DCMOM a plurality of the modules may interface with other modules, including but not limited to other modules described herein as well as third party modules, and share, request or provide data; generate or respond to queries; store or retrieve data; transmit or receive signals; or do any of the other common tasks or manipulations which computer systems are commonly known to perform with data by those skilled in the art. It will be appreciated that any description of a particular module performing or not performing a specific functionality does not limit the invention, as the functionality of any module could be combined with any other module to form a single module; likewise the functionality of a single module could be divided between two other modules to form two modules. In some embodiments, some modules will share data only with specific other modules, or not at all, or as configured by a user.

[00104] It will further be appreciated by one skilled in the art that the written descriptions of DCMOM, described herein to enable one of ordinary skill in the art to practice this invention, are illustrative only, and elements of the invention may be composed of modules without specific mention of such modularization herein. Furthermore, modules may be combined with one another, or separated from one another, to form hybrid modules or sub-modules which are still within the scope of this invention. Likewise, it will be appreciated that any descriptions of the data outputs and inputs to the modules are illustrative only and that the invention is not so limited. A module could receive input data from any other module or data source, and could output its data, analyses, or alerts to any other module, to an end-user, or to any other recipient or device. Similarly, it should be appreciated that a particular analysis or other technique described in conjunction with a module is not limiting but rather merely illustrative. Any module could employ any analysis technique commonly known in the art, or described herein, to accomplish any of the analyses described herein.

Biological applications

[00105] As described herein above, the systems and methods of the invention can be used to align and assemble any kind of sequence information, and particularly, any kind of biological sequence information. In some embodiments, the invention can effectively align any kind of nucleic acid sequences, including but not limited to coding or non-coding RNA, double or single stranded RNA, mRNA, tRNA, rRNA, micro-RNA, siRNA, genomic DNA, cDNA, double or single-stranded DNA, as well as proteins and amino acids, from any kind of organism, including

but not limited to animals, plants, fungi, yeast, mammals, including humans, non-mammals, eukaryotes, prokaryotes, bacteria, archaea, mitochondria, chloroplasts, viruses and virions.

[00106] There is no particular limit to the size of the sequence that can be compared, mapped, assembled and/or aligned by the methods and systems of the invention, and as such, in some embodiments, entire genes or segments of genes, entire genomes, entire chromosomes, or entire transcriptomes of an organism can be assembled, aligned and/or can be compared with other genes, segments of genes, entire genomes (comparative genomics), entire chromosomes, or entire transcriptomes, either of the same organism or different organisms. In some embodiments, the invention can be used in the analysis of gene fusion, whether the result of translocation, interstitial deletion, chromosomal inversion, or other means. The detection of fusion genes or the chromosomal aberrations, e.g., mutations, resulting therefrom, by the systems and methods of the invention can be used in the diagnosis of certain cancers or cellular pathologies, diseases, or conditions, for example. In other embodiments, the systems and methods of the invention can detect fusion genes created specifically for research applications, such as in fluorescence microscopy. The fusion proteins and amino acids resulting or encoded from the fusion genes can also be aligned by the systems and methods of the invention. The systems and methods of the invention can also be used in assessing genome rearrangement; genotyping; gene expression; personal genomics; genome-wide variation; gene mutation or aberrations, including deletions, additions, or duplication; identification of protein binding sites; the study of genome-wide methylation patterns; and the assembly of new genomes or transcriptomes.

[00107] In some preferred embodiments, the entire transcriptome of an organism can be aligned and assembled by the invention. In some embodiments, the invention can detect and identify the presence of and location of introns and exons within the transcriptome of an organism, including the junctions between introns and exons, between introns and introns, and between exons and exons, as well as splice junctions, and can identify splicing and alternative splicing events. The systems and methods of the invention can also detect and identify known or predicted combinations of exons, and unexpected exon pairs that occur through exon skipping, cryptic splicing, gene fusions, or by any other means. Boundaries between exons or introns, or combinations thereof need not be known in order for the systems and methods of the invention to function and be practiced. Through analysis and alignment of an organism's transcriptome, the systems and methods of the invention can determine which specific genes are being actively

expressed (or not expressed) at a given time, in a given tissue or in the specific cells from which a sample is obtained. In addition, such transcriptome analysis by the systems and methods of the invention can identify novel transcripts and junctions between exons and introns, as well as estimate the abundance of the transcripts that are identified and aligned (e.g., quantitative analysis of the transcriptome).

[00108] In other embodiments, the invention can detect the presence and location of mutations, polymorphisms, allelic polymorphisms, and single nucleotide polymorphisms.

[00109] In some embodiments, the systems and methods of the invention can be used in histology and pathology applications. For example, the systems and methods of the invention can be used to identify and align sequence information from formalin-fixed, paraffin-embedded cancer tissues or other diseased tissues, such as tumors, which samples often contain highly degraded and fragmented RNA. Samples for analysis may be obtained from a subject's body fluids, cells or tissues, such as from blood, urine, saliva, swab, tissue biopsy, tumor biopsy, or autopsy material.

[00110] In some embodiments, the systems and methods of the invention can be used in the diagnosis of certain diseases. For example, the systems and methods of the invention can align nucleic acid sequences such that a sequences from a subject or test organism can be compared with a sequence from a control or normal organism, and mutations, aberrations, polymorphisms, or single nucleotide polymorphisms (SNPs) can be detected and identified, which may be related to or associated with certain disease states. Nucleic acids for diagnosis may be obtained from a subject's body fluids, cells or tissues, such as from blood, urine, saliva, swab, tissue biopsy, tumor biopsy, or autopsy material. The genetic material, or genomic DNA, may be used directly for detection, or it may be amplified enzymatically by using PCR, or other amplification techniques, prior to analysis. RNA or cDNA, from any type of sample may also be used in similar fashion. The presence or absence of mutations, aberrations, polymorphisms, or SNPs in a sample can provide information to a subject regarding a disease state, or predisposition to developing a disease, or whether the subject may be a carrier of a particular genetic abnormality, which could have direct implications for the subject. For example, the information provided from the practice of the systems and methods of the invention can inform a medical care provider

and/or a subject about a particular treatment, therapy, drug effect, or a life decision, e.g., child bearing.

[00111] For purposes of the invention, sequence information may be obtained from any biological samples obtained from a subject, such as, for example, a subject's bodily fluids, cells, or tissues, such as from blood, umbilical cord blood, urine, saliva, swab, tissue biopsy, tumor biopsy, amniocentesis samples, or autopsy material. Nucleic acids can be extracted from such biological samples by any methods known in the art.

[00112] The samples can be processed by any available sequencing instruments known in the art, regardless of the form in which the data are outputted to a user. For example, in analyses relating to the transcriptome, total RNA may be extracted by any means known in the art, to prepare a cDNA library. The cDNA library can then be sequenced by any sequencing machines in the art, the output of which is then utilized by the systems and methods of the invention, for alignment and assembly of the nucleic acid sequences. Sequencing instruments for nucleic acids can include, for example, any of the high throughput sequencing machines from 454 Life Sciences/Roche, Illumina/Solexa, Applied Biosystems/Life Technologies (SOLiD), Helicos Biosciences, Complete Genomics, and Ion Torrent Systems (which generate "next generation sequencing" or "NGS" data), as well as the more traditional machines such as the Sanger sequencing machines.

[00113] The systems and methods of the invention can utilize the sequencing output based on any methodology, such as those employed by available sequencing instruments, and may include single molecule real-time sequencing, ion semiconductor sequencing, pyrosequencing, sequencing by synthesis, sequencing by ligation (SOLID sequencing), polony sequencing, direct RNA sequencing, sequencing technology based on reversible dye-terminators, DNA nanoball sequencing, Heliscope single molecule sequencing, chain termination sequencing, nanopore DNA sequencing, sequencing by hybridization, sequencing with mass spectrometry, microfluidic Sanger sequencing, and transmission electron microscopy DNA sequencing. The systems and methods of the invention are amenable to modifications or variations of the foregoing nucleic acid sequencing methodologies.

[00114] The systems and methods of the invention can utilize the resulting sequence data from any sequencing instruments, including but not limited to, those described herein, and are

able to utilize the resulting sequence data outputted from any available methodology that can sequence nucleic acids, whether the output is from more traditional methodologies such as Sanger sequencing, which yields relatively large read outputs, for example, in the range of about 400 to about 900 base pairs, or whether the output is in the form of smaller reads, such as the output from NGS technologies. The output from NGS sequencers often consist of millions of short sequences, or reads, that are collected randomly from the target genome or transciptome, for example. In some embodiments, the sequences that are outputted by the NGS sequencers and utilized by the systems and methods of the invention are between about 15 base pairs and about 500 base pairs in length. In particular embodiments, sequence outputs of 75 or 100 nucleotides are provided and used. However, the systems and methods of the invention are not limited by the length of these reads, and are capable of aligning sequences regardless of the particular sequence read length.

[00115] In one embodiment in which transcriptome assembly is desired, samples are prepared such that they are suitable for the particular sequencing instruments and methods chosen. Samples may be obtained from an organism (e.g., from blood, serum, umbilical cord blood, urine, saliva, swab, tissue biopsy, tumor biopsy, amniocentesis samples, or autopsy material), and RNA (mRNA or total RNA) is extracted from the samples by any suitable nucleic acid extraction / isolation technique or means known in the art. DNA contamination is then removed from the samples, and the remaining RNA is broken up into short fragments, for example, using divalent cations under elevated temperature. The cleaved RNA fragments are then reverse transcribed into cDNA, sequencing adaptors are ligated, and fragment size selection takes place.

