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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 April 2018 (19.04.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/071055 A1

(51) International Patent Classification:
G06F 7/00 (2006.01)

(21) International Application Number:
PCT/US2017/017842

(22) International Filing Date:
14 February 2017 (14.02.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/EP2016/074311
11 October 2016 (11.10.2016) EP
PCT/EP2016/074307
11 October 2016 (11.10.2016) EP
PCT/EP2016/074301
11 October 2016 (11.10.2016) EP
PCT/EP2016/074297
11 October 2016 (11.10.2016) EP

(71) Applicant: **GENOMSYS SA** [CH/CH]; chemin de la Raye 13, 1024 Ecublens VD (CH).

(72) Inventor; and
(71) Applicant: **BALUCH, Mohamed Khoso** [US/US]; 4439 Woodsedge CT, Chantilly, Virginia 20151 (US).

(72) Inventors: **ZOIA, Giorgio**; Chemin des Croix-Rouges 10, 1007 Lausanne (CH). **RENZI, Daniele**; Route Aloys-Fauquez 105, 1018 Lausanne (CH).

(74) Agent: **ETTEHADIEH, Aslan**; Aslan Law, P.C., 7804 Holmes Run Drive, Falls Church, Virginia 22042 (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, DJ, 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, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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,

(54) Title: METHOD AND APPARATUS FOR THE COMPACT REPRESENTATION OF BIOINFORMATICS DATA

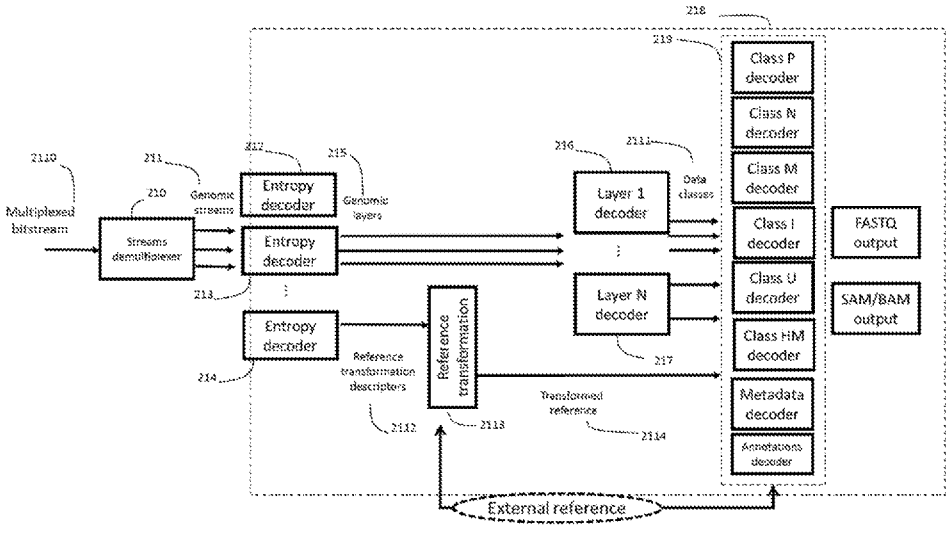


Figure 21

(57) Abstract: Method and apparatus for the compression of genome sequence data produced by genome sequencing machines. Sequence reads are coded by aligning them with respect to pre-existing or constructed reference sequences, the coding process is composed of a classification of the reads into data classes followed by the coding of each class in terms of a multiplicity of descriptors layers. Specific source models and entropy coders are used for each data class in which the data is partitioned, and each associated descriptor layer.

WO 2018/071055 A1

[Continued on next page]

GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, 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:

— *with international search report (Art. 21(3))*

METHOD AND APPARATUS FOR THE COMPACT REPRESENTATION OF BIOINFORMATICS DATA

TECHNICAL FIELD

[0001] This disclosure provides a novel method of representation of genome sequencing data which reduces the utilized storage space and improves access performance by providing new functionality that are not available with known prior art methods of representation.

BACKGROUND

[0002] An appropriate representation of genome sequencing data is fundamental to enable efficient genomic analysis applications such as genome variants calling and all other analysis performed with various purposes by processing the sequencing data and metadata.

Human genome sequencing has become affordable by the emergence of high-throughput low cost sequencing technologies. Such opportunity opens new perspectives in several fields ranging from the diagnosis and treatment of cancer to the identification of genetic illnesses, from pathogen surveillance for the identification of antibodies to the creation of new vaccines, drugs and the customization of personalized treatments.

Hospitals, genomic analysis providers, bioinformatics and large biological data storage centers are looking for affordable, fast, reliable and interconnected genomic information processing solutions which could enable scaling genomic medicine to a world-wide scale. Since one of the bottleneck in the sequencing process has become data storage, methods for representing genome sequencing data in a compressed form are increasingly investigated.

[0003] The most used genome information representations of sequencing data are based on zipping FASTQ and SAM formats. The objective is to compress the traditionally used file formats (respectively FASTQ and SAM for non-aligned and aligned data). Such files are constituted by plain text characters and are compressed, as mentioned above, by using general purpose approaches such as LZ (from Lempel and Ziv, the authors who published the first versions) schemes (the well-known zip, gzip etc). When general purpose compressors such as gzip are used, the result of compression is usually a single blob of binary data. The information in such monolithic form results quite difficult to archive, transfer and elaborate particularly when like in the case of high throughput sequencing the volume of data are extremely large. The BAM format is characterized by poor compression performance due to the focus on compression of the inefficient and redundant SAM format rather

than on extracting the actual genomic information conveyed by SAM files and due to the adoption of general purpose text compression algorithms such as gzip rather than exploiting the specific nature of each data source (the genomic data itself).

A more sophisticated approach to genomic data compression that is less used, but more efficient than BAM is CRAM. CRAM provides more efficient compression for the adoption of differential encoding with respect to a reference (it partially exploits the data source redundancy), but it still lacks features such as incremental updates, support for streaming and selective access to specific classes of compressed data.

[0004] These approaches generate poor compression ratios and data structures that are difficult to navigate and manipulate once compressed. Downstream analysis can be very slow due to the necessity of handling large and rigid data structures even to perform simple operation or to access selected regions of the genomic dataset. CRAM relies on the concept of the CRAM record. Each CRAM record represents a single mapped or unmapped reads by coding all the elements necessary to reconstruct it.

[0005] CRAM presents the following drawbacks and limitations that are solved and removed by the invention described in this document:

1. CRAM does not support data indexing and random access to data subsets sharing specific features. Data indexing is out of the scope of the specification (see section 12 of CRAM specification v 3.0) and it is implemented as a separate file. Conversely the approach of the invention described in this document employs a data indexing method that is integrated with the encoding process and indexes are embedded in the encoded (i.e. compressed) bit stream.
2. CRAM is built by core data blocks that can contain any type of mapped reads (perfectly matching reads, reads with substitutions only, reads with insertions or deletions (also referred to as "indels")). There is no notion of data classification and grouping of reads in classes according to the result of mapping with respect to a reference sequence. This means that all data need to be inspected even if only reads with specific features are searched. Such limitation is solved by the invention by classifying and partitioning data in classes before coding.
3. CRAM is based on the concept of encapsulating each read into a "CRAM record". This implies the need to inspect each complete "record" when reads characterized by specific biological features (e.g. reads with substitutions, but without "indels", or perfectly mapped reads) are searched.

Conversely, in the present invention there is the notion of data classes coded separately in separated information layers and there is no notion of record encapsulating each read. This enables

more efficient access to set of reads with specific biological characteristics (e.g. reads with substitutions, but without "indels", or perfectly mapped reads) without the need of decoding each (block of) read(s) to inspect its features.

4. In a CRAM record each record field is associated to a specific flag and each flag must always have the same meaning as there is no notion of context since each CRAM record can contain any different type of data. This coding mechanism introduces redundant information and prevents the usage of efficient context based entropy coding.

Instead in the present invention, there is no notion of flag denoting data because this is intrinsically defined by the information "layer" the data belongs to. This implies a largely reduced number of symbols to be used and a consequent reduction of the information source entropy which results into a more efficient compression. Such improvement is possible because the use of different "layers" enables the encoder to reuse the same symbol across each layer with different meanings according to the context. In CRAM each flag must always have the same meaning as there is no notion of contexts and each CRAM record can contain any type of data.

5. In CRAM substitutions, insertions and deletions are represented by using different syntax elements, option that increases the size of the information source alphabet and yields a higher source entropy. Conversely, the approach of the disclosed invention uses a single alphabet and encoding for substitutions, insertions and deletions. This makes the encoding and decoding process simpler and produces a lower entropy source model which coding yields bitstreams characterized by high compression performance.

[0006] The present invention aims at compressing genomic sequences by classifying and partitioning sequencing data so that the redundant information to be coded is minimized and features such as selective access and support for incremental updates are directly enabled in the compressed domain.

One of the aspects of the presented approach is the definition of classes of data and metadata structured in different layers and encoded separately. The more relevant improvements of such approach with respect to existing methods consist in:

1. the increase of compression performance due to the reduction of the information source entropy constituted by providing an efficient source model for each class of data or metadata;

2. the possibility of performing selective accesses to portions of the compressed data and metadata for any further processing purpose directly in the compressed domain;

3. the possibility to incrementally (i.e. without the need of decoding and re-encoding) update compressed data and metadata with new sequencing data and/or metadata and/or new analysis results associated to specific sets of sequencing reads.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows how the position of the mapped reads pairs are encoded in the "pos" layer as difference from the absolute position of the first mapped read.

Figure 2 shows how two reads in a pair may originate from the two DNA strands.

Figure 3 shows how the reverse complement of read 2 is encoded if strand 1 is used as reference.

Figure 4 shows the four possible combinations of reads composing a reads pair and the respective coding in the "rcomp" layer.

Figure 5 shows how to calculate the pairing distance in case of constant reads length for three read pairs.

Figure 6 show how the pairing errors encoded in the "pair" layer enable the decoder to reconstruct the correct read pairing using the encoded "MPPPD".

Figure 7 shows the encoding of a pairing distance when a read is mapped on a difference reference than its mate. In this case additional descriptors are added to the pairing distance. One is a signaling flag, the second is a reference identifier and then the pairing distance.

Figure 8 shows the encoding of "n type" mismatches in a "nmis" layer.

Figure 9 shows a mapped read pair which presents substitutions with respect to a reference sequence.

Figure 10 shows how to calculate the positions of substitutions either as absolute or differential values.

Figure 11 shows how to calculate the symbols encoding substitutions types when no IUPAC codes are used. The symbols represent the distance - in a circular substitution vector - between the molecule present in the read and the one present on the reference at that position.

Figure 12 shows how to encode the substitutions into the "snpt" layer.

Figure 13 shows how to calculate substitution codes when IUPAC ambiguity codes are used.

Figure 14 shows how the "snpt" layer is encoded when IUPAC codes are used.

Figure 15 shows how for reads of class I the substitution vector used is the same as for class M with the addition of special codes for insertions of the symbols A, C, G, T, N.

Figure 16 shows some examples of encoding of mismatches and indels in case of IUPAC ambiguity codes. The substitution vector is much longer in this case and therefore the possible calculated symbols are more than in the case of five symbols.

Figure 17 shows a different source model for mismatches and indels where each layer contains the position of the mismatches or inserts of a single type. In this case no symbols are encoded for the mismatch or indel type.

Figure 18 shows an example of mismatches and indels encoding. When no mismatches or indels of a given type are present for a read, a 0 is encoded in the corresponding layer. The 0 acts as reads separator and terminator in each layer.

Figure 19 shows how a modification in the reference sequence can transform M reads in P reads. This operation can reduce the information entropy of the data structure especially in case of high coverage data.

Figure 20 shows a genomic encoder 2010 according to one embodiment of this invention.

Figure 21 shows a genomic decoder 218 according to one embodiment of this invention.

Figure 22 shows how a "internal" reference can be constructed by clustering reads and assembling the segments taken from each cluster.

Figure 23 shows how a strategy of constructing a reference consists in storing the most recent reads once a specific sorting (e.g. lexicographic order) has been applied to the reads.

Figure 24 shows how a read belonging to the class of "unmapped" reads (Class U) can be coded using six descriptors stored or carried in the corresponding layers.

Figure 25 shows how an alternative coding of reads belonging to Class U where a signed pos descriptor is used to code the mapping position of a read on the constructed reference.

Figure 26 shows how reference transformations can be applied to remove mismatches from reads. In some cases reference transformations may generate new mismatches or change the type of mismatches found when referring to the reference before the transformation has been applied.

Figure 27 shows how reference transformations can change the class reads belong to when all or a subset of mismatches are removed (i.e. the read belonging to class M before transformation is assigned to Class P after the transformation of the reference has been applied).

Figure 28 shows how half mapped read pairs (class HM) can be used to fill unknown regions of a reference sequence by assembling longer contigs with unmapped reads.

Figure 29 shows how encoders of data of class N, M and I are configured with vectors of thresholds and generate separate subclasses of N, M and I data classes.

Figure 30 shows how all classes of data can use the same transformed reference for re-encoding or a different transformation can be used for each class N, M and I, or any combination thereof.

Figure 31 shows the structure of a Genomic Dataset Header.

Figure 32 shows the generic structure of a Master Index Table where each row contains genomic intervals of the several classes of data P, N, M, I, U, HM and further pointers to Metadata and annotations. The columns refer to specific positions on the reference sequences related to the encoded genomic data.

Figure 33 shows an example of one row of the MIT containing genomic intervals related to reads of Class P. Genomic regions related to different reference sequences are separated by a special flag ('S' in the example).

Figure 34 shows the generic structure of the Local Index Table (LIT) and how it is used to store pointers to the physical location of the encoded genomic information in the stored or transmitted data.

Figure 35 shows an example of LIT used to access Access Units no. 7 and 8 in the layer payload.

Figure 36 shows the functional relationship among the several rows of the MIT and the LIT contained in the genomic layers headers.

Figure 37 shows how an Access Unit is composed by several blocks of genomic data carried by different genomic streams containing data belonging to different classes. Each block is further composed by data packets used as data transmission units.

Figure 38 shows how Access Units are composed by a header and multiplexed blocks belonging to one or more layers of homogeneous data. Each block can be composed by one or more packets containing the actual descriptors of the genomic information.

SUMMARY

The features of the claims below solve the problem of existing prior art solutions by providing

A method for encoding genome sequence data, said genome sequence data comprising reads of sequences of nucleotides, said method comprising the steps of:

aligning said reads to one or more reference sequences thereby creating aligned reads,

classifying said aligned reads according to specified matching rules with said one or more reference sequences, thereby creating classes of aligned reads,

encoding said classified aligned reads as a multiplicity of layers of syntax elements,

wherein encoding said classified aligned reads as a multiplicity of layers of syntax elements comprises selecting said syntax elements according to said classes of aligned reads,

structuring said layers of syntax elements with header information thereby creating successive Access Units.

In another aspect the coding method further comprises further classifying said reads that do not satisfy said specified matching rules into a class of unmapped reads
constructing a set of reference sequences using at least some unmapped reads
aligning said class of unmapped reads to the set of constructed reference sequences
encoding said classified aligned reads as a multiplicity of layers of syntax elements,
encoding said set of constructed reference sequences
structuring said layers of syntax elements and said encoded reference sequences with header information thereby creating successive Access Units.

In another aspect the coding method further comprises identifying genomic reads without any mismatch in the reference sequence as first "Class P"

In another aspect the coding method further comprises identifying genomic reads as a second "Class N" when mismatches are only found in the positions where the sequencing machine was not able to call any "base" and the number of mismatches in each read does not exceed a given threshold.

In another aspect the coding method further comprises identifying genomic reads as a third "Class M" when mismatches are found in the positions where the sequencing machine was not able to call any "base", named "n type" mismatches, and/or it called a different "base" than the reference sequence, named "s type" mismatches, and the number of mismatches does not exceed given thresholds for the number of mismatches of "n type", of "s type" and a threshold obtained from a given function (f(n,s)) calculated on the number of "n type" and "s type" mismatches.

In another aspect the coding method further comprises identifying genomic reads as a fourth "Class I" when they can possibly have the same type of mismatches of "Class M", and in addition at least one mismatch of type: "insertion" ("i type") "deletion" ("d type") soft clips ("c type"), and wherein the number of mismatches for each type does not exceed the corresponding given threshold and a threshold provided by a given function (w(n,s,i,d,c)) calculated on the number of "n type", "s type", "i type", "d type" and "c type" mismatches.

In another aspect the coding method further comprises identifying genomic reads as a fifth "Class U" as comprising all reads that do not find any classification in the Classes P, N, M, I, as previously defined.

In another aspect the coding method further comprises that the reads of the genomic sequence to be encoded are paired

In another aspect the coding method further comprises that said classifying further comprises identifying genomic reads as a sixth "Class HM" as comprising all reads pairs where one read belong to Class P, N, M or I and the other read belong to "Class U"

In another aspect the coding method further comprises the steps of identifying if the two mate reads are classified in the same class (each of: P, N, M, I, U), then assigning the pair to the same identified class,
Identifying if the two mate reads are classified in different classes, and in case none of them belongs to the "Class U", then assigning the pair of reads to the class with the highest priority defined according to the following expression:
P < N < M < I
in which "Class P" has the lowest priority and "Class I" has the highest priority;
identifying if only one of the two mate reads has been classified as belonging to "Class U" and classifying
the pair of reads as belonging to the "Class HM" sequences

In another aspect the coding method further comprises that each class of reads N, M, I of reads N, M, I is further partitioned into two or more subclasses (296, 297, 298) according to a vector of thresholds (292, 293, 294) defined respectively for each class N, M and I, by the number of "n type" mismatches (292), the function f(n,s) (293) and the function w(n,s,i,d,c) (294).

identifying if the two mate reads are classified in the same subclass, then assigning the pair to the same sub-class
identifying if the two mate reads are classified into sub-classes of different Classes, then assigning the pair to the subclass belonging to the Class of higher priority according to the following expression:
N < M < I
where N has the lowest priority and I has the highest priority;

identifying if the two mate reads are classified in the same class, and such class is N or M or I, but in different sub-classes, then assigning the pair to the sub-class with the highest priority according to the following expressions:

$$N_1 < N_2 < \ldots < N_k$$

$$M_1 < M_2 < \ldots < M_j$$

$$I_1 < I_2 < \ldots < I_h$$

where the highest index has the highest priority.

In another aspect the coding method further comprises that said layers of syntax elements comprise a master index table, containing one section for each Class and sub-class of aligned reads, said section comprising the mapping positions on said one or more reference sequences of the first read of each Access Units of each Class or sub-class of data; jointly coding said master index table and said access unit data.

In another aspect the coding method further comprises that said layers of syntax elements further comprise information related to the type of reference used (pre-existing or constructed) and the segments of the read that do not match on the reference sequence.

In another aspect the coding method further comprises that said reference sequences are first transformed into different reference sequences by applying substitutions, insertions, deletions and clipping, then the encoding of said classified aligned reads as a multiplicity of layers of syntax elements refers to the transformed reference sequences.

In another aspect the coding method further comprises that the same transformation is applied to the reference sequences for all classes of data.

In another aspect the coding method further comprises that different transformations are applied to the reference sequences per each class of data.

In another aspect the coding method further comprises that the reference sequences transformations are encoded as layers of syntax elements and structured with header information thereby creating successive Access Units.

In another aspect the coding method further comprises that the encoding of said classified aligned reads and the related reference sequences transformations as multiplicity of layers of syntax elements comprises the step of associating a specific source model and a specific entropy coder to each descriptor layer.

In another aspect the coding method further comprises that said entropy coder is one of a context adaptive arithmetic coder, a variable length coder or a golomb coder

The present invention further provides a method for decoding encoded genomic data comprising the steps of:
parsing Access Units containing said encoded genomic data to extract multiple layers of syntax elements by employing header information
decoding said multiplicity of layers of syntax elements to extract aligned reads according to specific matching rules defining their classification with respect to one or more reference sequences

In another aspect the decoding method further comprises the decoding of unmapped genomic reads.

In another aspect the decoding method further comprises the decoding of classified genomic reads.

In another aspect the decoding method further comprises decoding a master index table containing one section for each class of reads and the associated relevant mapping positions

In another aspect the decoding method further comprises decoding information related to the type of reference used: pre-existing, transformed or constructed.

In another aspect the decoding method further comprises decoding information related to one or more transformations to be applied to the pre-existing reference sequences.

In another aspect the decoding method further comprises genomic reads that are paired

In another aspect the decoding method further comprises the case wherein said genomic data are entropy decoded

The present invention further provides a genomic encoder (2010) for the compression of genome sequence data 209, said genome sequence data 209 comprising reads of sequences of nucleotides,said genomic encoder (2010) comprising:
an aligner unit (201), configured to align said reads to one or more reference sequences thereby creating aligned reads,
a constructed-reference generator unit (202), configured to produce constructed reference sequences
a data classification unit (204), configured to classify said aligned reads according to specified matching rules with the one or more pre-existing reference sequences or constructed reference sequences thereby creating classes of aligned reads (208);
one or more layers encoding units (205-207), configured to encode said classified aligned reads as layers of syntax elements by selecting said syntax elements according to said classes of aligned reads.
a multiplexer (2016) for multiplexing the compressed genomic data and metadata.

In another aspect the genomic encoder further comprises
a reference sequence transformation unit (2019) configured to transform the pre-existing references and data classes (208) into transformed data classes (2018)

In another aspect the genomic encoder further comprises a
data classification unit (204) contains encoders of data classes N, M and I configured with vectors of thresholds generating sub-classes of data classes N, M and I

In another aspect the genomic encoder further comprises the feature that reference transformation unit (2019) applies the same reference transformation (300) for all classes and sub-classes of data.

In another aspect the genomic encoder further comprises the feature that the reference transformation decoder (2019) applies different reference transformations (301, 302, 303) for the different classes and sub-classes of data.

In another aspect the genomic encoder further comprises the features suitable for executing all the aspects of the previously mentioned coding methods.

The present invention further provides a genomic decoder (218) for the decompression of a compressed genomic stream (211) said genomic decoder (218) comprising:
a demultiplexer (210) for demultiplexing compressed genomic data and metadata
parsing means (212-214) configured to parse said compressed genomic stream into genomic layers of syntax elements (215),
one or more layer decoders (216-217), configured to decode the genomic layers into classified reads of sequences of nucleotides (2111),
genomic data classes decoders (219) configured to selectively decode said classified reads of sequences of nucleotides on one or more reference sequences so as to produce uncompressed reads of sequences of nucleotides.

In another aspect the genomic decoder further comprises a reference transformation decoder (2113) configured to decode reference transformation descriptors (2112) and produce a transformed reference (2114) to be used by genomic data class decoders (219).

In another aspect the genomic decoder further comprises that the one or more reference sequences are stored in the compressed genome stream (211).

In another aspect the genomic decoder further comprises that the one or more reference sequences are provided to the decoder via an out of band mechanism.

In another aspect the genomic decoder further comprises that the one or more reference sequences are built at the decoder.

In another aspect the genomic decoder further comprises that the one or more reference sequences are transformed at the decoder by a reference transformation decoder (2113).

The present invention further provides a computer-readable medium comprising instructions that when executed cause at least one processor to perform all the aspects of the previously mentioned coding methods.

The present invention further provides a computer-readable medium comprising instructions that when executed cause at least one processor to perform all the aspects of the previously mentioned decoding methods.

The present invention further provides a support data storing genomic encoded according perform all the aspects of the previously mentioned coding methods.

DETAILED DESCRIPTION

The genomic or proteomic sequences referred to in this invention include, for example, and not as a limitation, nucleotide sequences, Deoxyribonucleic acid (DNA) sequences, Ribonucleic acid (RNA), and amino acid sequences. Although the description herein is in considerable detail with respect to genomic information in the form of a nucleotide sequence, it will be understood that the methods and systems for compression can be implemented for other genomic or proteomic sequences as well, albeit with a few variations, as will be understood by a person skilled in the art.

Genome sequencing information is generated by High Throughput Sequencing (HTS) machines in the form of sequences of nucleotides (a. k. a. "bases") represented by strings of letters from a defined vocabulary. The smallest vocabulary is represented by five symbols: {A, C, G, T, N} representing the 4 types of nucleotides present in DNA namely Adenine, Cytosine, Guanine, and Thymine. In RNA Thymine is replaced by Uracil (U). N indicates that the sequencing machine was not able to call any base and so the real nature of the position is undetermined. In case the IUPAC ambiguity codes are adopted by the sequencing machine, the alphabet used for the symbols is (A, C, G, T, U, W, S, M, K, R, Y, B, D, H, V, N or -).

The nucleotides sequences produced by sequencing machines are called "**reads**". Sequence reads can be between a few dozens to several thousand nucleotides long. Some technologies produce sequence reads in "**pairs**" where one read is from one DNA strand and the second is from the other strand. In genome sequencing the term "**coverage**" is used to express the level of redundancy of the sequence data with respect to a "**reference sequence**". For example, to reach a coverage of 30x on a human genome (3.2 billion bases long) a sequencing machine shall produce a total of 30 x 3.2 billion bases so that in average each position in the reference is "covered" 30 times.

Throughout this disclosure, a **reference sequence** is any sequence on which the nucleotides sequences produced by sequencing machines are aligned/mapped. One example of sequence could actually be a "**reference genome**", a sequence assembled by scientists as a representative example of a species' set of genes. For example GRCh37, the Genome Reference Consortium human genome (build 37) is derived from thirteen anonymous volunteers from Buffalo, New York. However, a reference sequence could also consist of a synthetic sequence conceived and constructed to merely

improve the compressibility of the reads in view of their further processing. This is described in more details in section "Descriptors of Class U and construction of an "internal" references for unmapped reads of "Class U" and "Class HM"" and depicted in figure 22 and 23.

Sequencing devices can introduce errors in the sequence reads such as:

1. the decision of skipping a base call due to the lack of confidence in calling any specific base. This is called an unknown base and labeled as "N" (denoted as mismatch of "n type");
2. the use of a wrong symbol (i.e. representing a different nucleic acid) to represent the nucleic acid actually present in the sequenced sample; this is usually called "substitution error" (denoted as mismatch of "s type");
3. the insertion in one sequence read of additional symbols that do not refer to any actually present nucleic acid; this is usually called "insertion error" (denoted as mismatch of "i type");
4. the deletion from one sequence read of symbols that represent nucleic acids that are actually present in the sequenced sample; this is usually called "deletion error" (denoted as mismatch of "d type");
5. the recombination of one or more fragments into a single fragment which does not reflect the reality of the originating sequence; this usually results in aligners decision to clip bases (denoted as mismatch of "c type").

The term "coverage" is used in literature to quantify the extent to which a reference genome or part thereof can be covered by the available sequence reads. Coverage is said to be:

- partial (less than 1X) when some parts of the reference genome are not mapped by any available sequence read;
- single (1X) when all nucleotides of the reference genome are mapped by one and only one symbol present in the sequence reads;
- multiple (2X, 3X, NX) when each nucleotide of the reference genome is mapped multiple times.

This invention aims at defining a genomic information representation format in which the relevant information is efficiently accessible and transportable and the weight of the redundant information is reduced.

The main innovative aspects of the disclosed invention are the following.

1 Sequence reads are classified and partitioned into data classes according to the results of the alignment with respect to reference sequences. Such classification and partitioning enables the

selective access to encoded data according to criteria related to the alignment results and to the matching accuracy.

2 The classified sequence reads and the associated metadata are represented by homogeneous layers of descriptors to obtain distinct information sources characterized by a low information entropy.

3 The possibility of modeling each separated information source with distinct source model adapted to the statistical characteristics of each class and the possibility of changing the source model within each class of reads and within each descriptor layer for each separately accessible data units (Access Units). The adoption of the appropriate context adaptive probability models and associated entropy coders according to the statistical properties of each source model.

4 The definition of correspondences and dependencies among the descriptors layers to enable the selective access to the sequencing data and associated metadata without the need to decode all the descriptors layers if not all information is required.

5 The coding of each sequence data class and associated metadata layers with respect to "pre-existing" (also denoted as "external") reference sequences or with respect to "transformed" reference sequences obtained by applying appropriate transformations to "pre-existing" reference sequences so as to reduce the entropy of the descriptors layers information sources. Said descriptors represent the reads partitioned into the different data classes. Following any encoding of reads using the corresponding descriptors with reference to a "pre-existing" reference or "transformed" "pre-existing" reference sequence, the occurrence of the various mismatches can be used to define the appropriate transformations to the reference sequences in order to find a final coded representation with low entropy and achieve higher compression efficiency.

6 The construction of one or more reference sequences (also referred to as "internal" references to distinguish from the "pre-existing" also referred here as "external" reference sequences) used to encode the class of reads that present a degree of matching accuracy with respect to the pre-existing reference sequences not satisfying a set of constraints. Such constraints are set with the objective that the coding costs of representing in compressed form the class of reads aligned with respect to the "internal" reference sequences and the cost of representing the "internal" reference sequences themselves, is lower than encoding the unaligned class of reads verbatim, or using the "external" reference sequences without or with transformations.

In the following, each of the above aspects will be further described in details.

Classification of the sequence reads according to matching rules

The sequence reads generated by sequencing machines are classified by the disclosed invention into six different "classes" according to the matching results of the alignment with respect to one or more "pre-existing" reference sequences.

When aligning a DNA sequence of nucleotides with respect to a reference sequence the following cases can be identified:

1. A region in the reference sequence is found to match the sequence read without any error (i.e. perfect mapping). Such sequence of nucleotides is referenced to as "perfectly matching read" or denoted as "Class P".
2. A region in the reference sequence is found to match the sequence read with a type and a number of mismatches determined only by the number of positions in which the sequencing machine generating the read was not able to call any base (or nucleotide). Such type of mismatches are denoted by an "N", the letter used to indicate an undefined nucleotide base. In this document this type of mismatch are referred to as "n type" mismatch. Such sequences belong to "Class N" reads. Once the read is classified to belong to "Class N" it is useful to limit the degree of matching inaccuracy to a given upper bound and set a boundary between what is considered a valid matching and what it is not. Therefore, the reads assigned to Class N are also constrained by setting a threshold (MAXN) that defines the maximum number of undefined bases (i.e. bases called as "N") that a read can contain. Such classification implicitly defines the required minimum matching accuracy (or maximum degree of mismatch) that all reads belonging to Class N share when referred to the corresponding reference sequence, which constitutes an useful criterion for applying selective data searches to the compressed data.