[00116] The ends of the cDNAs are then sequenced by any available sequencing technology, such as those described herein. If both ends of the cDNAs are sequenced, then paired-end reads are generated. Regardless of the type and length of reads generated by these processes, the systems and methods of the invention can utilize reads generated by any process or system for at least any of the purposes described herein.

Examples

[00117] Throughout this disclosure, examples of DCMOM are presented. These examples are intended only to illustrate the principles of the invention and do not limit it. Where examples are provided, the example may not be repeated for each of the items in a list or category because it

will be apparent to a person of ordinary skill in the art that the description of the set, list, or category provided, in combination with the illustrative example, describes in sufficient detail, distinctly points out, and enables the person of ordinary skill in the art to practice the invention disclosed herein by generalizing any provided illustrative example across the set of similar items to which the example applies. Indeed, even where no example is provided, such a person of ordinary skill in the art will be able to read this written description and know how to predictably practice any of the multiple aspects or embodiments of this invention. Likewise, the examples presented herein expand on, and are expanded upon by, the descriptions elsewhere in this disclosure and description of the invention.

[0100] A test of the DCMOM system on biological data was conducted. 69 million paired-end reads were generated, each 100 base pairs ("bp") in length, from a pool of human adult brain samples using the HiSeq 2000 (Illumina). A read alignment was performed using several alignment systems to align the same reads to a reference genome. The alignment systems tested were: DCMOM, TopHat without annotation, TopHat with annotation, RUM (with annotation) and GSNAP (with annotation). **FIGs. 8-13** depict the results of these tests.

[0101] In the result of one test, **FIG. 8** depicts the output of DCMOM mapping four reads against a reference genome known as TNNT3. The four reads (read1 – read4) labeled in red at the top of **FIG. 8** were selected from the DCMOM results mapped to TNNT3. Here, read1 covers 6 exons, read2 and read3 cover 5 exons, and read4 covers 3 exons. For reference, the UCSC annotation track is shown in blue, even though it was not used by DCMOM. It is apparent from the figure that read1 came from the second isoform from the bottom, while read2 came from the isoform at the bottom. This result shows that DCMOM provides a novel way to align reads across multiple exons.

[0102] In another test of DCMOM's capabilities, **FIG. 9** displays the output from each of the tested read alignment systems, specifically showing mappings against the reference sequence named ABI1. The results were displayed graphically using the IGV display program. The five result tracks, starting at the top, are: DCMOM, TopHat without annotation, TopHat with annotation, RUM, and GSNAP. On the bottom is the HG19 Ref-Seq. Except for the Ref-seq track, each track is displayed with two sub tracks, a coverage track and an alignment track. In the alignment track, a small block in gray indicates a read or its segment, and the blue line

between these blocks is a junction. The figure shows that for DCMOM, and for those other tools with annotation results, there are many reads that span three exons, including the short exon. However, TopHat without annotation did not recognize the exon. This result shows that DCMOM, without annotation, provides a previously unknown way to discover novel short exons.

[0103] In an additional test of DCMOM's capabilities, **FIG. 10** depicts the output of DCMOM's mapping against a portion of the 5' UTR of HEXIM 1, using *Integrated Genome Viewer* to display both data from the *UCSC Gene Browser* as well as results from DCMOM. This location has previously been reported to include some variants such as mixed insertion (rs7216041), deletions (rs67016906 and rs7216036) and a SNP (rs116601299) according to the annotated reference sequence dbSNP 135. The DCMOM results **FIG. 10** (bottom) indicate insertions that exactly match with rs71806821, rs72506984 and rs11430584. Thus, DCMOM is able to duplicate the results of other systems, e.g., the systems used to originally record the variants.

[0104] **FIG. 11** depicts another test of DCMOM's capabilities, namely results from DCMOM using *Integrated Genome Viewer* to display both data from the *UCSC Gene Browser* as well as results from DCMOM. Here, the DCMOM results show the identification of a previously unknown variant consisting of a TT insertion in the NBL1 reference sequence. Thus, DCMOM is able to identify novel variants undetected by other systems.

[0105] Separately, to examine the relative coverage performance of the systems, BED Tools was used to generate exon coverage statistics for reads aligned within the genomic regions annotated as exons in Ref-seq. First, the breadth of coverage of exons are compared with other methods in the range of depth of coverage up to 100, as displayed in **FIG. 12** and **FIG. 13**.

[0106] These results show that, below a depth of coverage of 19, Tophat (with or without annotation) has a higher breadth of coverage than other systems, including DCMOM. However, in regions with a higher depth of coverage, DCMOM has a higher breadth of coverage than TopHat (without annotation) and has similar performance to RUM, GSNAP, and TopHat (with annotation). The regions identified by DCMOM tend to have high depth of coverage, implying that the alignment is reliable and consistent. As shown in **FIG. 13**, DCMOM was also found to have the lowest breadth of coverage at zero depth of coverage, which means that DCMOM has

the smallest proportion of uncovered annotated regions among the methods tested. It will be apparent from **FIG. 13** that DCMOM, when used with a plant or animal genome reference sequence, may have a sequence alignment result of between 16% and 17% zero coverage.

[0107] These results demonstrate that DCMOM is superior to TopHat (without annotation). In addition, DCMOM's overall performance compares favorably to RUM, GSNAP or TopHat with annotation; however, DCMOM does not require the annotation that these other methods require. Not requiring annotation is advantageous at least because systems that require annotation are limited to identifying already discovered exons, junctions and variants. Thus, because DCMOM – a system that does not use annotation – has comparable performance to those systems that do use annotation, it is apparent that DCMOM provides a way to discover novel exons, junctions and variants, especially mini exons, previously unreported or marked in annotations.

[0108] Additionally, the runtime performance of DCMOM was measured on a Linux system with 80 processors and 1TB memory. Forty (40) processors and 50GB memory were each allocated to each run with 69 million sequence reads. DCMOM, implemented as a multi-threaded java program, took 5 hours to generate a SAM file and additional 40 minutes to create the BAM file.

[0109] Certain modifications and improvements will occur to those skilled in the art upon a reading of the foregoing description. It should be understood that all such modifications and improvements have been deleted herein for the sake of conciseness and readability but are properly within the scope of the following claims.

What is claimed is:

1. An electronic apparatus, the apparatus comprising:

a CPU in communication with a memory; and

a plurality of machine readable storage media, in communication with the CPU, comprising software modules for determining match information corresponding to a first query sequence and a first reference sequence, said determining comprising:

a) selecting a first root node subsequence, having a first seed length greater than or equal to 9 elements, from the first query sequence;

b) determining a location of the first root node subsequence in the first reference sequence;

c) identifying a maximally extended subsequence match including the first root node subsequence and any sequence elements adjacent to the first root node subsequence that are present in both the first query sequence and the first reference sequence;

d) identifying an unmatched sequence element not present in both the first query sequence and the first reference sequence, adjacent to the maximally extended subsequence match; and

e) repeating steps a-d with a second query sequence and a second reference sequence, wherein the second query sequence comprises at least a portion of the first query sequence other than the maximally extended subsequence match, and wherein the second reference sequence comprises at least a portion of the first reference sequence other than the maximally extended subsequence match, and

f) storing the match information into at least one of the plurality of machine readable storage media.

2. The apparatus of claim 1, wherein the second query sequence comprises at least a portion of the first query sequence to the right of the maximally extended subsequence match.

3. The apparatus of claim 1, wherein the second reference sequence comprises at least a portion of the first reference sequence to the right of the maximally extended subsequence match.

4. The apparatus of claim 1, wherein the number of elements in the second reference sequence is less than 30,000 elements.

5. The apparatus of claim 1, wherein step a further comprises calculating a hash value using a first seed length portion of the first root node subsequence, and wherein step b further comprises generating a hash value for each of a plurality of first seed length portions of the reference sequence.

6. The apparatus of claim 5, wherein the seed length used to generate the hash value for a portion of the second query sequence is shorter than the first seed length used to generate the hash value for a portion of the first query sequence.

7. The apparatus of claim 5, wherein the first seed length is at least 9 elements.

8. The apparatus of claim 5, wherein the first seed length is between 9 to 25 elements.

9. The apparatus of claim 1, wherein the first query sequence represents a nucleotide sequence.

10. The apparatus of claim 9, wherein the determining further comprises:

determining intron information by identifying at least 30 elements between two adjacent root nodes mapped to the first reference sequence; and

storing the intron information onto a machine readable storage medium.

11. The apparatus of claim 9, wherein the determining further comprises:

determining exon information by identifying at least 2 adjacent elements mapped to the first reference sequence as a node; and

storing the exon information onto a machine readable storage medium.

12. The apparatus of claim 9, wherein the determining further comprises:

determining single nucleotide polymorphism information by identifying one element in the query sequence which are not identical to elements in the reference sequence; and

storing the single nucleotide polymorphism information onto a machine readable storage medium.