3. A region in the reference sequence is found to match the sequence read with types and number of mismatches determined by the number of positions in which the sequencing machine generating the read was not able to call any nucleotide base, if present (i.e. "n type" mismatches), plus the number of mismatches in which a different base, than the one present in the reference, has been called. Such type of mismatch denoted as "substitution" is also called Single Nucleotide Variation (SNV) or Single Nucleotide Polymorphism (SNP). In this document this type of mismatch is also referred to as "s type" mismatch. The sequence read is then referenced to as "M mismatching reads" and assigned to "Class M". Like in the case of "Class N", also for all reads belonging to "Class M" it is useful to limit the degree of matching inaccuracy to a given upper bound, and set a boundary between what is considered a valid matching and what

it is not. Therefore, the reads assigned to Class M are also constrained by defining a set of thresholds, one for the number "n" of mismatches of "n type" (MAXN) if present, and another for the number of substitutions "s" (MAXS). A third constraint is a threshold defined by any function of both numbers "n" and "s", f(n,s). Such third constraint enables to generate classes with an upper bound of matching inaccuracy according to any meaningful selective access criterion. For instance, and not as a limitation, f(n,s) can be (n+s)1/2 or (n+s) or any linear or non-linear expression that sets a boundary to the maximum matching inaccuracy level that is admitted for a read belonging to "Class M". Such boundary constitutes a very useful criterion for applying the desired selective data searches to the compressed data when analyzing sequence reads for various purposes because it makes possible to set a further boundary to any possible combination of the number of "n type" mismatches and "s type" mismatches (substitutions) beyond the simple threshold applied to the one type or to the other.

4. A fourth class is constituted by sequencing reads presenting at least one mismatch of any type among "insertion", "deletion" (a.k.a. *indels*) and "clipped", plus, if present, any mismatches type belonging to class N or M. Such sequences are referred to as "I mismatching reads" and assigned to "Class I". Insertions are constituted by an additional sequence of one or more nucleotides not present in the reference, but present in the read sequence. In this document this type of mismatch is referred to as "i type" mismatch. In literature when the inserted sequence is at the edges of the sequence it is also referred to as "soft clipped" (i.e. the nucleotides are not matching the reference but are kept in the aligned reads contrarily to "hard clipped" nucleotides which are discarded). In this document this type of mismatch is referred to as "c type" mismatch. Keeping or discarding nucleotides is a decisions taken by the aligner stage and not by the classifier of reads disclosed in this invention which receives and processes the reads as they are determined by the sequencing machine or by the following alignment stage. Deletion are "holes" (missing nucleotides) in the read with respect to the reference. In this document this type of mismatch is referred to as "d type" mismatch. Like in the case of classes "N" and "M" it is possible and appropriate to define a limit to the matching inaccuracy. The definition of the set of constraints for "Class I" is based on the same principles used for "Class M" and is reported in Table 1 in the last table lines. Beside a threshold for each type of mismatch admissible for Class I data, a further constraint is defined by a threshold determined by any function of the number of the mismatches "n", "s", "d", "i" and "c", w(n,s,d,i,c). Such additional constraint make possible to generate classes with an upper bound of matching inaccuracy according to any meaningful user defined selective access criterion. For instance, and not as a limitation, w(n,s,d,i,c) can be (n+s+d+i+c)1/5 or (n+s+d+i+c) or any linear or non-linear expression that sets a boundary to the

maximum matching inaccuracy level that is admitted for a read belonging to "Class I". Such boundary constitutes a very useful criterion for applying the desired selective data searches to the compressed data when analyzing sequence reads for various purposes because it enables to set a further boundary to any possible combination of the number of mismatches admissible in "Class I" reads beyond the simple threshold applied to each type of admissible mismatch.

5. A fifth class includes all reads that do not find any mapping considered valid (i.e not satisfying the set of matching rules defining an upper bound to the maximum matching inaccuracy as specified in Table 1) for each data class when referring to the reference sequence. Such sequences are said to be "Unmapped" when referring to the reference sequences and are classified as belonging to the "Class U".

Classification of read pairs according to matching rules

The classification specified in the previous section concerns single sequence reads. In the case of sequencing technologies that generates read in pairs (i.e. Illumina Inc.) in which two reads are known to be separated by an unknown sequence of variable length, it is appropriate to consider the classification of the entire pair to a single data class. A read that is coupled with another is said to be its "mate".

If both paired reads belong to the same class the assignment to a class of the entire pair is obvious: the entire pair is assigned to the same class for any class (i.e. P, N, M, I, U). In the case the two reads belong to a different class, but none of them belongs to the "Class U", then the entire pair is assigned to the class with the highest priority defined according to the following expression:

$$P < N < M < I$$

in which "Class P" has the lowest priority and "Class I" has the highest priority.

In case only one of the reads belongs to "Class U" and its mate to any of the Classes P, N, M, I a sixth class is defined as "Class HM" which stands for "Half Mapped".

The definition of such specific class of reads is motivated by the fact that it is used for attempting to determine gaps or unknown regions existing in reference genomes (a.k.a. little known or unknown regions). Such regions are reconstructed by mapping pairs at the edges using the pair read that can be mapped on the known regions. The unmapped mate is then used to build the so called "contigs" of the unknown region as it is shown in Figure 28. Therefore providing a selective access to only such type of read pairs greatly reduces the associated computation burden enabling much

efficient processing of such data originated by large amounts of data sets that using the state of the art solutions would require to be entirely inspected.

The table below summarizes the matching rules applied to reads in order to define the class of data each read belongs to. The rules are defined in the first five columns of the table in terms of presence or absence of type of mismatches (n, s, d, i and c type mismatches). The sixth column provide rules in terms of maximum threshold for each mismatch type and any function f(n,s) and w(n,s,d,i,c) of the possible mismatch types.

<table>
<tr><td colspan="5">Number and types of mismatches found when matching a read with a reference sequence</td><td>Set of matching accuracy constraints</td><td>Assignement Class</td></tr>
<tr><td>Number of unknown bases (“N”)</td><td>Number of substitutions</td><td>Number of deletions</td><td>Number of Insertions</td><td>Number of clipped bases</td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>P</td></tr>
<tr><td rowspan="2">n > 0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td>n ≤ MAXN</td><td>N</td></tr>
<tr><td>n > MAXN</td><td>U</td></tr>
<tr><td rowspan="2">n ≥ 0</td><td rowspan="2">s > 0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td>n ≤ MAXN and
s ≤ MAXS and
f(n,s) ≤ MAXM</td><td>M</td></tr>
<tr><td>n > MAXN or
s > MAXS or
f(n,s) > MAXM</td><td>U</td></tr>
<tr><td rowspan="3">n ≥ 0</td><td rowspan="3">s ≥ 0</td><td>d ≥ 0 *</td><td>i ≥ 0*</td><td>c ≥ 0*</td><td rowspan="2">n ≤ MAXN and
s ≤ MAXS and
d ≤ MAXD and
i ≤ MAXI and
c ≤ MAXC
w(n,s,d,i,c) ≤ MAXTOT</td><td rowspan="2">I</td></tr>
<tr><td colspan="3">*At least one mismatch of type d, i, c must be present (i.e. d>0 or i>0 or c>0)</td></tr>
<tr><td>d ≥ 0</td><td>i ≥ 0</td><td>c ≥ 0</td><td>n > MAXN or
s > MAXS or
d > MAXD or</td><td>U</td></tr>
</table>

					i > MAXI or c > MAXC w(n,s,d,i,c) > MAXTOT	

Table 1. Type of mismatches and set of constrains that each sequence reads must satisfy to be classified in the data classes defined in this invention disclosure.

Matching rules partition of sequence read data Classes N, M and I into subclasses with different degrees of matching accuracy

The data classes of type N, M and I as defined in the previous sections can be further decomposed into an arbitrary number of distinct sub-classes with different degrees of matching accuracy. Such option is an important technical advantage in providing a finer granularity and as consequence a much more efficient selective access to each data class. As an example and not as a limitation, to partition the Class N into a number k of subclasses (Sub-Class N_1, …, Sub-Class N_k) it is necessary to define a vector with the corresponding components $MAXN_1$, $MAXN_2$, …, $MAXN_{(k-1)}$, $MAXN_{(k)}$, with the condition that $MAXN_1 < MAXN_2 < \ldots < MAXN_{(k-1)} < MAXN$ and assign each read to the lowest ranked sub-class that satisfy the constrains specified in Table 1 when evaluated for each element of the vector. This is shown in Figure 29 where a data classification unit 291 contains Class P, N, M, I, U, HM encoder and encoders for annotations and metadata. Class N encoder is configured with a vector of thresholds, $MAXN_1$ to $MAXN_k$ 292 which generates k subclasses of N data (296).

In the case of the classes of type M and I the same principle is applied by defining a vector with the same properties for MAXM and MAXTOT respectively and use each vector components as threshold for checking if the functions f(n,s) and w(n,s,d,I,c) satisfy the constraint. Like in the case of sub-classes of type N, the assignment is given to the lowest sub-class for which the constraint is satisfied. The number of sub-classes for each class type is independent and any combination of subdivisions is admissible. This is shown in figure 29 where a Class M encoder 293 and a Class I encoder 294 are configured respectively with a vector of thresholds $MAXM_1$ to $MAXM_j$ and $MAXTOT_1$ to $MAXTOT_h$. The two encoders generate respectively j subclasses of M data (297) and h subclasses of I data (298).

When two reads in a pair are classified in the same sub-class, then the pair belongs to the same sub-class.

When two reads in a pair are classified into sub-classes of different classes, then the pair belongs to the sub-class of the class of higher priority according to the following expression:

$$N < M < I$$

where N has the lowest priority and I has the highest priority.

When two reads belong to different sub-classes of one of classes N or M or I, then the pair belongs to the sub-class with the highest priority according to the following expressions:

$$N_1 < N_2 < \ldots < N_k$$
$$M_1 < M_2 < \ldots < M_j$$
$$I_1 < I_2 < \ldots < I_h$$

where the highest index has the highest priority.

Transformations of the "external" reference sequences

The mismatches found for the reads classified in the classes N, M and I can be used to create "transformed" references to be used to compress more efficiently the read representation.

Reads classified as belonging to the Classes N, M or I (with respect to the "pre-existing" (i.e."external") reference sequence denoted as RS_0) can be coded with respect to the "transformed" reference sequence RS_1 according to the occurrence of the actual mismatches with the "transformed" reference. For example if $read^M_{in}$ belonging to Class M (denoted as the i^{th} read of class M) containing mismatches with respect to the reference sequence RS_n, then after "transformation" $read^M_{in} = read^P_{i(n+1)}$ can be obtained with $A(Ref_n)=Ref_{n+1}$ where A is the transformation from reference sequence RS_n to reference sequence RS_{n+1}.

Figure 19 shows an example on how reads containing mismatches (belonging to Class M) with respect to reference sequence 1 (RS_1) can be transformed into perfectly matching reads with respect to the reference sequence 2 (RS_2) obtained from RS_1 by modifying the bases corresponding to the mismatch positions. They remain classified and they are coded together the other reads in the same data class access unit, but the coding is done using only the descriptors and descriptor values needed for a Class P read. This transformation can be denoted as:

$RS_2 = A(RS_1)$

When the representation of the transformation A which generates RS_2 when applied to RS_1 plus the representation of the reads versus RS_2 corresponds to a lower entropy than the representation of

the reads of class M versus RS_1, it is advantageous to transmit the representation of the transformation A and the corresponding representation of the read versus RS_2 because an higher compression of the data representation is achieved.

The coding of the transformation A for transmission in the compressed bitstream requires the definition of two additional syntax elements as defined in the table below.

Syntax elements	**Semantic**	**Comments**
rftp	Reference transformation position	position of difference between reference and contig used for prediction
rftt	Reference transformation type	type of difference between reference and contig used for prediction. Same syntax described for the snpt descriptor defined below in this document.

Figure 26 shows an example on how a reference transformation is applied to reduce the number of mismatches to be coded on the mapped reads.

It has to be observed that, in some cases the transformation applied to the reference:

- May introduce mismatches in the representations of the reads that were not present when referring to the reference before applying the transformation.
- May modify the types of mismatches, a read may contain A instead of G while all other reads contain C instead of G), but mismatches remain in the same position.
- Different data classes and subsets of data of each data class may refer to the same "transformed" reference sequences or to reference sequences obtained by applying different transformations to the same pre-existing reference sequence.

Figure 27 further shows an example of how reads can change the type of coding from a data class to another by means of the appropriate set of descriptors (e.g. using the descriptors of a Class P to code a read from Class M) after a reference transformation is applied and the read is represented using the "transformed" reference. This occurs for example when the transformation changes all bases corresponding to the mismatches of a read in the bases actually present in the read, thus virtually transforming a read belonging to Class M (when referring to the original non "transformed" reference sequence) into a virtual read of Class P (when referring to the "transformed" reference).

The definition of the set of descriptors used for each class of data is provided in the following sections.

Figure 30 shows how the different classes of data can use the same "transformed" reference $R_1 = A_0(R_0)$ (300) to re-encode the reads, or different transformations A_N (301), A_M (302), A_I (303) can be separately applied to each class of data.

Definition of the information necessary to represent sequence reads into layers of descriptors

Once the classification of reads is completed with the definition of the Classes, further processing consists in defining a set of distinct syntax elements which represent the remaining information enabling the reconstruction of the read sequence when represented as being mapped on a given reference sequence. The data structure of these syntax elements requires the storage of global parameters and metadata to be used by the decoding engine. These data are structured in a ***Genomic Dataset Header*** described in the table below. A dataset is defined as the ensemble of coding elements needed to reconstruct the genomic information related to a single genomic sequencing run and all the following analysis. If the same genomic sample is sequenced twice in two distinct runs, the obtained data will be encoded in two distinct datasets.

Element	Type	Description
Unique ID	Byte array	Unique identifier for the encoded content
Major_Brand	Byte array	Major + Minor version of the encoding algorithm
Minor_Version	Byte array	
Header Size	Integer	Size in bytes of the entire encoded content
Reads Length	Integer	Size of reads in case of constant reads length. A special value (e.g. 0) is reserved for variable reads length
Ref count	Integer	Number of reference sequences used
Access Units counters	Byte array (e.g. integers)	Total Number of encoded Access Units per reference

		sequence
Ref ids	Byte array	Unique identifiers for reference sequences
for (i=0; i<Ref_count; i++) {		
Reference_genome:Ref_ID	string:string	Unambiguous ID, as a characters string, identifying the reference sequence(s) used in this Dataset
}		
for (i=0; i<Ref_count; i++) {		
Ref blocks	Byte array	Number of encoded blocks per each reference
}		
Dataset label size	Integer	The size of the following element
Dataset label	String	A string of character used to identify the dataset
Dataset type	Integer	The type of data encoded in the dataset (e.g. aligned, not aligned)
Master index table *Alignment positions of first read in each block (Access Unit).* *I.e. smaller position of the first read on the reference genome per each block of the six classes* *1 per pos class (six) per reference*	Byte array	This is a multidimensional array supporting random access to Access Units.
Label List *Sub-part of the main header indicating* • *number of Labels* • *for each Label:* ○ the Label ID ○ the number of reference sequences concerned by the label	Byte array	This is a list of Labels, each one represented as a multidimensional array in order to support selective access to specific genomic regions or sub-regions or aggregations of regions or sub-regions.

○ for each reference sequence ▪ the reference identifier ▪ the number of regions covered by the label, ▪ for each region: • the class ID • the start position in the genomic range • the end position in the genomic range *Start position and end position can be replaced by "block numbers", composing, together with reference sequence ID and class ID, a three dimensional vector addressing the coordinates of the Master Index Table.*		
Parameters set	Byte array	Encoding parameters used to configure the encoding process and sent to the decoder.

Table 1 – Genomic Dataset Header structure.

A sequence read (i.e. a DNA segment) referred to a given reference sequence can be fully expressed by:

- The starting position on the reference sequence (pos)
- A flag signaling if the read has to be considered as a reverse complement versus the reference (rcomp).
- A distance, to the mate pair in case of paired reads (pair).
- The value of the read length in case of the sequencing technology produces variable length reads (len). In case of constant reads length the read length associated to each reads can obviously be omitted and can be stored in the main file header.
- For each mismatch:
    - mismatch position (*nmis* for class N, *snpp* for class M, and *indp* for class I)
    - mismatch type (not present in class N, *snpt* in class M, *indt* in class I)
- Flags indicating specific characteristics of the sequence read such as
    - template having multiple segments in sequencing
    - each segment properly aligned according to the aligner
    - unmapped segment
    - next segment in the template unmapped
    - signalization of first or last segment
    - quality control failure
    - PCR or optical duplicate
    - secondary alignment
    - supplementary alignment
- Soft clipped nucleotides string when present (*indc* in class I)

- Flag indicating the reference used for alignment and compression (e.g. “internal” reference for class U) if applicable (descriptor rtype).