13. The apparatus of claim 9, wherein the first query sequence further represents a single end read.

14. The apparatus of claim 9, wherein the first query sequence further represents a paired end read.

15. The apparatus of claim 9, wherein the first query sequence further represents a uniform length read.

16. The apparatus of claim 9, wherein the first query sequence further represents a variable length read.

17. The apparatus of claim 1, wherein the first query sequence is a nucleotide sequence generated from mRNA.

18. The apparatus of claim 1, wherein the first query sequence is a read output from a DNA sequencer.

19. The apparatus of claim 1, further comprising a DNA sequencing device in communication with the CPU, for sequencing genetic information from a sample to obtain the first query sequence.

20. The apparatus of claim 1, wherein the first reference sequence represents at least a portion of the human genome.

21. The apparatus of claim 1, wherein the first reference sequence is not annotated.

22. The apparatus of claim 1, wherein determining further comprises additional recursive mapping steps until all elements of the query sequence are mapped to the first reference sequence or determined not to match the first reference sequence.

23. The apparatus of claim 1, wherein the determining further comprises merging the first and second root node subsequences by classifying a joint between them, and storing classification information into a machine readable storage medium.

24. The apparatus of claim 1, wherein the determining further comprises merging a plurality of nodes by starting with the two leftmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at, then continuing with the two rightmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at.

25. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to a number of unmatched elements in the query sequence, and the classification information includes information indicating a substitution.

26. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and there is an unmatched element between the nodes, and wherein the classification information includes information indicating an insertion.

27. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and the two nodes overlap, and wherein the classification information includes information indicating an insertion.

28. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion or intron.

29. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero but less than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion.

30. The apparatus of claim 23, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a intron.

31. The apparatus of claim 1, wherein the determining further comprises recursively repeating steps a-d with additional query sequences until the remaining unmatched portion of the first query sequence is smaller than 3 elements.

32. A method for determining match information corresponding to a first query sequence and a first reference sequence, said method comprising:

a) selecting, by a CPU, a first root node subsequence, having a first seed length, from the first query sequence;

b) determining, by the CPU, a location of the first root node subsequence in the first reference sequence;

c) identifying, by the CPU, a maximally extended subsequence match including the first root node subsequence and any sequence elements adjacent to the first root node subsequence that are present in both the first query sequence and the first reference sequence;

d) identifying, by the CPU, an unmatched sequence element not present in both the first query sequence and the first reference sequence, adjacent to the maximally extended subsequence match; and

e) repeating steps a-d with a second query sequence and a second reference sequence, wherein the second query sequence comprises at least a portion of the first query sequence other than the maximally extended subsequence match, and wherein the second reference sequence comprises at least a portion of the first reference sequence other than the maximally extended subsequence match, and

f) storing, by the CPU, the match information into a machine readable storage medium.

33. The method of claim 32, wherein the second query sequence comprises at least a portion of the first query sequence to the right of the maximally extended subsequence match.

34. The method of claim 32, wherein the second reference sequence comprises at least a portion of the first reference sequence to the right of the maximally extended subsequence match.

35. The method of claim 32, wherein the number of elements in the second reference sequence is less than 30,000 elements.

36. The method of claim 32, wherein step a further comprises calculating a hash value using a first seed length portion of the first root node subsequence, and wherein step b further comprises generating a hash value for each of a plurality of first seed length portions of the reference sequence.

37. The method of claim 36, wherein the seed length used to generate the hash value for a portion of the second query sequence is shorter than the first seed length used to generate the hash value for a portion of the first query sequence.

38. The method of claim 36, wherein the first seed length is at least 9 elements.

39. The method of claim 36, wherein the first seed length is between 9 to 25 elements.

40. The method of claim 32, wherein the first query sequence represents a nucleotide sequence.

41. The method of claim 40, wherein the determining further comprises:

determining intron information by identifying at least 30 elements between two adjacent root nodes mapped to the first reference sequence; and

storing the intron information onto a machine readable storage medium.

42. The method of claim 40, wherein the determining further comprises:

determining exon information by identifying at least 2 adjacent elements mapped to the first reference sequence as a node; and

storing the exon information onto a machine readable storage medium.

43. The method of claim 40, wherein the determining further comprises:

determining single nucleotide polymorphism information by identifying one element in the query sequence which are not identical to elements in the reference sequence; and

storing the single nucleotide polymorphism information onto a machine readable storage medium.

44. The method of claim 40, wherein the first query sequence further represents a single end read.

45. The method of claim 40, wherein the first query sequence further represents a paired end read.

46. The method of claim 40, wherein the first query sequence further represents a uniform length read.

47. The method of claim 40, wherein the first query sequence further represents a variable length read.

48. The method of claim 32, wherein the first query sequence is a nucleotide sequence generated from mRNA.

49. The method of claim 32, wherein the first query sequence is a read output from a DNA sequencer.

50. The method of claim 32, further comprising a DNA sequencing device in communication with the CPU, for sequencing genetic information from a sample to obtain the first query sequence.

51. The method of claim 32, wherein the first reference sequence represents at least a portion of the human genome.

52. The method of claim 32, wherein the first reference sequence is not annotated.

53. The method of claim 32, wherein determining further comprises additional recursive mapping steps until all elements of the query sequence are mapped to the first reference sequence or determined not to match the first reference sequence.

54. The method of claim 32, wherein the determining further comprises merging the first and second root node subsequences by classifying a joint between them, and storing classification information into a machine readable storage medium.

55. The method of claim 32, wherein the determining further comprises merging a plurality of nodes by starting with the two leftmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at, then continuing with the two rightmost nodes of the plurality of nodes and recursively merging nodes until a root node is arrived at.

56. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to a number of unmatched elements in the query sequence, and the classification information includes information indicating a substitution.

57. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and there is an unmatched element between the nodes, and wherein the classification information includes information indicating an insertion.

58. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is equal to zero and the two nodes overlap, and wherein the classification information includes information indicating an insertion.

59. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion or intron.

60. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than zero but less than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a deletion.

61. The method of claim 54, wherein the distance between the location of the first and second root node subsequences on the reference sequence is greater than 30 elements and there are no unmatched elements between the two nodes, and wherein the classification information includes information indicating a intron.

62. The method of claim 32, wherein the determining further comprises recursively repeating steps a-d with additional query sequences until the remaining unmatched portion of the first query sequence is smaller than 3 elements.

63. The method of claim 32, further comprising:

wherein the reference sequence is at least a portion of a plant or animal genome, and the sequence alignment result has between 16% and 16.4% zero coverage.

64. A method for sequence alignment, the method comprising the step of using a CPU adapted to execute a recursive divide and conquer method to determine match information corresponding to a query sequence and a reference sequence, said match information including

the location where a plurality of elements of the query sequence are present in the reference sequence.

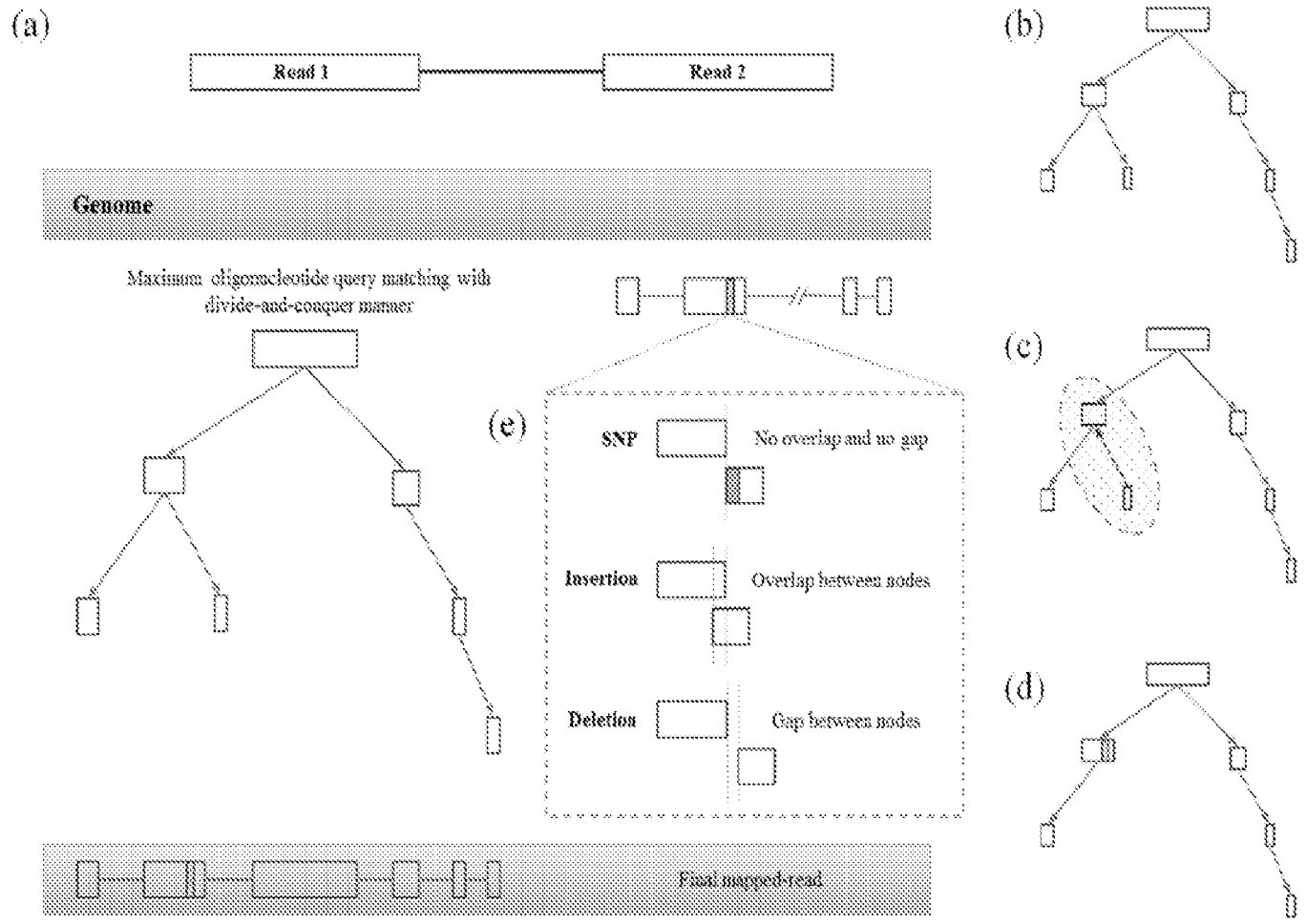
(a)
Read 1
Read 2
Genome
Maximum oligonucleotide query matching with divide-and-conquer manner
(e)
SNP
No overlap and no gap
Insertion
Overlap between nodes
Deletion
Gap between nodes
Final mapped read
(b)
(c)
(d)

Figure 1

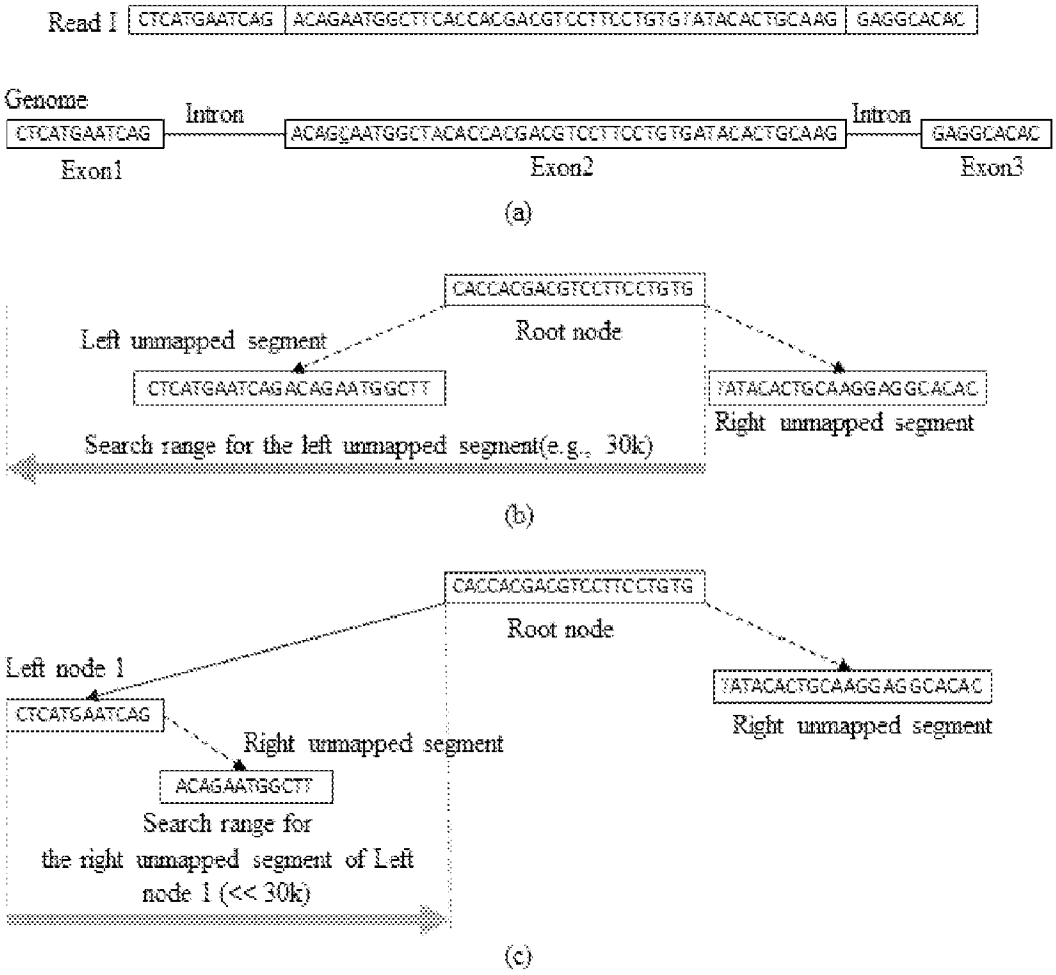
Read 1
CTCATGAATCAG
ACAGAATGGCTTCACCACGACGTCCTTCCTGTGTATACACTGCAAG
GAGGCACAC
Genome
Intron
Intron
CTCATGAATCAG
ACAGGAATGGCTACACCACGACGTCCTTCCTGTGATACACTGCAAG
GAGGCACAC
Exon1
Exon2
Exon3
(a)
CACCACGACGTCCTTCCTGTG
Root node
Left unmapped segment
CTCATGAATCAGACAGAATGGCTT
TATACACTGCAAGGAGGCACAC
Right unmapped segment
Search range for the left unmapped segment(e.g., 30k)
(b)
CACCACGACGTCCTTCCTGTG
Root node
Left node 1
CTCATGAATCAG
TATACACTGCAAGGAGGCACAC
Right unmapped segment
Right unmapped segment
ACAGAATGGCTT
Search range for
the right unmapped segment of Left
node 1 (<< 30k)
(c)

Figure 2

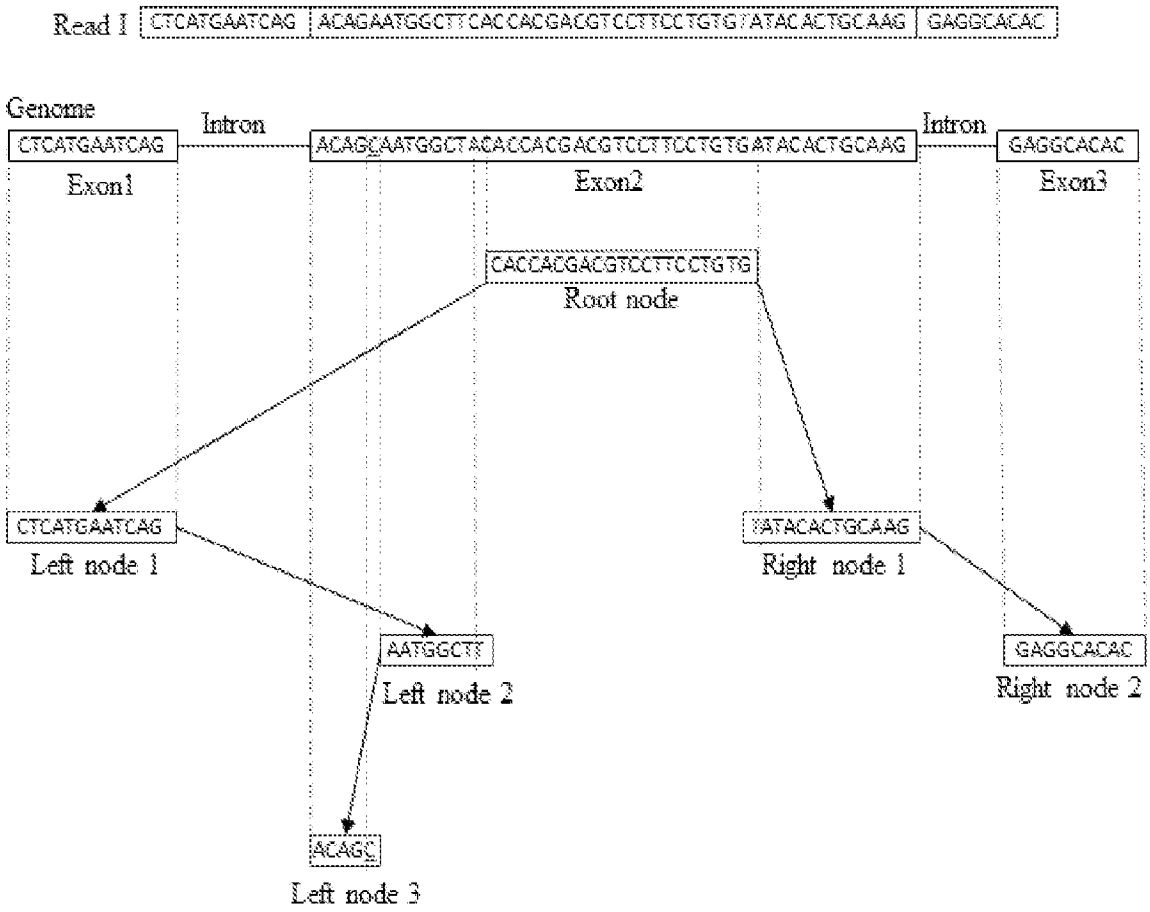
Read 1
CTCATGAATCAG
ACAGAATGGCTTCACCACGACGTCCTTCCTGTGTATACACTGCAAG
GAGGCACAC
Genome
Intron
Intron
CTCATGAATCAG
Exon1
ACAGCAATGGCTACACCACGACGTCCTTCCTGTGATACACTGCAAG
Exon2
GAGGCACAC
Exon3
CACCACGACGTCCTTCCTGTG
Root node
CTCATGAATCAG
Left node 1
ATACACTGCAAG
Right node 1
AATGGCTT
Left node 2
GAGGCACAC
Right node 2
ACAGC
Left node 3