- For class U, descriptor indc identifies those parts of the reads (typically the edges) that do not match, with a specified set of matching accuracy constraints, with the “internal” references.
- Descriptor ureads is used to encode verbatim the reads that cannot be mapped on any available reference being it a pre-existing (i.e. “external” like an actual reference genome) or an “internal” reference sequence.

This classification creates groups of descriptors (syntax elements) that can be used to univocally represent genome sequence reads. The table below summarizes the syntax elements needed for each class of reads aligned with "external" (i.e. "pre-existing") or "internal" (i.e. "constructed") references.

	P	N	M	I	U	HM
pos	X	X	X	X	X	X
pair	X	X	X	X		
rcomp	X	X	X	X	X	X
flags	X	X	X	X	X	X
rlen	X	X	X	X	X	X
nmis		X				
snpp			X		X	
snpt			X		X	
indp				X		X
indt				X		X
indc				X	X	X
ureads					X	X
rtype					X	

Table 2 – Defined descriptors layers per class of data.

Reads belonging to class P are characterized and can be perfectly reconstructed by only a position, a reverse complement information and an offset between mates in case they have been obtained by a sequencing technology yielding mated pairs, some flags and a read length.

The next section further details how these descriptors are defined for classes P, N, M and I while for class U they are described in a following section

Class HM is applied to read pairs only and it is a special case for which one read belongs to class P, N, M or I and the other to class U.

Position descriptor layer

In the position (**pos**) layer only the mapping position of the first encoded read is stored as absolute value on the reference sequence. All the other position descriptors assume a value expressing the difference with respect to the previous position. Such modeling of the information source defined by

the sequence of read position descriptors is in general characterized by a reduced entropy particularly for sequencing processes generating high coverage results.

For example, figure 1 shows how after describing the starting position of the first alignment as position "10000" on the reference sequence, the position of the second read starting at position 10180 is described as "180". With high coverages (> 50x) most of the descriptors of the position vector present very high occurrences of low values such as 0 and 1 and other small integers. **Error! Reference source not found.** shows how the positions of three read pairs are described in a **pos** Layer.

Reverse complement descriptor layer

Each read of the read pairs produced by sequencing technologies can be originated from either genome strands of the sequenced organic sample. However, only one of the two strands is used as reference sequence. Figure 2 shows how in a reads pair one read (read 1) can be originated from one strand and the other (read 2) can be originated from the other strand.

When the strand 1 is used as reference sequence, read 2 can be encoded as reverse complement of the corresponding fragment on strand 1. This is shown in figure 3.

In case of coupled reads, four are the possible combinations of direct and reverse complement mate pairs. This is shown in figure 4. The rcomp layer encodes the four possible combinations.

The same encoding is used for the reverse complement information of reads belonging to classes N, M, P and I. In order to enable selective access to the different data classes, the reverse complement information of reads belonging to the four classes are encoded in different layers as depicted in Table 2.

Pairing information descriptor layer

The pairing descriptor is stored in the pair layer. Such layer stores descriptors encoding the information needed to reconstruct the originating reads pairs when the employed sequencing technology produces reads by pairs. Although at the date of the disclosure of the invention the vast majority of sequencing data is generated by using a technology generating paired reads, it is not the case of all technologies. This is the reason for which the presence of this layer is not necessary to reconstruct all sequencing data information if the sequencing technology of the genomic data considered does not generate paired reads information.

Definitions:

- **mate pair**: read associated to another read in a read pair (e.g. Read 2 is the mate pair of Read 1 in the previous example)
- **pairing distance**: number of nucleotide positions on the reference sequence which separate one position in the first read (pairing anchor, e.g. last nucleotide of first read) from one position of the second read (e.g. the first nucleotide of the second read)
- **most probable pairing distance (MPPD)**: this is the most probable pairing distance expressed in number of nucleotide positions.
- **position pairing distance (PPD)**: the PPD is a way to express a pairing distance in terms of the number of reads separating one read from its respective mate present in a specific position descriptor layer.
- **most probable position pairing distance (MPPPD)**: is the most probable number of reads separating one read from its mate pair present in a specific position descriptor layer.
- **position pairing error (PPE)**: is defined as the difference between the MPPD or the MPPPD and the actual position of the mate.
- **pairing anchor**: position of first read last nucleotide in a pair used as reference to calculate the distance of the mate pair in terms of number of nucleotide positions or number of read positions.

Figure 5 shows how the pairing distance among read pairs is calculated.

The pair descriptor layer is the vector of pairing errors calculated as number of reads to be skipped to reach the mate pair of the first read of a pair with respect to the defined decoding pairing distance.

Figure 6 shows an example of how pairing errors are calculated, both as absolute value and as differential vector (characterized by lower entropy for high coverages).

The same descriptors are used for the pairing information of reads belonging to classes N, M, P and I. In order to enable the selective access to the different data classes, the pairing information of reads belonging to the four classes are encoded in different layer as depicted in figures 8 (class N), figures 10, 12 and 14 (class M) and figures 15 and 16 (class I).

Pairing information in case of reads mapped on different reference sequences

In the process of mapping sequence reads on a reference sequence it is not uncommon to have the first read in a pair mapped on one reference sequence (e.g. chromosome 1) and the second on a different reference sequence (e.g. chromosome 4). In this case the pairing information described

above has to be integrated by additional information related to the reference sequence used to map one of the reads. This is achieved by coding:

1. A reserved value (flag) indicating that the pair is mapped on two different sequences (different values indicate if read1 or read2 are mapped on the sequence that is not currently encoded).
2. An unique reference identifier referring to the reference identifiers encoded in the main header structure as described Table 1.
3. The third element contains the mapping information on the reference identified at point 2 and expressed as offset with respect to the last encoded position.

Figure 7 provides an example of this scenario.

In figure 7, since Read 4 is not mapped on the currently encoded reference sequence, the genomic encoder signals this information by crafting additional descriptors in the pair layer. In the example shown below Read 4 of pair 2 is mapped on reference no. 4 while the currently encoded reference is no. 1. This information is encoded using 3 components:

1) One special reserved value is encoded as pairing distance (in this case 0xffffff).
2) A second descriptor provides a reference ID as listed in the main header (in this case 4).
3) The third element contains the mapping information on the concerned reference (170).

Mismatch descriptors for class N reads

Class N includes all reads in which only "n type" mismatches are present, at the place of an A, C, G or T base a N is found as called base. All other bases of the read perfectly match the reference sequence.

Figure 8 shows how:

the positions of "N" in read 1 are coded as

- absolute position in read 1 or
- as differential position with respect to the previous "N" in the same read.

the positions of "N" in read 2 are coded as

- absolute position in read 2 + read 1 length or
- differential position with respect to the previous N

In the **nmis** layer, the coding of each reads pair is terminated by a special "separator" symbol.

Descriptors coding Substitutions (Mismatches or SNPs), Insertions and Deletions

A substitution is defined as the presence, in a mapped read, of a different nucleotide base with respect to the one that is present in the reference sequence at the same position.

Figure 9 shows examples of substitutions in a mapped read pair. Each substitution is encoded as "position" (snpp layer) and "type" (snpt layer). Depending on the statistical occurrence of substitutions, insertion or deletion, different source models of the associated descriptors can be defined and the generated symbols coded in the associated layer.

Source model 1: Substitutions as Positions and Types

Substitutions Positions Descriptors

A substitution position is calculated like the values of the nmis layer, i.e.

In read 1 substitutions are encoded

- as absolute position in read 1 or
- as differential position with respect to the previous substitution in the same read In read 2 substitutions are encoded
- as absolute position in read 2 + read 1 length or
- as differential position with respect to the previous substitution

Figure 10 shows how substitutions (where, at a given mapping position, a symbol in a read is different from the symbol in the reference sequence) are coded as

1. the position of the mismatch
    - with respect to the beginning of the read or
    - with respect to the previous mismatch (differential encoding)
2. the type of mismatch represented as a code calculated as described in Figure 10

In the snpp layer, the coding of each reads pair is terminated by a special "separator" symbol.

Substitutions Types Descriptors

For class M (and I as described in the next sections), mismatches are coded by an index (moving from right to left) from the actual symbol present in the reference to the corresponding substitution symbol present in the read {A, C, G, T, N, Z}. For example if the aligned read presents a C instead of a T which is present at the same position in the reference, the mismatch index will be denoted as "4". The decoding process reads the encoded syntax element, the nucleotide at the given position on the reference and moves from left to right to retrieve the decoded symbol. E.g. a "2" received for a position where a G is present in the reference will be decoded as "N". Figure 11 shows all the possible substitutions and the respective encoding symbols. Obviously different and context

adaptive probability models can be assigned to each substitution index according to the statistical properties of each substitution type for each data class to minimize the entropy of the descriptors.

In case of adoption of the IUPAC ambiguity codes the substitution mechanism results to be exactly the same however the substitution vector is extended as: S = {A, C, G, T, N, Z, M, R, W, S, Y, K, V, H, D, B}.

Figure 12 provides an example of encoding of substitutions types in the ***snpt*** layer.

Some examples of substitutions encoding when IUPAC ambiguity codes are adopted are provided in Figure 13. A further example of substitution indexes is provided in Figure 14.

Encoding of insertions and deletions

For class I, mismatches and deletions are coded by an indexes (moving from right to left) from the actual symbol present in the reference to the corresponding substitution symbol present in the read: {A, C, G, T, N, Z}. For example if the aligned read presents a C instead of a T present at the same position in the reference, the mismatch index will be "4". In case the read presents a deletion where a "A" is present in the reference, the coded symbol will be "5". The decoding process reads the coded syntax element, the nucleotide at the given position on the reference and moves from left to right to retrieve the decoded symbol. E.g. a "3" received for a position where a G is present in the reference will be decoded as "Z".

Inserts are coded as 6, 7, 8, 9, 10, respectively for inserted A, C, G, T, N.

Figure 15 shows an example of how to encode substitutions, inserts and deletions in a reads pair of class I. In order to support the entire set of IUPAC ambiguity codes, the substitution vector S= {A, C, G, T, N, Z} shall be replaced by S = {A, C, G, T, N, Z, M, R, W, S, Y, K, V, H, D, B} as described in the previous paragraph for mismatches. In this case the insertion codes need to have different values, namely 16, 17, 18, 19, 20 in case the substitution vector has 16 elements. The mechanism is illustrated in Figure 16.

Source model 2: one layer per substitution type and indels

For some data statistics a different coding model from the one described in the previous section can be developed for substitutions and indels resulting into a source with lower entropy. Such coding model is an alternative to the techniques described above for mismatches only and for mismatches and indels.

In this case one data layer is defined for each possible substitution symbol (5 without IUPAC codes, 16 with IUPAC codes), plus one layer for deletions and 4 more layers for insertions. For simplicity of

the explanation, but not as a limitation for the application of the model, the following description will focus on the case where no IUPAC codes are supported.

Figure 17 shows how each layer contains the position of the mismatches or inserts of a single type. If no mismatches or inserts for that type is present in the encoded read pair, a 0 is encoded in the corresponding layer. To enable the decoder to start the decoding process for the layers described in this section, the header of each Access Units contains a flag signaling the first layer to be decoded. In the example of figure 18 the first element to be decoded is position 2 in the C layer. When no mismatches or indels of a given type are present in a read pair, a 0 is added to the corresponding layers. On the decoding side, when the decoding pointer for each layer points to a value of 0, the decoding process moves to the next read pair.

Encoding of additional signaling flags

Each data class introduced above (P, M, N, I) may require the encoding of additional information on the nature of the encoded reads. This information may be related for example to the sequencing experiment (e.g. indicating a probability of duplication of one read) or can express some characteristic of the read mapping (e.g. first or second in pair). In the context of this invention this information is encoded in a separate layer for each data class. The main advantage of such approach is the possibility to selectively access this information only in case of need and only in the required reference sequence region. Other examples of the use of such flags are:

- read paired
- read mapped in proper pair
- read or mate unmapped
- read or mate from reverse strand
- first/second in pair
- not primary alignment
- read fails platform/vendor quality checks
- read is PCR or optical duplicate
- supplementary alignment

Descriptors for class U and construction of "internal" references for unmapped reads of "Class U" and "Class HM"

In the case of the reads belonging to Class U or the unmapped pair of "Class HM" since they cannot be mapped to any "external" reference sequence satisfying the specified set of matching accuracy constraints for belonging to any of the classes P, N, M, or I, one or more "internal" reference sequences are "constructed" and used for the compressed representation of the reads belonging to these data classes.

Several approaches are possible to construct appropriate "internal" references such as for instance and not as limitation:

- the partitioning of the unmapped reads into clusters containing reads that share a common contiguous genomic sequence of at least a minimal size (signature). Each cluster can be uniquely identified by its signature as shown in figure 22.
- the sorting of reads in any meaningful order (e.g. lexicographic order) and the use of the last N reads as "internal" reference for the encoding of the N+1. This method is shown in figure 23.
- performing a so called "de-novo assembly" on a subset of the reads of class U so as to be able to align and encode all or a relevant sub-set of the reads belonging to said class according to the specified matching accuracy constraints or a new set of constraints.

If the read being coded can be mapped on the "internal" reference satisfying the specified set of matching accuracy constraints, the information necessary to reconstruct the read after compression is coded using syntax elements that can be of the following types:

1. Start position of the matching portion on the internal reference in terms of read number in the internal reference (pos layer). This position can be encoded either as absolute or differential value with respect to the previously encoded read.
2. Offset of the start position from the beginning of the corresponding read in the internal reference (pair layer). E.g. in case of constant read length the actual position is pos *length + pair.
3. Possibly present mismatches coded as mismatch position (snpp layer) and type (snpt layer)
4. Those parts of the reads (typically the edges identified by pair) that do not match with the internal reference (or do so, but with a number of mismatches above a defined threshold) are encoded in the indc layer. A padding operation can be performed to the edges of the part of the internal reference used in order to reduce the entropy of the mismatches encoded in the indc layer, as shown in figure 24. The most appropriate padding strategy can be chosen by the encoder according to the statistical properties of the genomic data being processed. Possible padding strategies include:

a. No padding
b. Constant padding pattern chosen according to its frequency in the currently encoded data.
c. Variable padding pattern according to the statistical properties of the current context defined in terms of the latest N encoded reads

The specific type of padding strategy will be signaled by special values in the indc layer header

5. A flag that indicates if the read has been encoded using an internal self-generated, external or no-reference (rtype layer)
6. Reads which are encoded verbatim (ureads).

Figure 24 provides an example of such coding procedure.

Figure 25 shows an alternative encoding of unmapped reads on the internal reference where pos + pair syntax elements are replaced by a signed pos. In this case pos would express the distance – in terms of positions on the reference sequence - of the left most nucleotide position of read n with respect of the position of the left most nucleotide of read n-1.

In case reads of class U present variable length, an additional descriptor rlen is used to store each read length.

This coding approach can be extended to support N start positions per read so that reads can be split over two or more reference positions. This can be particularly useful to encode reads generated by those sequencing technology (e.g. from Pacific Bioscience) producing very long reads (50K+ bases) which usually present repeated patterns generated by loops in the sequencing methodology. The same approach can be used as well to encode chimeric sequence reads defined as reads that align to two distinct portions of the genome with little or no overlap.