Figure 3

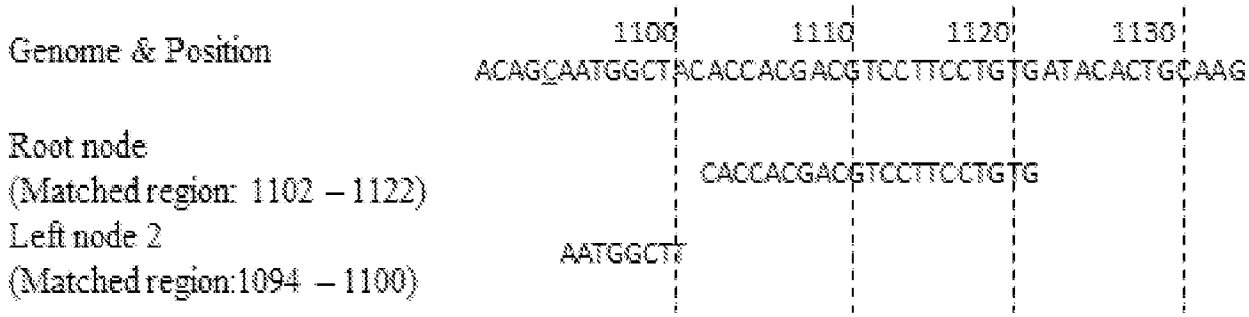
Genome & Position
1100
1110
1120
1130
ACAGCAATGGCTACACCACGACGTCCTTCCTGTGATACACTGCAAG
Root node
(Matched region: 1102 – 1122)
CACCACGACGTCCTTCCTGTG
Left node 2
(Matched region:1094 – 1100)
AATGGCT

Figure 4

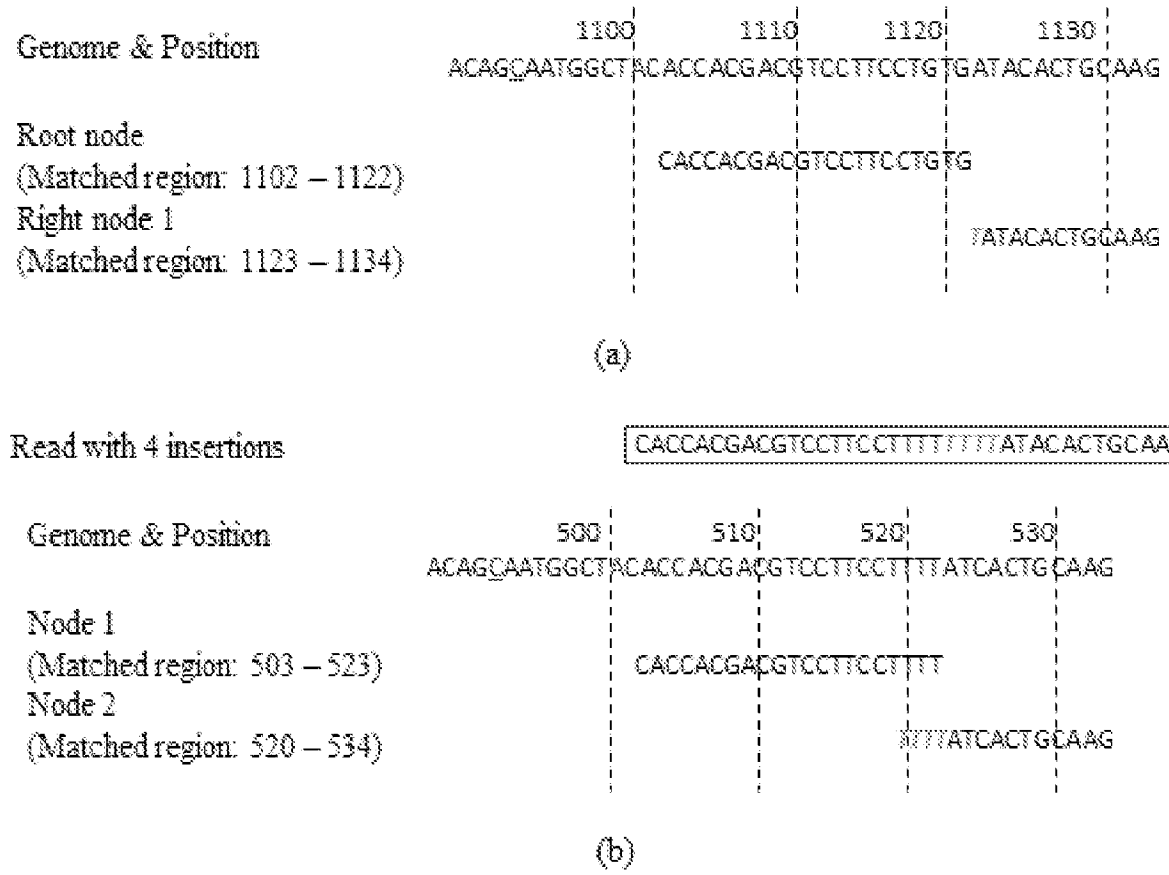
Genome & Position
1100
1110
1120
1130
ACAGCAATGGCTACACCACGACGTCCTTCCTGTGATACACTGCAAG
Root node
(Matched region: 1102 – 1122)
CACCACGACGTCCTTCCTGTG
Right node 1
(Matched region: 1123 – 1134)
ATACACTGCAAG
(a)
Read with 4 insertions
CACCACGACGTCCTTCCTTTTTTTTATACACTGCAAG
Genome & Position
500
510
520
530
ACAGCAATGGCTACACCACGACGTCCTTCCTTTTATCACTGCAAG
Node 1
(Matched region: 503 – 523)
CACCACGACGTCCTTCCTTTT
Node 2
(Matched region: 520 – 534)
TTTATCACTGCAAG
(b)

Figure 5

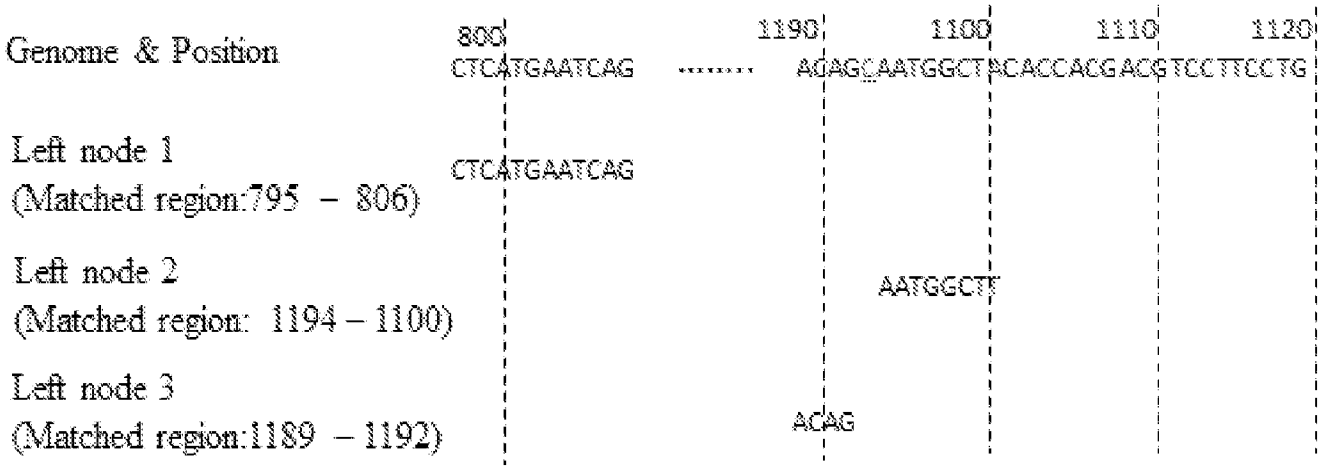
Genome & Position
800
1190
1100
1110
1120
CTCATGAATCAG
.....
ACAGCAATGGCTCCACCACGACGTCCTTCCTG
Left node 1
(Matched region:795 – 806)
CTCATGAATCAG
Left node 2
(Matched region: 1194 – 1100)
AATGGCTT
Left node 3
(Matched region:1189 – 1192)
ACAG

Figure 6


Read 1
CTCATGAATCAG
ACAGAATGGCTTCACCACGACGTCCTTCCTGTGTATACACTGCAAG
GAGGCACAC
Genome
Intron
Intron
CTCATGAATCAG
ACAGCAATGGCTACACCACGACGTCCTTCCTGTGATACACTGCAAG
GAGGCACAC
Exon1
Exon2
Exon3
ACAG-AATGGCTTCACCACGACGTCCTTCCTGTGATACACTGCAAG
Junction
Junction
Deletion: C
Substitution: T ≠ A
Insertion: T
CTCATGAATCAG
GAGGCACAC

Figure 7

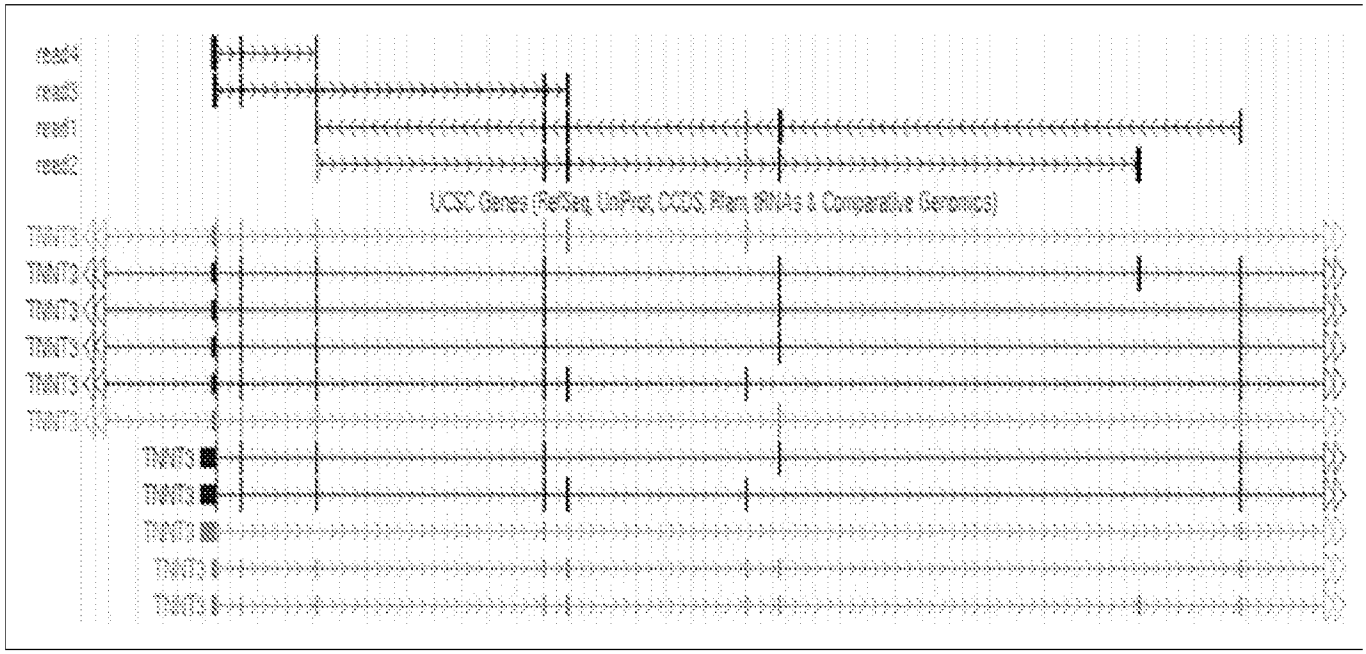

Figure 8

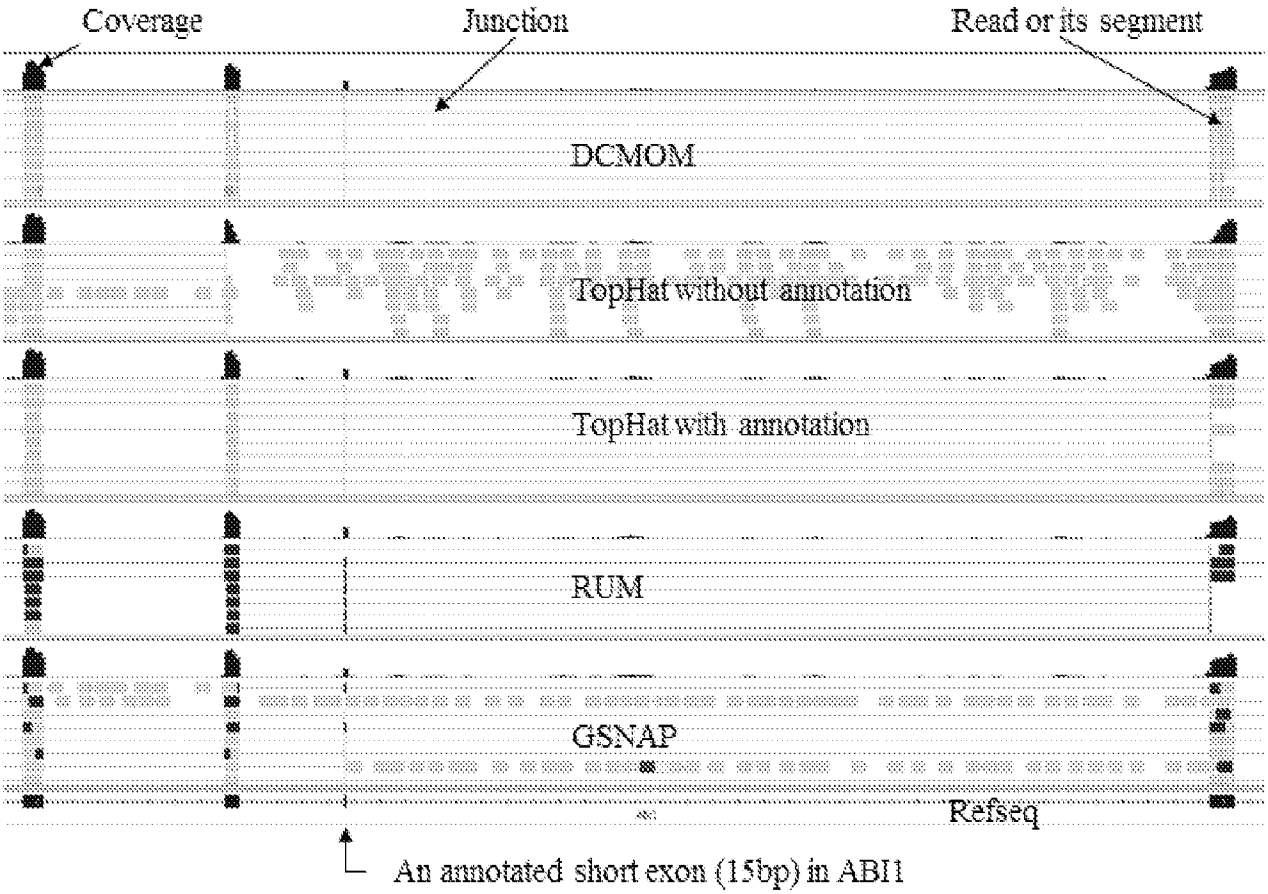
Coverage
Junction
Read or its segment
DCMOM
TopHat without annotation
TopHat with annotation
RUM
GSNAP
Refseq
An annotated short exon (15bp) in ABI1