The approach described above can be clearly applied beyond the simple class U and could be applied to any layer containing syntax elements related to reads positions (pos layers).

Source models, entropy coders and coding modes

For each data class, sub-class and associated descriptor layer of the genomic data structure disclosed in this invention different coding algorithms may be adopted according to the specific features of the data or metadata carried by each layer and its statistical properties. The “coding algorithm” has to be intended as the association of a specific “source model” of the descriptor layer with a specific “entropy coder”. The specific “source model” can be specified and selected to obtain the most efficient coding of the data in terms of minimization of the source entropy. The selection of the entropy coder can be driven by coding efficiency considerations and/or probability distribution features and associated implementation issues. Each selection of a specific “coding algorithm”, also referred to as “coding mode” can be applied to an entire “descriptor layer” associated to a data class

or sub-class for the entire data set, or different "coding modes" can be applied for each portion of descriptors partitioned into Access Units.

Each "source model" associated to a coding mode is characterized by:

- The definition of the syntax elements emitted by each source (i.e.. the set of descriptors used to represent a class of data such as reads position, reads pairing information, mismatches with respect to a reference sequence as defined in Table 2).
- The definition of the associated probability model.
- The definition of the associated entropy coder.

Further advantages

The classification of sequence data into the defined data classes and sub-classes permits the implementation of efficient coding modes exploiting the lower information source entropy characterizing by modelling the sequences of syntax elements by single separate data sources (e.g. distance, position, etc.).

Another advantage of the invention is the possibility to access only the subset of type of data of interest. For example one of the most important application in genomics consists in finding the differences of a genomic sample with respect to a reference (SNV) or a population (SNP). Today such type of analysis requires the processing of the complete sequence reads whereas by adopting the data representation disclosed by the invention the mismatches are already isolated into one to three data classes only (depending on the interest in considering also "n type" and "i type" mismatches).

A further advantage is the possibility of performing efficient transcoding from data and metadata compressed with reference to a specific "external" reference sequence to another different "external" reference sequence when new reference sequences are published or when re-mapping is performed on the already mapped data (e.g. using a different mapping algorithm) obtaining new alignments.

Figure 20 shows an encoding apparatus 207 according to the principles of this invention. The encoding apparatus 207 receives as input a raw sequence data 209, for example produced by a genome sequencing apparatus 200. Genome sequencing apparatus 200 are known in the art, like the Illumina HiSeq 2500 or the Thermo-Fisher Ion Torrent devices. The raw sequence data 209 is fed to an aligner unit 201, which prepares the sequences for encoding by aligning the reads to a reference sequence 2020. Alternatively, a dedicated module 202 can be used to generate a reference sequence from the available reads by using different strategies as described in this document in section "Construction of internal references for unmapped reads of Class U" and "Class HM". After

having been processed by the reference generator 202, reads can be mapped on the obtained longer sequence. The aligned sequences are then classified by data classification module 204. A further step of reference transformation is then applied on the reference in order to reduce the entropy of the data generated by the data classification unit 204. This implies processing the external reference 2020 into a reference transformation unit 2019 which produces transformed data classes 2018 and reference transformation descriptors 2021. The transformed data classes 2018 are then fed to layers encoders 205-207 together with the reference transformation descriptors 2021. The genomic layers 2011 are then fed to arithmetic encoders 2012-2014 which encode the layers according to the statistical properties of the data or metadata carried by the layer. The result is a genomic stream 2015.

Figure 21 shows a decoding apparatus 218 according to the principles of this disclosure. A decoding apparatus 218 receives a multiplexed genomic bitstream 2110 from a network or a storage element. The multiplexed genomic bitstream 2110 is fed to a demultiplexer 210, to produce separate streams 211 which are then fed to entropy decoders 212-214, to produce genomic layers 215 and reference transformation descriptors 2112. The extracted genomic layers are fed to layer decoders 216-217 to further decode the layers into classes of data and the reference transformation descriptors are fed to a reference transformation unit 2113. Class decoders 219 further process the genomic descriptors 2111 and the transformed reference 2114, and merge the results to produce uncompressed reads of sequences, which can then be further stored in the formats known in the art, for instance a text file or zip compressed file, or FASTQ or SAM/BAM files.

Class decoders 219 are able to reconstruct the original genomic sequences by leveraging the information on the original reference sequences carried by one or more genomic streams and the reference transformation descriptors 2112 carried in the encoded bitstream. In case the reference sequences are not transported by the genomic streams they must be available at the decoding side and accessible by the class decoders.

The inventive techniques herewith disclosed may be implemented in hardware, software, firmware or any combination thereof. When implemented in software, these may be stored on a computer medium and executed by a hardware processing unit. The hardware processing unit may comprise one or more processors, digital signal processors, general purpose microprocessors, application specific integrated circuits or other discrete logic circuitry.

The techniques of this disclosure may be implemented in a variety of devices or apparatuses, including mobile phones, desktop computers, servers, tablets and similar devices.

File Format: Selective Access to Regions of Genomic Data by Using the Master Index Table

In order to support selective access to specific regions of the aligned data, the data structure described in this document implements an indexing tool called Master Index Table (MIT). This is a multi-dimensional array containing the loci at which specific reads map on the associated reference sequences. The values contained in the MIT are the mapping positions of the first read in each ***pos*** layer so that non-sequential access to each Access Unit is supported. The MIT contains one section per each class of data (P, N, M, I, U and HM) and per each reference sequence. The MIT is contained in the Genomic Dataset Header of the encoded data. Figure 31 shows the structure of the Genomic Dataset Header, figure 32 shows a generic visual representation of the MIT and figure 33 shows an example of MIT for the class P of encoded reads.

The values contained in the MIT depicted in figure 33 are used to directly access the region of interest (and the corresponding AU) in the compressed domain.

For example, with reference to figure 33, if it is required to access the region comprised between position 150,000 and 250,000 on reference 2, a decoding application would skip to the second reference in the MIT and would look for the two values k1 and k2 so that k1 < 150,000 and k2 > 250,000. Where k1 and k2 are 2 indexes read from the MIT. In the example of **Error! Reference source not found.** this would result in positions 3 and 4 of the second vector of the MIT. These returned values will then be used by the decoding application to fetch the positions of the appropriate data from the ***pos*** layer Local Index Table as described in the next section.

Together with pointers to the layer containing the data belonging to the four classes of genomic data described above, the MIT can be uses as an index of additional metadata and/or annotations added to the genomic data during its life cycle.

Local Index Table

Each genomic data layer is prefixed with a data structure referred to as ***local header***. The local header contains a unique identifier of the layer, a vector of Access Units counters per each reference sequence, a Local Index Table (LIT) and optionally some layer specific metadata. The LIT is a vector of pointers to the physical position of the data belonging to each Access Unit in the layer payload. Figure 34 depicts the generic layer header and payload where the LIT is used to access specific regions of the encoded data in a non-sequential way.

In the previous example, in order to access region 150,000 to 250,000 of reads aligned on the reference sequence no. 2, the decoding application retrieved positions 3 and 4 from the MIT. These values shall be used by the decoding process to access the 3rd and 4th elements of the corresponding section of the LIT. In the example shown in figure 35, the Total Access Units counters contained in

the layer header are used to skip the LIT indexes related to AUs related to reference 1 (5 in the example). The indexes containing the physical positions of the requested AUs in the encoded stream are therefore calculated as:

position of the data blocks belonging to the requested AU = data blocks belonging to AUs of reference 1 to be skipped + position retrieved using the MIT, i.e.

First block position: 5 + 3 = 8

Last block position: 5 + 4 = 9

The blocks of data retrieved using the indexing mechanism called Local Index Table, are part of the Access Units requested.

FIgure 36 shows how the blocks contained in the MIT table correspond to blocks of the LIT per each class or sub-class of data.

Figure 37 shows how the data blocks retrieved using the MIT and the LIT compose one or more Access Units as defined in the following section.

Access Units

The genomic data classified in data classes and structured in compressed or uncompressed layers are organized into different Access Units.

Genomic Access Units (AU) are defined as sections of genome data (in a compressed or uncompressed form) that reconstructs nucleotide sequences and/or the relevant metadata, and/or sequence of DNA/RNA (e.g. the virtual reference) and/or annotation data generated by a genome sequencing machine and/or a genomic processing device or analysis application. An example of Access Unit is provided in figure 37.

An Access Unit is a block of data that can be decoded either independently from other Access Units by using only globally available data (e.g. decoder configuration) or by using information contained in other Access Units.

Access Units are differentiated by:

- type, characterizing the nature of the genomic data and data sets they carry and the way they can be accessed,
- order, providing a unique order to Access Units belonging to the same type.

Access units of any type can be further classified into different “categories”.

Hereafter follows a non-exhaustive list of definition of different types of genomic Access Units:

1) Access units of type 0 do not need to refer to any information coming from other Access Units to be accessed or decoded and accessed. The entire information carried by the data or data sets they contain can be independently read and processed by a decoding device or processing application.

2) Access units of type 1 contain data that refer to data carried by Access Units of type 0. Reading or decoding and processing the data contained in Access Units of type 1 requires having access to one or more Access Units of type 0. Access unit of type 1 encode genomic data related to sequence reads of "Class P"

3) Access Units of type 2 contain data that refer to data carried by Access Units of type 0. Reading or decoding and processing the data contained in Access Units of type 2 requires having access to one or more Access Units of type 0. Access unit of type 2 encode genomic data related to sequence reads of "Class N"

4) Access Units of type 3 contain data that refer to data carried by Access Units of type 0. Reading or decoding and processing the data contained in Access Units of type 3 requires having access to one or more Access Units of type 0. Access unit of type 3 encode genomic data related to sequence reads of "Class M"

5) Access Units of type 4 contain data that refer to data carried by Access Units of type 0. Reading or decoding and processing the data contained in Access Units of type 4 requires having access to one or more Access Units of type 0. Access unit of type 4 encode genomic data related to sequence reads of "Class I"

6) Access Units of type 5 contain reads that cannot be mapped on any available reference sequence ("Class U") and are encoded used an internally constructed reference sequence. Access Units of type 5 contain data that refer to data carried by Access Units of type 0. Reading or decoding and processing the data contained in Access Units of type 5 requires having access to one or more Access Units of type 0.

7) Access Units of type 6 contain read pairs where one read can belong to any of the four classes P, N, M, I and the other cannot be mapped on any available reference sequence ("Class HM"). Access Units of type 6 contain data that refer to data carried by Access Units of

type 0. Reading or decoding and processing the data contained in Access Units of type 6 requires having access to one or more Access Units of type 0.

8) Access Units of type 7 contain metadata (e.g. quality scores) and/or annotation data associated to the data or data sets contained in the access unit of type 1. Access Units of type 7 may be classified and labelled in different layers.

9) Access Units of type 8 contain data or data sets classified as annotation data. Access Units of type 8 may be classified and labelled in layers.

10) Access Units of additional types can extend the structure and mechanisms described here. As an example, but not as a limitation, the results of genomic variant calling, structural and functional analysis can be encoded in Access Units of new types. The data organization in Access Units described herein does not prevent any type of data to be encapsulated in Access Units being the mechanism completely transparent with respect to the nature of encoded data.

Access Units of type 0 are ordered (e.g. numbered), but they do not need to be stored and/or transmitted in an ordered manner (technical advantage: parallel processing/parallel streaming, multiplexing)
Access Units of type 1, 2, 3, 4, 5 and 6 do not need to be ordered and do not need to be stored and/or transmitted in an ordered manner (technical advantage: parallel processing / parallel streaming).

Figure 37 shows how Access Units are composed by a header and one or more layers of homogeneous data. Each layer can be composed by one or more blocks. Each block contains several packets and the packets are a structured sequence of the descriptors introduced above to represent e.g. reads positions, pairing information, reverse complement information, mismatches positions and types etc.
Each Access unit can have a different number of packets in each block, but within an Access Unit all blocks have the same number of packets.
Each data packet can be identified by the combination of 3 identifiers X Y Z where:

- X identifies the access unit it belongs to
- Y identifies the block it belongs to (i.e. the data type it encapsulates)

- Z is an identifier expressing the packet order with respect to other packets in the same block

Figure 38 shows an example of Access Units and packets labelling where ***AU_T_N*** is an access unit of type T with identifier N which may or may not imply a notion of order according to the Access Unit Type. Identifiers are used to uniquely associate Access Units of one type with those of other types required to completely decode the carried genomic data.

Access Units of any type can be further classified and labelled in different "categories" according to different sequencing processes. For example, but not as a limitation, classification and labelling can take place when

1. sequencing the same organism at different times (Access Units contain genomic information with a "temporal" connotation),
2. sequencing organic samples of different nature of the same organisms (e.g. skin, blood, hair for human samples) . These are Access Units with "biological" connotation.

CLAIMS

1 A method for encoding genome sequence data, said genome sequence data comprising reads of sequences of nucleotides, said method comprising the steps of:
aligning said reads to one or more reference sequences thereby creating aligned reads,

classifying said aligned reads according to specified matching rules with said one or more reference sequences, thereby creating classes of aligned reads,

encoding said classified aligned reads as a multiplicity of layers of syntax elements,

wherein encoding said classified aligned reads as a multiplicity of layers of syntax elements comprises selecting said syntax elements according to said classes of aligned reads,
structuring said layers of syntax elements with header information thereby creating successive Access Units.

2 The encoding method of claim 1 further comprising:
further classifying said reads that do not satisfy said specified matching rules into a class of unmapped reads

constructing a set of reference sequences using at least some unmapped reads

aligning said class of unmapped reads to the set of constructed reference sequences

encoding said classified aligned reads as a multiplicity of layers of syntax elements,

encoding said set of constructed reference sequences

structuring said layers of syntax elements and said encoded reference sequences with header information thereby creating successive Access Units.

3 The method of claims 1-2, wherein said classifying comprises identifying genomic reads without any mismatch in the reference sequence as first “Class P”

4 The method of claims 1-3, wherein said classifying further comprises identifying genomic reads as a second "Class N" when mismatches are only found in the positions where the sequencing machine was not able to call any "base" and the number of mismatches in each read does not exceed a given threshold

5 The method of any preceding claims 1-4, wherein said classifying further comprises identifying genomic reads as a third "Class M" when mismatches are found in the positions where the sequencing machine was not able to call any "base", named "n type" mismatches, and/or it called a different "base" than the reference sequence, named "s type" mismatches, and the number of mismatches does not exceed given thresholds for the number of mismatches of "n type", of "s type" and a threshold obtained from a given function (f(n,s))

6 The method of any preceding claims 1-5, wherein said classifying further comprises identifying genomic reads as a fourth "Class I" when they can possibly have the same type of mismatches of "Class M", and in addition at least one mismatch of type: "insertion" ("i type") "deletion" ("d type") soft clips ("c type"), and wherein the number of mismatches for each type does not exceed the corresponding given threshold and a threshold provided by a given function (w(n,s,i,d,c))

7 The method of any preceding claims 1-6, wherein said classifying further comprises identifying genomic reads as a fifth "Class U" as comprising all reads that do not find any classification in the Classes P, N, M, I, as defined in claims 3-6

8 The encoding method of claims 1-2 wherein the reads of the genomic sequence to be encoded are paired

9 The method of any preceding claims 1-8, wherein said classifying further comprises identifying genomic reads as a sixth "Class HM" as comprising all reads pairs where one read belong to Class P, N, M or I and the other read belong to "Class U"

10 The encoding method of claims 8-9 further comprising the steps of

Identifying if the two mate reads are classified in the same class (each of: P, N, M, I, U), then assigning the pair to the same identified class,

Identifying if the two mate reads are classified in different classes, and in case none of them belongs to the "Class U", then assigning the pair of reads to the class with the highest priority defined according to the following expression:

P < N < M < I

in which "Class P" has the lowest priority and "Class I" has the highest priority;

identifying if only one of the two mate reads has been classified as belonging to "Class U" and classifying

the pair of reads as belonging to the "Class HM" sequences

11 The method of any preceding claims 1-7 where each Class of reads N, M, I is further partitioned into two or more subclasses (296, 297, 298) according to a vector of thresholds (292, 293, 294) defined respectively for each class N, M and I, by the number of "n type" mismatches (292), the function f(n,s) (293) and the function w(n,s,i,d,c) (294).