Figure 9

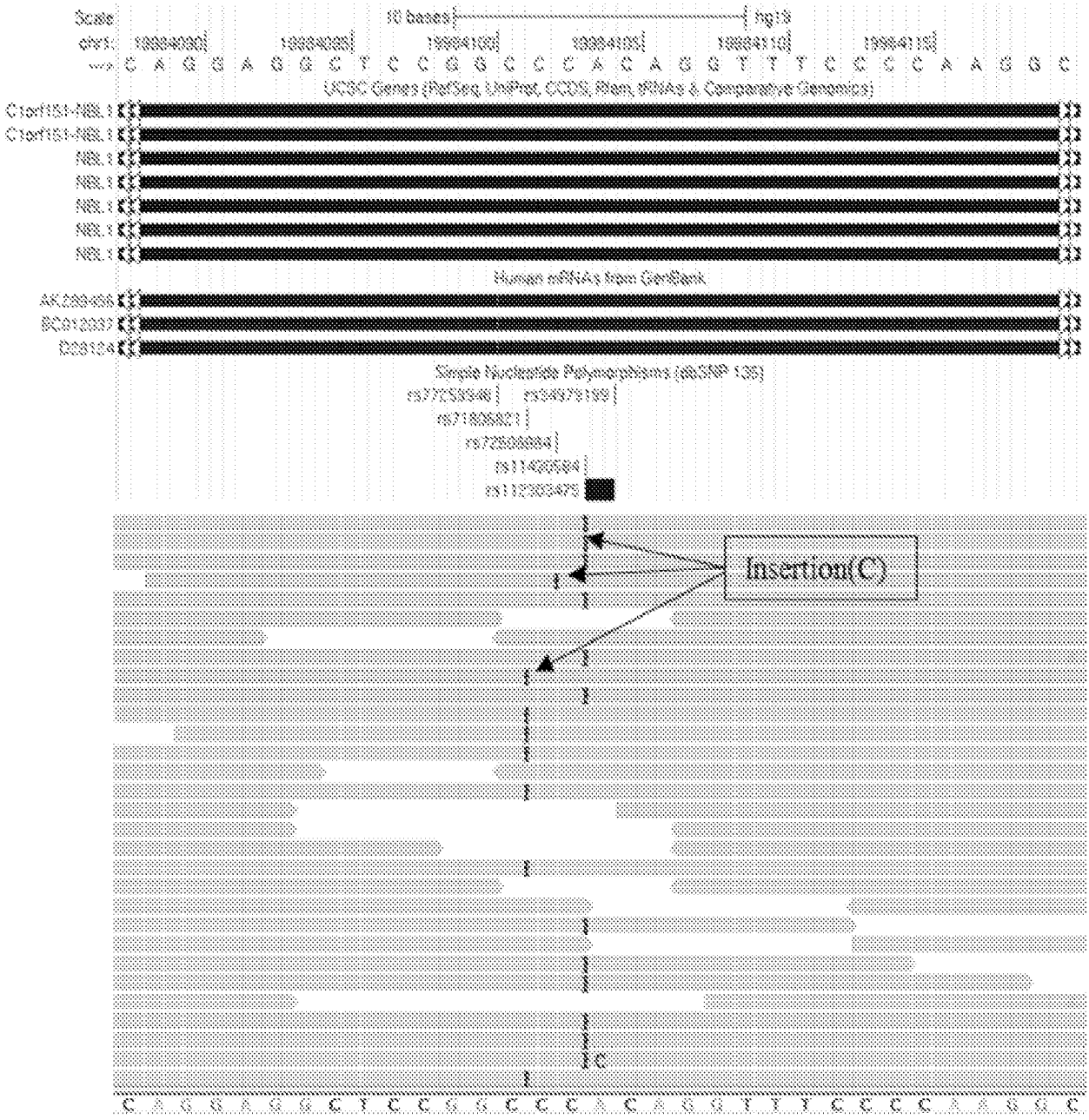

Figure 10

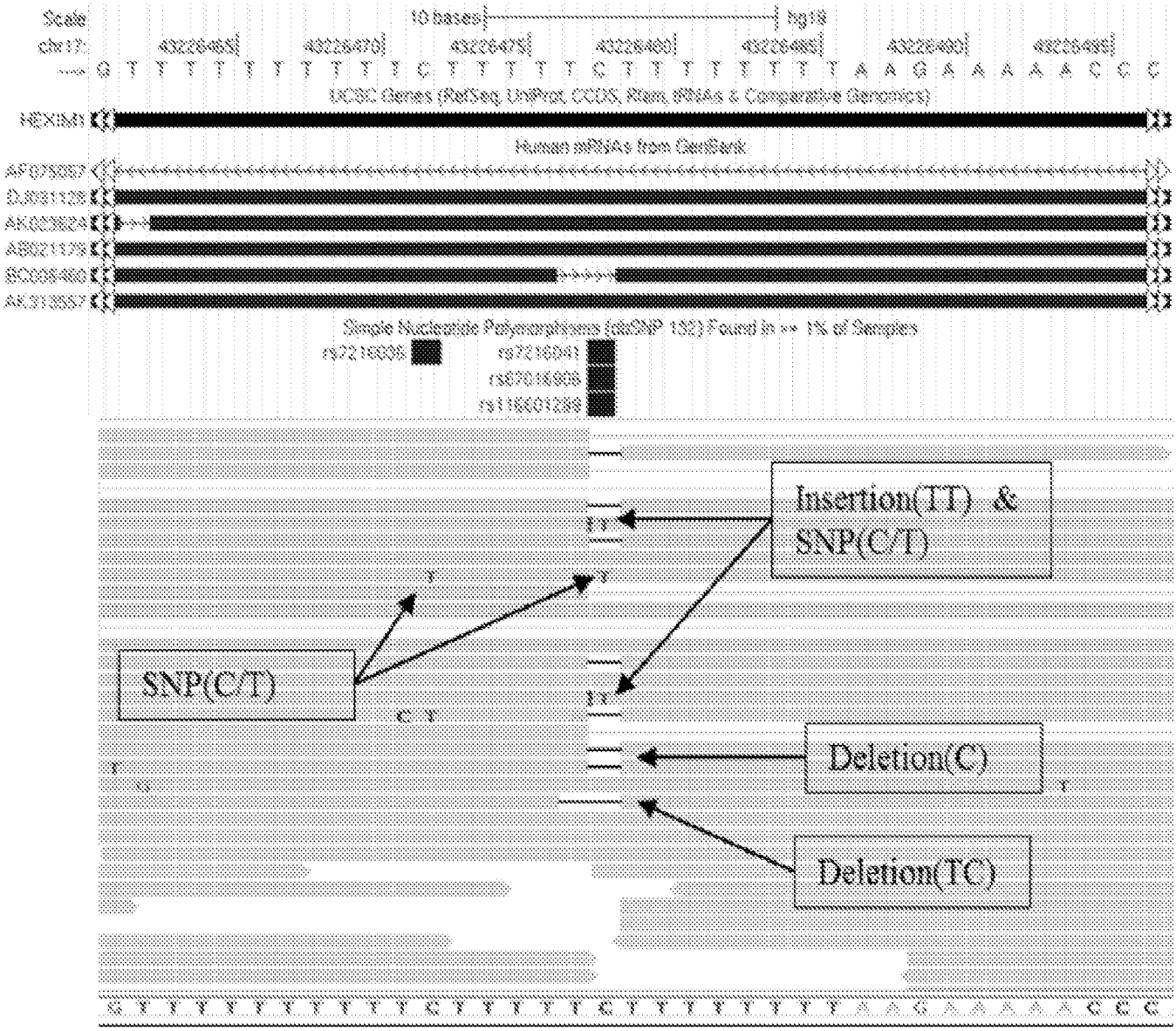

Figure 11

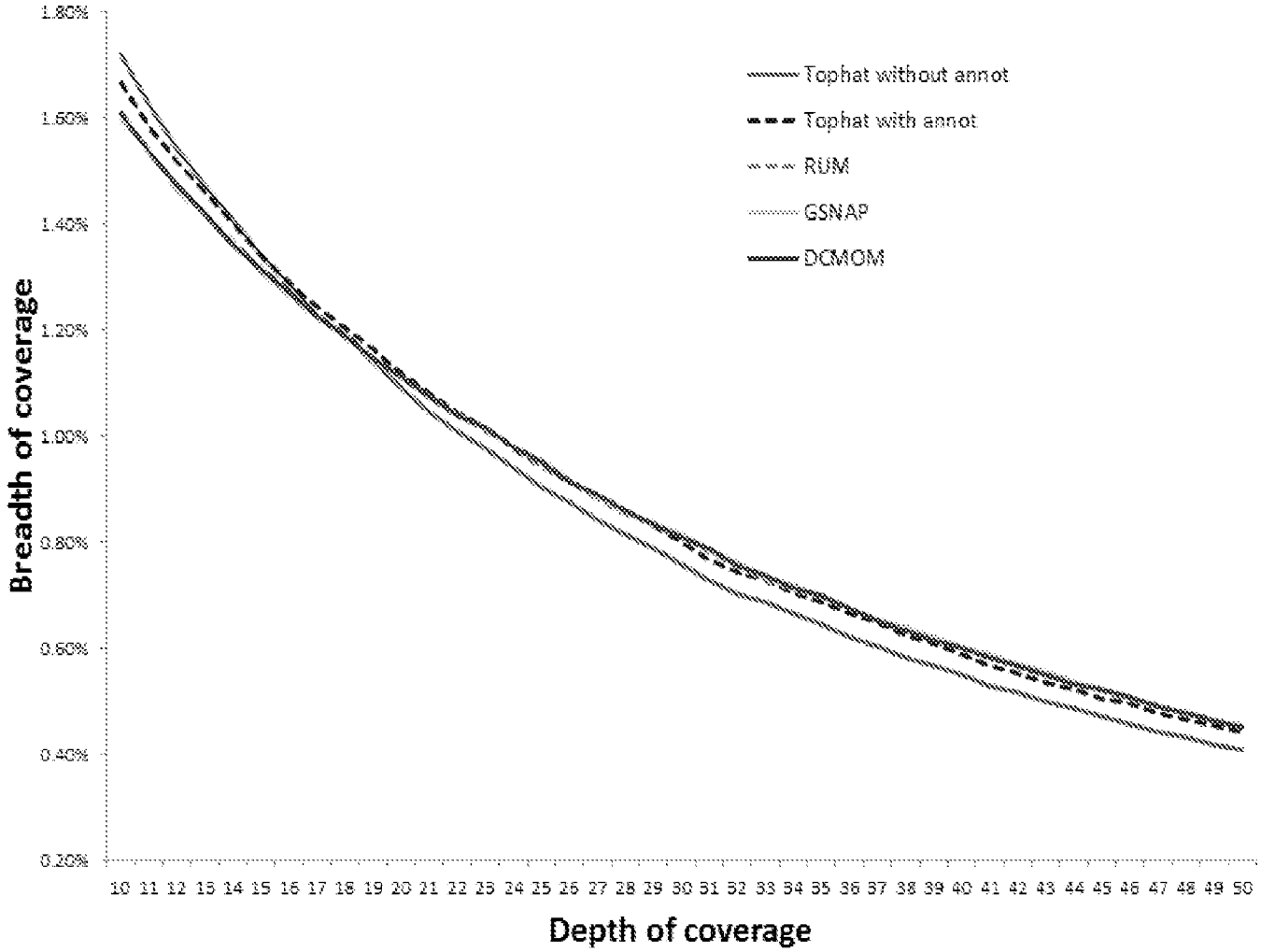
Tophat without annot
Tophat with annot
RUM
GSNAP
DCMOM
Breadth of coverage
1.80%
1.60%
1.40%
1.20%
1.00%
0.80%
0.60%
0.40%
0.20%
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Depth of coverage