12 The encoding method of claims 8-11 further comprising the steps of:

identifying if the two mate reads are classified in the same subclass, then assigning the pair to the same sub-class

identifying if the two mate reads are classified into sub-classes of different Classes, then assigning the pair to the subclass belonging to the Class of higher priority according to the following expression:

N < M < I

where N has the lowest priority and I has the highest priority;

identifying if the two mate reads are classified in the same class, and such class is N or M or I, but in different sub-classes, then assigning the pair to the sub-class with the highest priority according to the following expressions:

$N_1 < N_2 < \ldots < N_k$

$M_1 < M_2 < \ldots < M_j$

$I_1 < I_2 < \ldots < I_h$

where the highest index has the highest priority.

13 The method of claims 1-12 wherein said layers of syntax elements comprise a master index table, containing one section for each Class and sub-class of aligned reads, said section comprising the mapping positions on said one or more reference sequences of the first read of each Access Units of each Class or sub-class of data; jointly coding said master index table and said access unit data.

14 The method of claim 13, wherein said layers of syntax elements further comprise information related to the type of reference used (pre-existing or constructed) and the segments of the read that do not match on the reference sequence.

15 The method of claim 1-14, wherein said reference sequences are first transformed into different reference sequences by applying substitutions, insertions, deletions and clipping, then the encoding of said classified aligned reads as a multiplicity of layers of syntax elements refers to the transformed reference sequences.

16 The method of claim 15 wherein the same transformation is applied to the reference sequences for all classes of data.

17 The method of claim 15 where different transformations are applied to the reference sequences per each class of data.

18 The methods of claims 15-17 where the reference sequences transformations are encoded as layers of syntax elements and structured with header information thereby creating successive Access Units.

19 The method of claims 1-18, wherein the encoding of said classified aligned reads and the related reference sequences transformations as multiplicity of layers of syntax elements comprises the step of associating a specific source model and a specific entropy coder to each descriptor layer.

20 The method of claim 19, wherein said entropy coder is one of a context adaptive arithmetic coder, a variable length coder or a golomb coder

21 A method for decoding encoded genomic data comprising the steps of:

parsing Access Units containing said encoded genomic data to extract multiple layers of syntax elements by employing header information

decoding said multiplicity of layers of syntax elements to extract aligned reads according to specific matching rules defining their classification with respect to one or more reference sequences

22 The decoding method of claim 21 further comprising the decoding of unmapped genomic reads coded as in claim 2

23 The decoding method of claims 21-22 wherein the genomic reads are classified as in claims 3-12

24 The decoding method of claims 21-23 further comprising decoding a master index table containing one section for each class of reads and the associated relevant mapping positions

25 The decoding method of claims 21-24 further comprising decoding information related to the type of reference used: pre-existing, transformed or constructed.

26 The decoding method of claims 21-25 further comprising decoding information related to one or more transformations to be applied to the pre-existing reference sequences.

27 The decoding method of claims 21-25 wherein said genomic reads are paired as in claims 11-12

28 The decoding method of claims 21-25 wherein said genomic data are entropy decoded

29 A genomic encoder (2010) for the compression of genome sequence data 209, said genome sequence data 209 comprising reads of sequences of nucleotides,
said genomic encoder (2010) comprising:
an aligner unit (201), configured to align said reads to one or more reference sequences thereby creating aligned reads,
a constructed-reference generator unit (202), configured to produce constructed reference sequences
a data classification unit (204), configured to classify said aligned reads according to specified matching rules with the one or more pre-existing reference sequences or constructed reference sequences thereby creating classes of aligned reads (208);

one or more layers encoding units (205-207), configured to encode said classified aligned reads as layers of syntax elements by selecting said syntax elements according to said classes of aligned reads.

a multiplexer (2016) for multiplexing the compressed genomic data and metadata

30 The genomic encoder of claim 29 further comprising
a reference sequence transformation unit (2019) configured to transform the pre-existing references and data classes (208) into transformed data classes (2018)

31 The genomic encoder of claim 29-30 where the data classification unit (204) contains encoders of data classes N, M and I configured with vectors of thresholds generating sub-classes of data classes N, M and I

32 The genomic encoder of claim 29, wherein the reference transformation unit (2019) applies the same reference transformation (300) for all classes and sub-classes of data.

33 The genomic decoder of claim 29, wherein the reference transformation decoder (2019) applies different reference transformations (301, 302, 303) for the different classes and sub-classes of data.

34 The genomic encoder of claim 29 further comprising coding means suitable for executing the coding method of claims 1-20

35 A genomic decoder (218) for the decompression of a compressed genomic stream (211) said genomic decoder (218) comprising:
a demultiplexer (210) for demultiplexing compressed genomic data and metadata
parsing means (212-214) configured to parse said compressed genomic stream into genomic layers of syntax elements (215),
one or more layer decoders (216-217), configured to decode the genomic layers into classified reads of sequences of nucleotides (2111),
genomic data classes decoders (219) configured to selectively decode said classified reads of sequences of nucleotides on one or more reference sequences so as to produce uncompressed reads of sequences of nucleotides.

36 The genomic decoder of claim 35 further comprising a reference transformation decoder (2113) configured to decode reference transformation descriptors (2112) and produce a transformed reference (2114) to be used by genomic data class decoders (219).

37 The genomic decoder of claims 35-36, wherein the one or more reference sequences are stored in the compressed genome stream (211).

38 The genomic decoder of claim 35-36, wherein the one or more reference sequences are provided to the decoder via an out of band mechanism.

39 The genomic decoder of claim 35-36, wherein the one or more reference sequences are built at the decoder.

40 The genomic decoder of claim 35-36, wherein one or more reference sequences are transformed at the decoder by a reference transformation decoder (2113).

41 A computer-readable medium comprising instructions that when executed cause at least one processor to perform the encoding method of any one of claims 1 to 20.

42 A computer-readable medium comprising instructions that when executed cause at least one processor to perform the decoding method of any one of claims 21 to 28.

43 Support data storing genomic encoded according to the method of claims 1-20

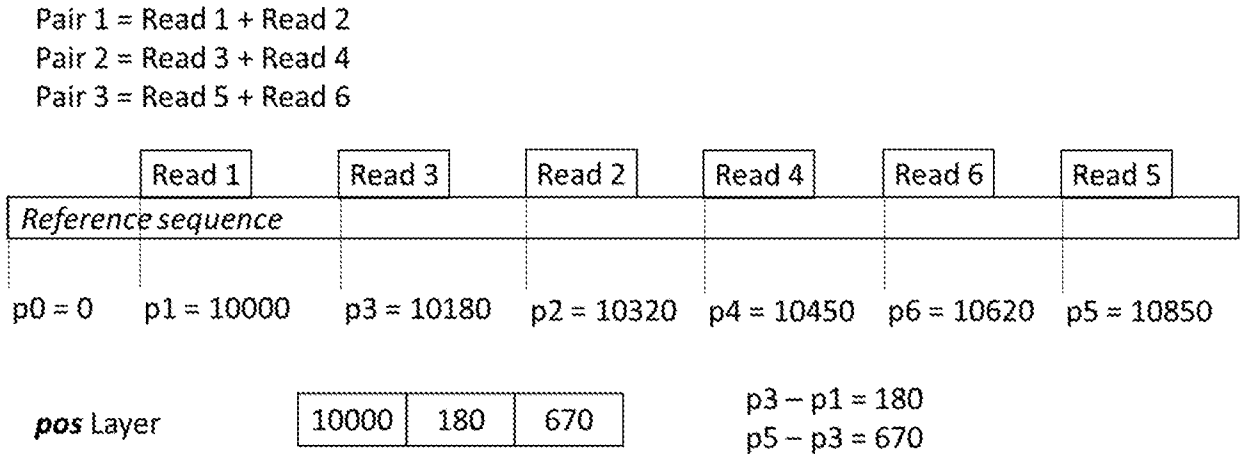


Figure 1

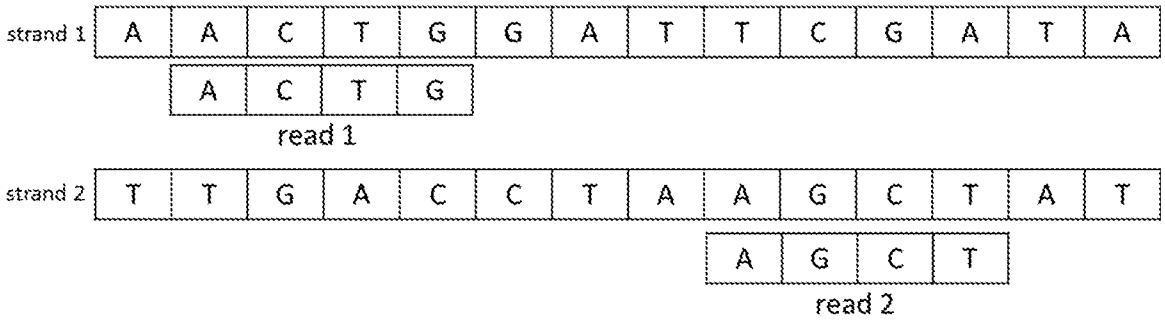


Figure 2

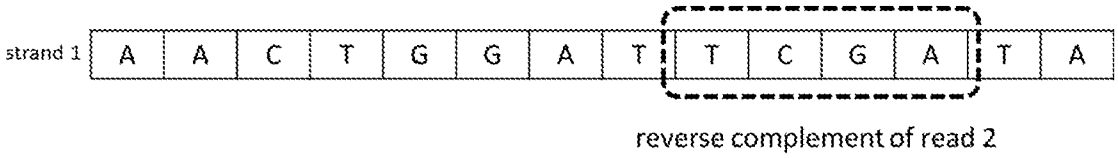


Figure 3

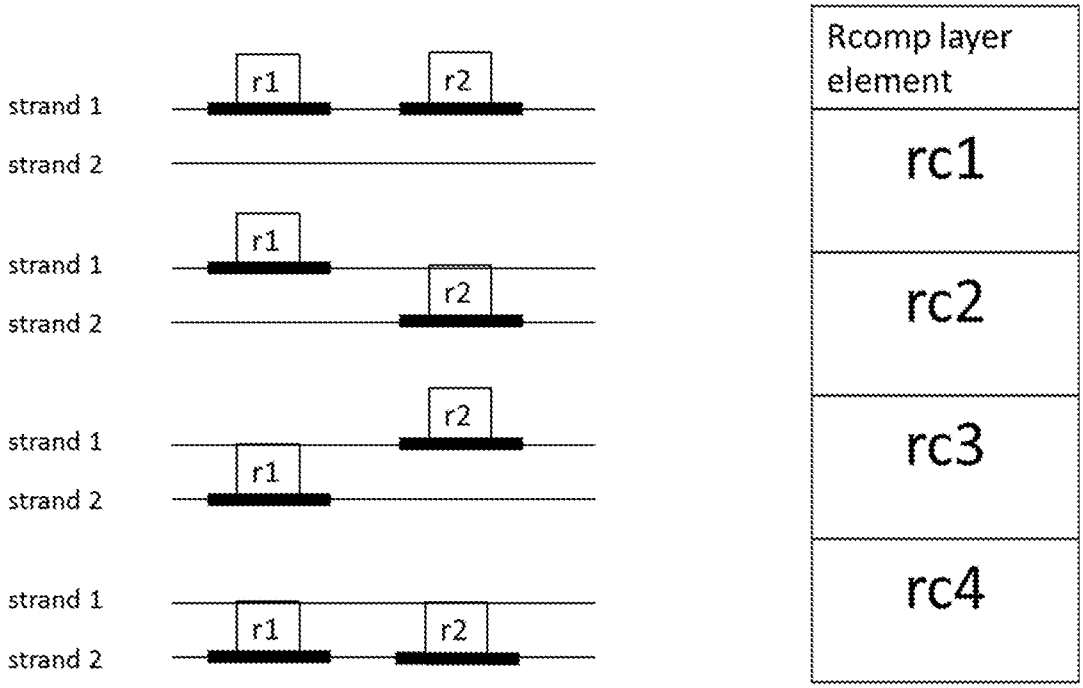


Figure 4

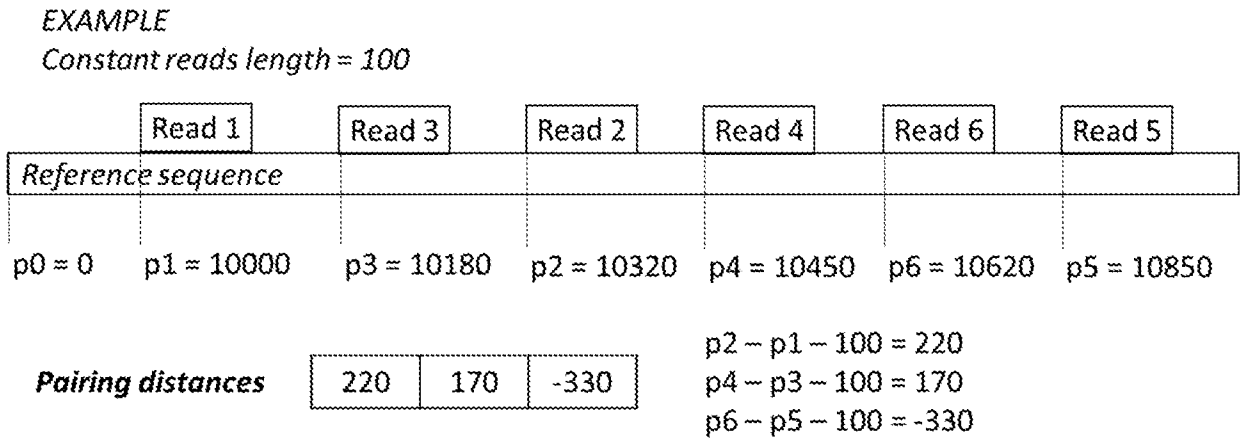


Figure 5

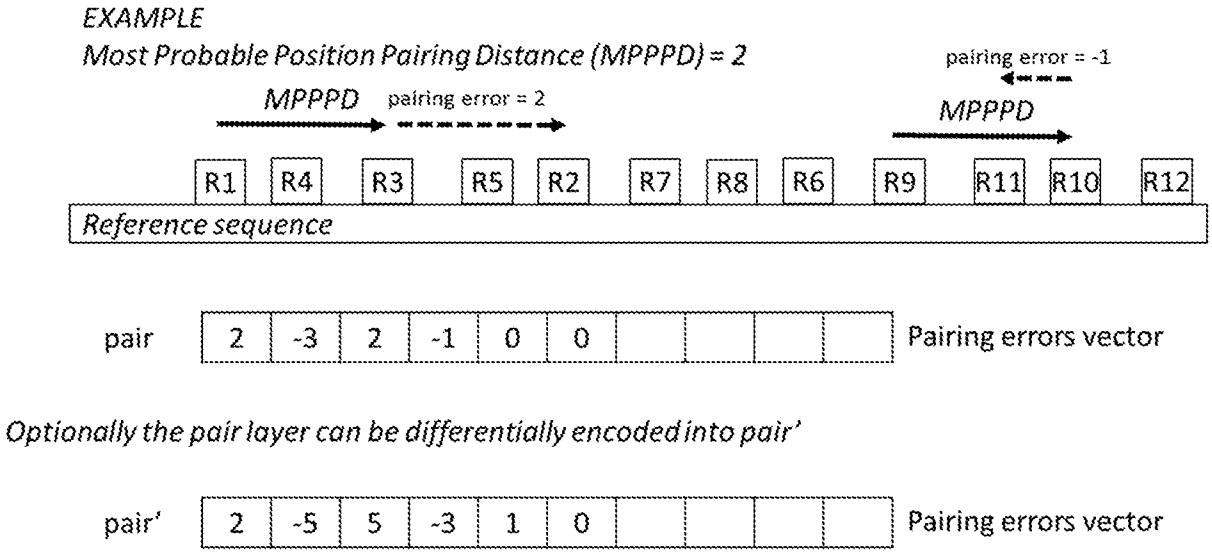


Figure 6

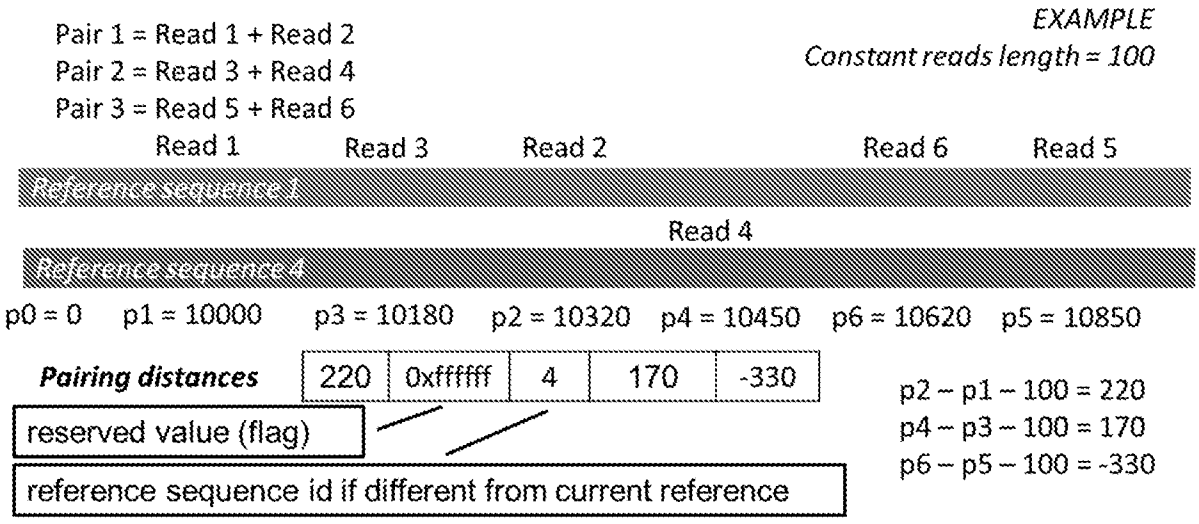


Figure 7

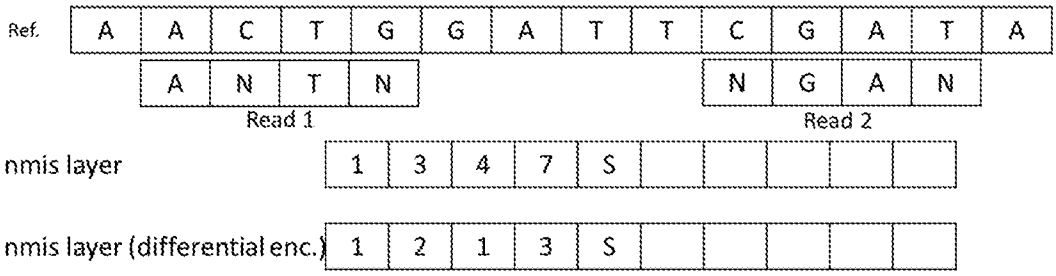


Figure 8

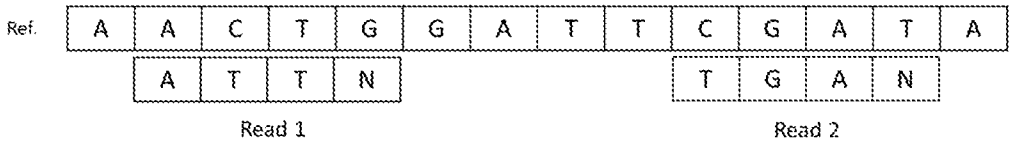


Figure 9

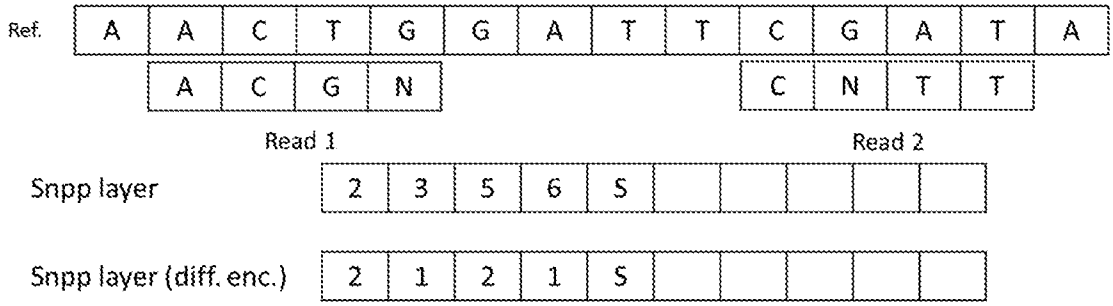


Figure 10

snpt Layer (without IUPAC codes)

A substitution type is calculated as index of a substitutions vector composed by all the possible symbols. For example:

S = [A, C, G, T, N, Z] ***where Z = deletion***

Index direction

- ENCODING is from right to left
- DECODING from left to right

Ref	Read	Encoded Symb.
A	del.	idx(A,Z) = 5
C	del.	idx(C,Z) = 4
G	del.	idx(G,Z) = 3
T	del.	idx(T,Z) = 2

Ref	Read	Encoded Symb.
N	A	idx(N,A) = 2
N	C	idx(N,C) = 3
N	G	idx(N,G) = 4
N	T	idx(N,T) = 5

Ref	Read	Encoded Symb.
A	C	idx(A,C) = 1
A	G	idx(A,G) = 2
A	T	idx(A,T) = 3
A	N	idx(A,N) = 4
C	A	idx(C,A) = 5
C	G	idx(C,G) = 1
C	T	idx(C,T) = 2
C	N	idx(C,N) = 3
G	A	idx(G,A) = 4
G	C	idx(G,C) = 5
G	T	idx(G,T) = 1
G	N	idx(G,N) = 2
T	A	idx(T,A) = 3
T	C	idx(T,C) = 4
T	G	idx(T,G) = 5
T	N	idx(T,N) = 1

Figure 11

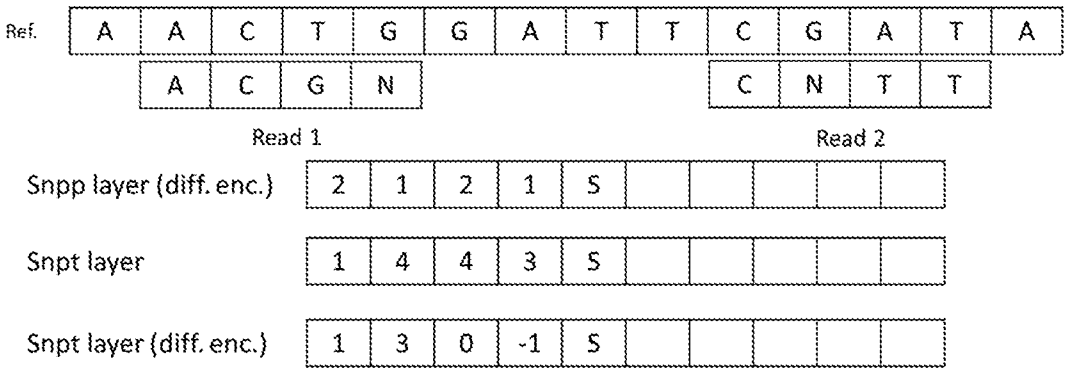


Figure 12

snpt Layer (with IUPAC codes)

A substitution type is calculated as index in a substitutions vector composed by all the possible symbols. For example:

S = [A, C, G, T, N, Z, M, R, W, S, Y, K, V, H, D, B]

Index direction

- ENCODING is from right to left
- DECODING from left to right

Ref	Read	Encoded Symb.
N	M	idx(N,M) = 2
N	W	idx(N,W) = 4
N	S	idx(N,S) = 5
N	B	idx(N,B) = 11

Ref	Read	Encoded Symb.
D	M	idx(D,M) = 8
A	Y	idx(A,Y) = 10
A	T	idx(A,T) = 3
A	N	idx(A,N) = 4
C	R	idx(C,R) = 6
C	G	idx(C,G) = 1
C	T	idx(C,T) = 2
C	W	idx(C,W) = 7
G	H	idx(G,H) = 11
G	C	idx(G,C) = 15
G	B	idx(G,B) = 13
G	N	idx(G,N) = 2
T	A	idx(T,A) = 13
T	M	idx(T,M) = 4
T	K	idx(T,K) = 8
T	V	idx(T,V) = 9

Figure 13

S = [A, C, G, T, N, Z, M, R, W, S, Y, K, V, H, D, B]

Direction

- ENCODING is from right to left
- DECODING from left to right

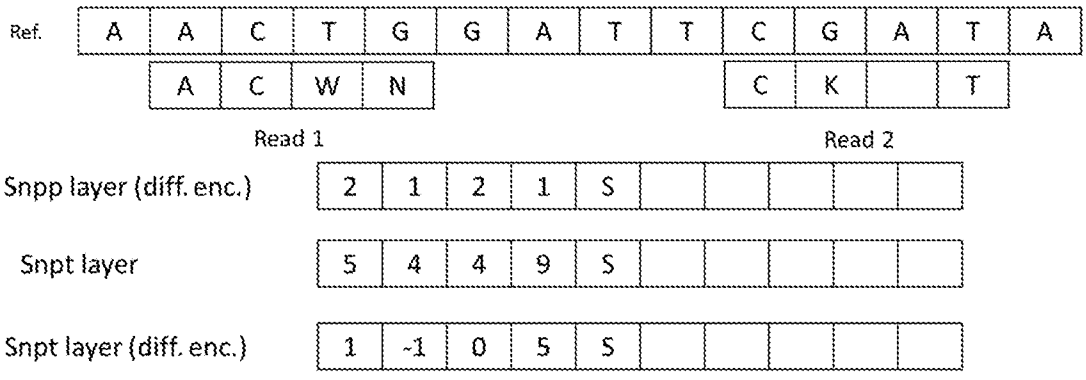


Figure 14

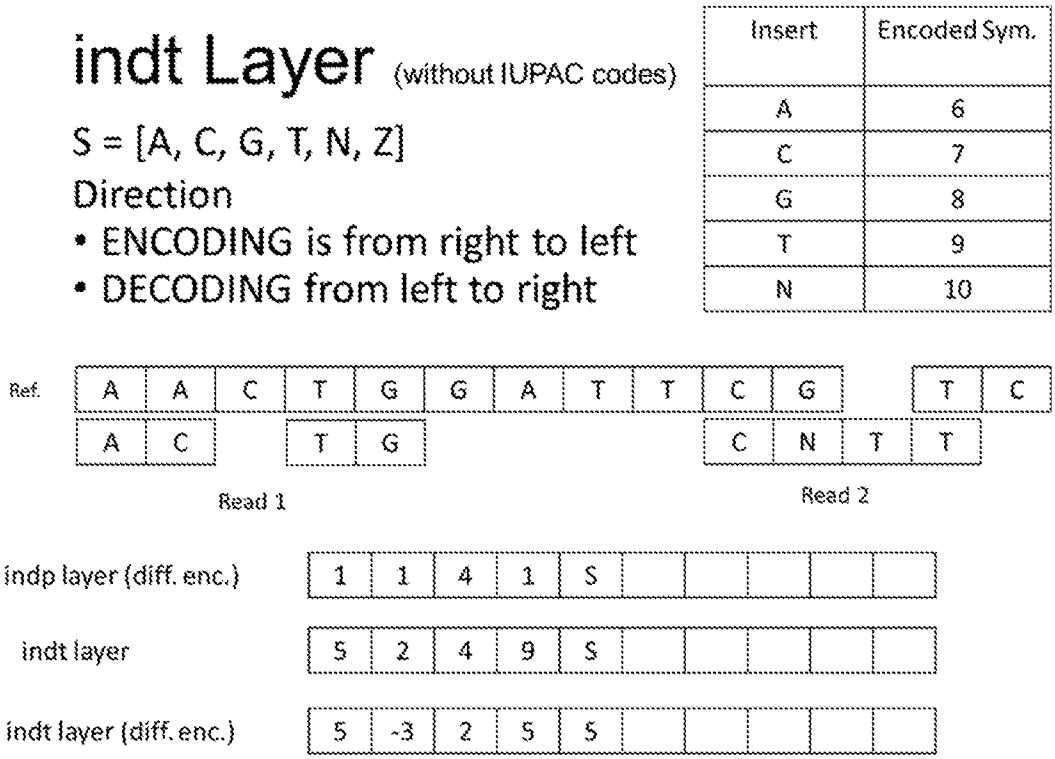

indt Layer (without IUPAC codes)
S = [A, C, G, T, N, Z]
Direction
• ENCODING is from right to left
• DECODING from left to right
Insert
Encoded Sym.
A
6
C
7
G
8
T
9
N
10
Ref.
A A C T G G A T T C G T C
A C T G C N T T
Read 1
Read 2
indp layer (diff. enc.)
1 1 4 1 S
indt layer
5 2 4 9 S
indt layer (diff. enc.)
5 -3 2 5 5

Figure 15

indt Layer (with IUPAC codes)

S = [A, C, G, T, N, Z, M, R, W, S, Y, K, V, H, D, B]
Direction
- ENCODING is from right to left
- DECODING from left to right