Figure 12

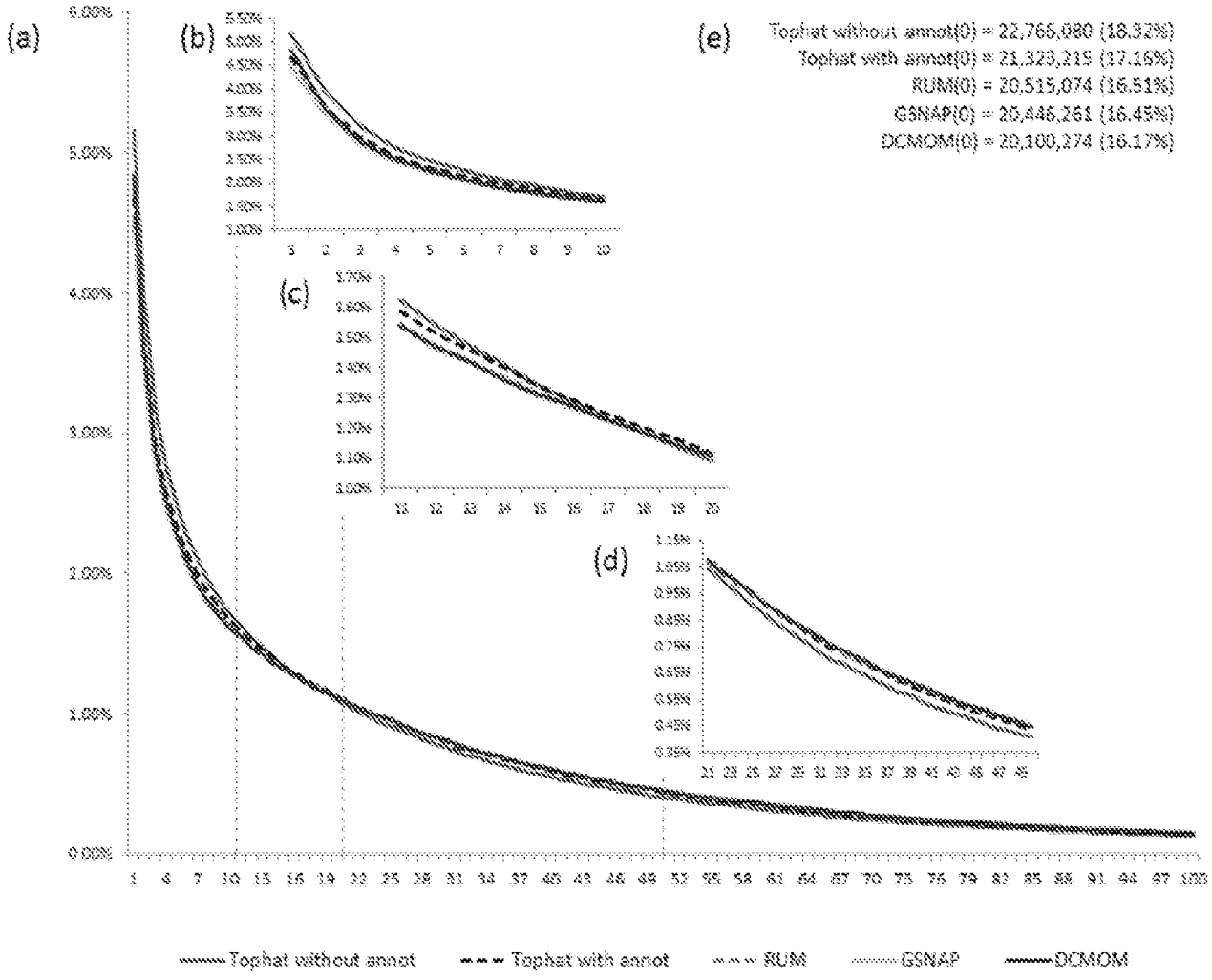
(a)
(b)
(c)
(d)
(e)
Tophat without annot(0) = 22,766,080 (18.32%)
Tophat with annot(0) = 21,323,215 (17.16%)
RUM(0) = 20,515,074 (16.51%)
GSNAP(0) = 20,446,261 (16.45%)
DCMOM(0) = 20,100,274 (16.17%)
Tophat without annot
Tophat with annot
RUM
GSNAP
DCMOM

Figure 13

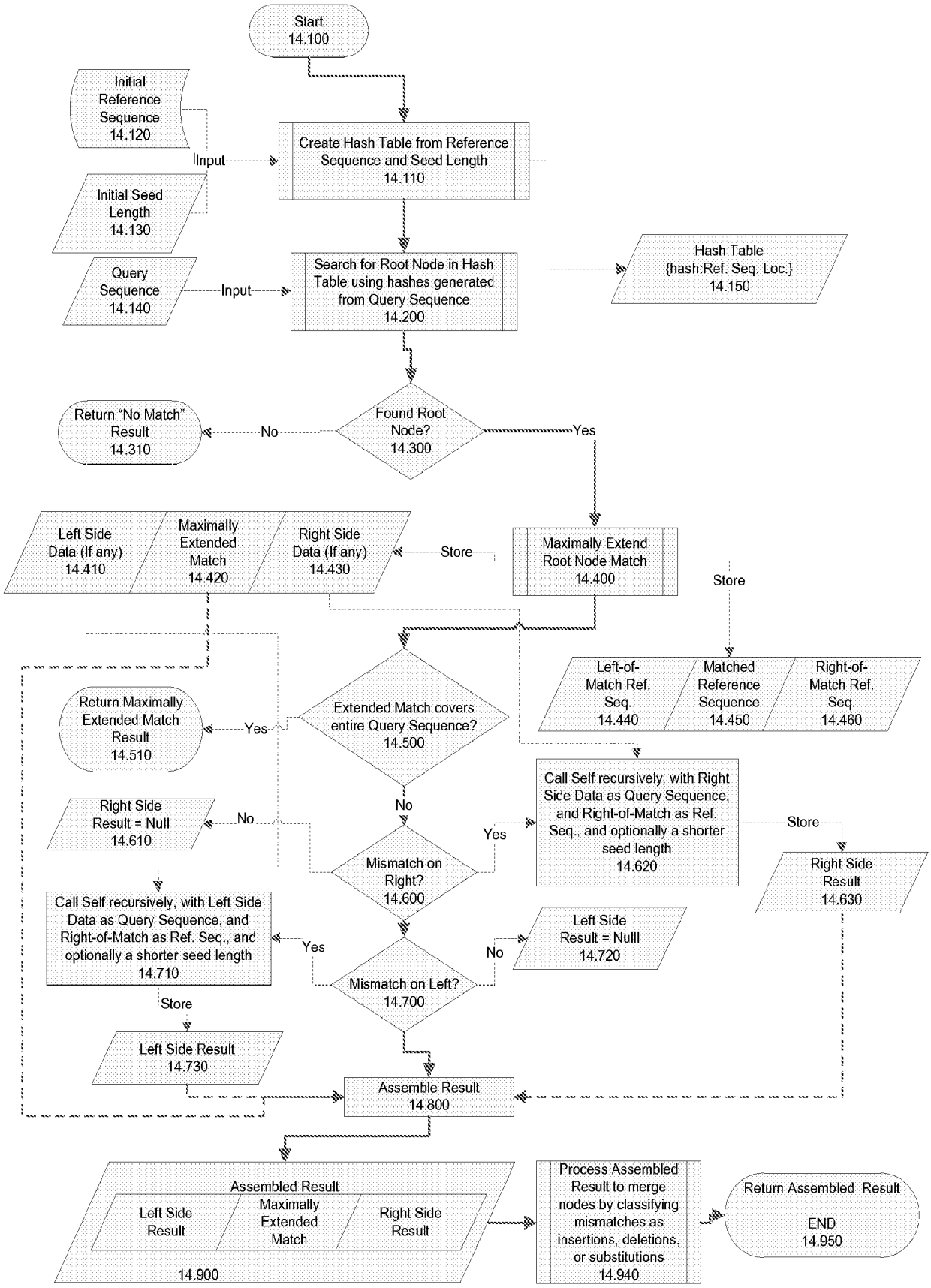
Start
14.100
Initial Reference Sequence
14.120
Input
Create Hash Table from Reference Sequence and Seed Length
14.110
Initial Seed Length
14.130
Hash Table
{hash:Ref. Seq. Loc.}
14.150
Query Sequence
14.140
Input
Search for Root Node in Hash Table using hashes generated from Query Sequence
14.200
Return "No Match" Result
14.310
No
Found Root Node?
14.300
Yes
Left Side Data (If any)
14.410
Maximally Extended Match
14.420
Right Side Data (If any)
14.430
Store
Maximally Extend Root Node Match
14.400
Store
Left-of-Match Ref. Seq.
14.440
Matched Reference Sequence
14.450
Right-of-Match Ref. Seq.
14.460
Return Maximally Extended Match Result
14.510
Yes
Extended Match covers entire Query Sequence?
14.500
No
Call Self recursively, with Right Side Data as Query Sequence, and Right-of-Match as Ref. Seq., and optionally a shorter seed length
14.620
Store
Right Side Result
14.630
Right Side Result = Null
14.610
No
Mismatch on Right?
14.600
Yes
Call Self recursively, with Left Side Data as Query Sequence, and Right-of-Match as Ref. Seq., and optionally a shorter seed length
14.710
Yes
Left Side Result = Null
14.720
No
Mismatch on Left?
14.700
Store
Left Side Result
14.730
Assemble Result
14.800
Assembled Result
Left Side Result
Maximally Extended Match
Right Side Result
14.900
Process Assembled Result to merge nodes by classifying mismatches as insertions, deletions, or substitutions
14.940
Return Assembled Result
END
14.950

Figure 14