Insert	Encoded Sym.
A	16
C	17
G	18
T	19
N	20

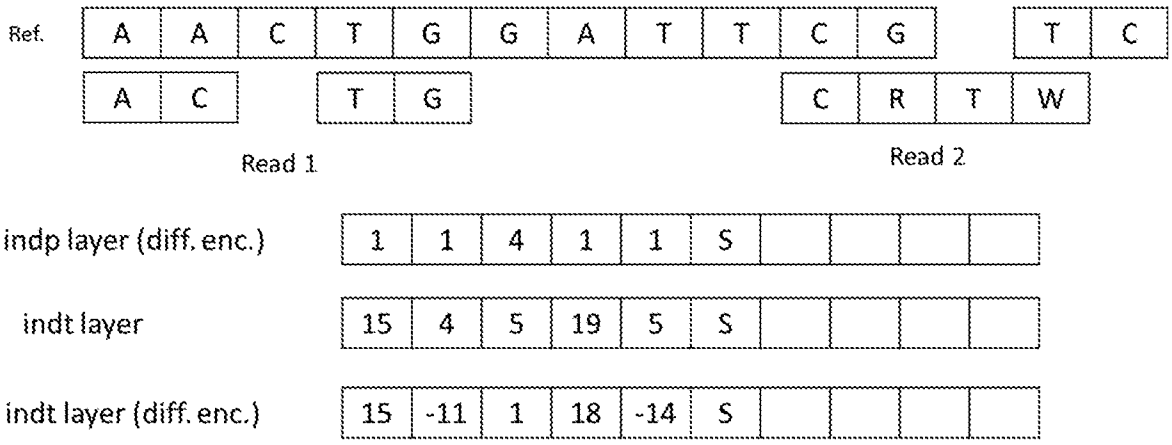


Figure 16

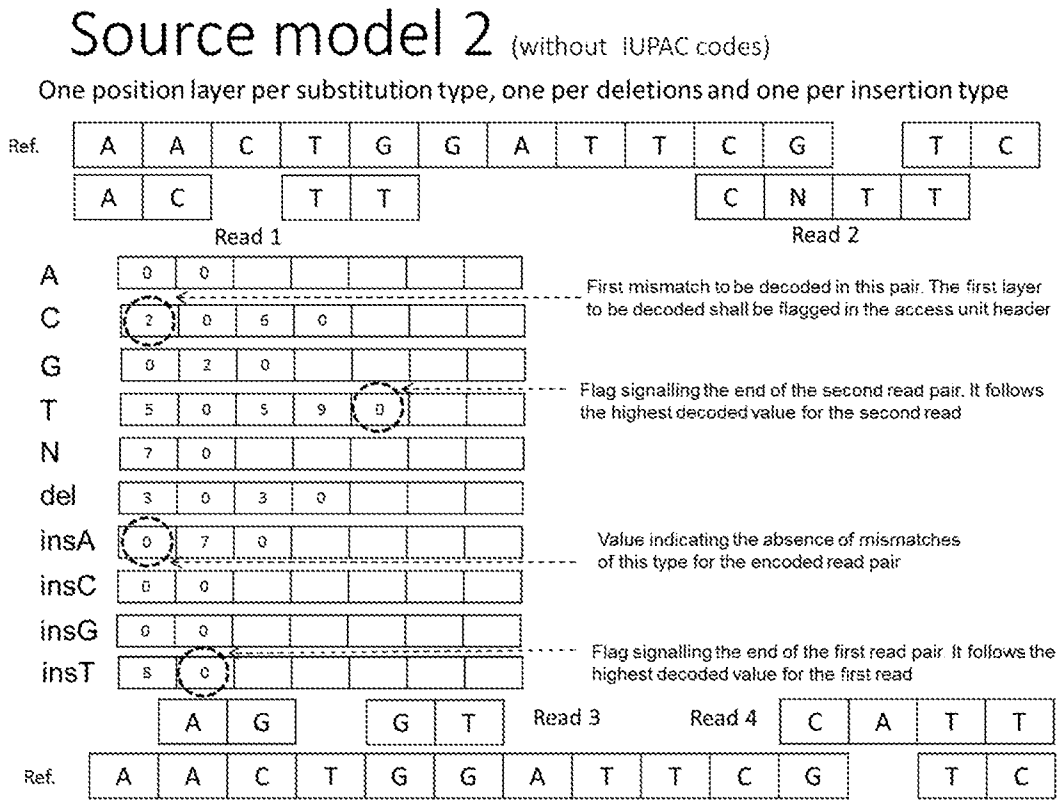

Source model 2 (without IUPAC codes)
One position layer per substitution type, one per deletions and one per insertion type
Ref.
A A C T G G A T T C G T C
A C T T C N T T
Read 1
Read 2
A
C
G
T
N
del
insA
insC
insG
insT
First mismatch to be decoded in this pair. The first layer to be decoded shall be flagged in the access unit header
Flag signalling the end of the second read pair. It follows the highest decoded value for the second read
Value indicating the absence of mismatches of this type for the encoded read pair
Flag signalling the end of the first read pair. It follows the highest decoded value for the first read
A G G T
Read 3
Read 4
C A T T
Ref.
A A C T G G A T T C G T C

Figure 17

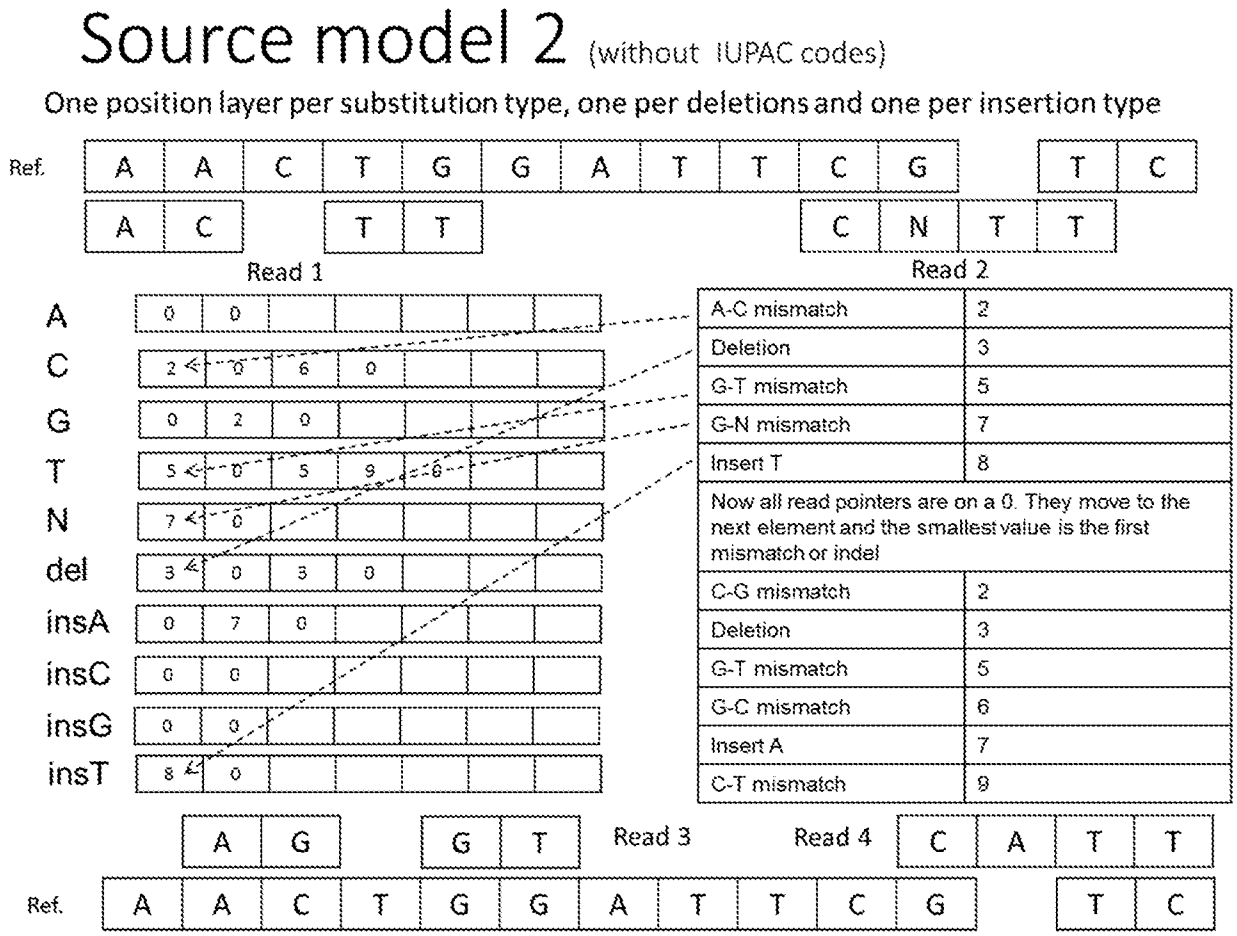

Source model 2 (without IUPAC codes)
One position layer per substitution type, one per deletions and one per insertion type
Ref.
A A C T G G A T T C G T C
A C T T C N T T
Read 1
Read 2
A
C
G
T
N
del
insA
insC
insG
insT
A-C mismatch 2
Deletion 3
G-T mismatch 5
G-N mismatch 7
Insert T 8
Now all read pointers are on a 0. They move to the next element and the smallest value is the first mismatch or indel
C-G mismatch 2
Deletion 3
G-T mismatch 5
G-C mismatch 6
Insert A 7
C-T mismatch 9
A G G T
Read 3
Read 4
C A T T
Ref.
A A C T G G A T T C G T C

Figure 18

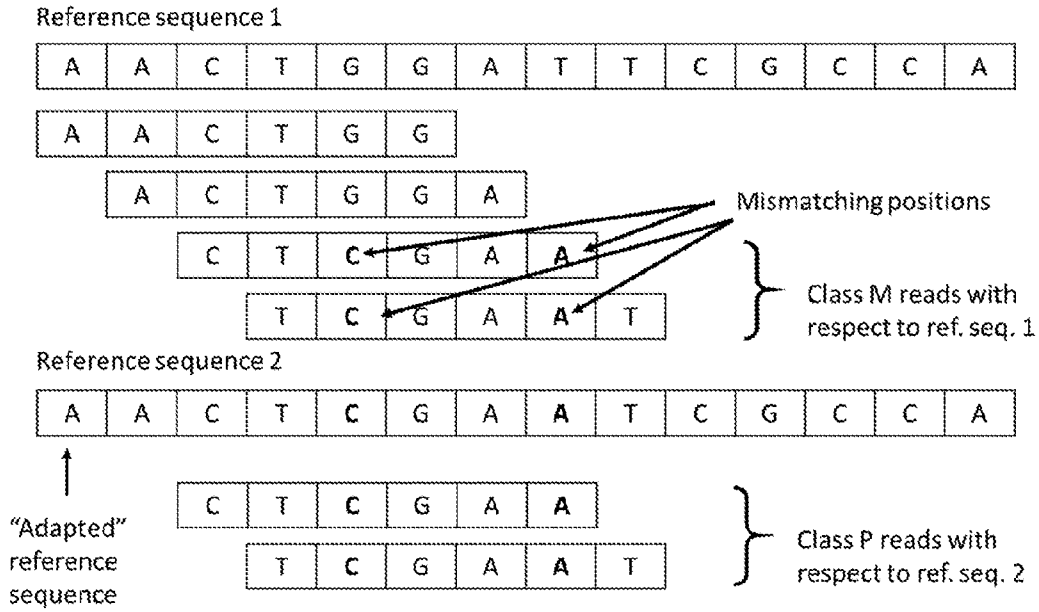


Figure 19

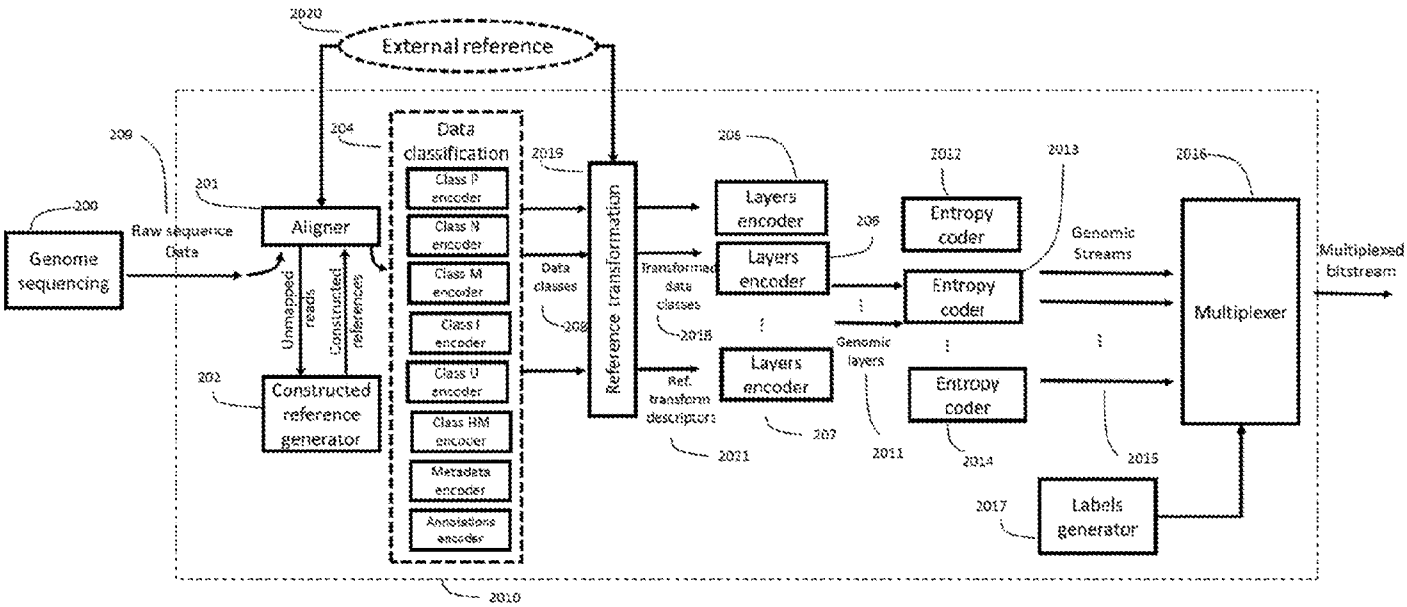


Figure 20

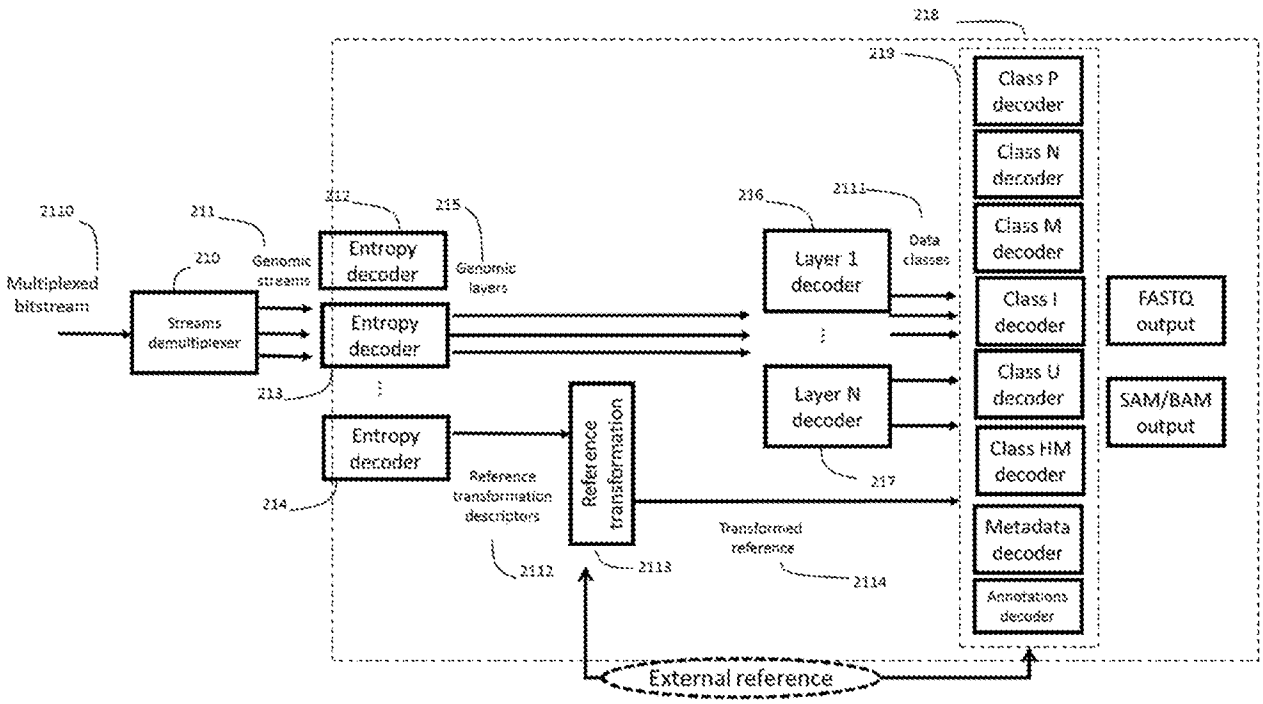


Figure 21

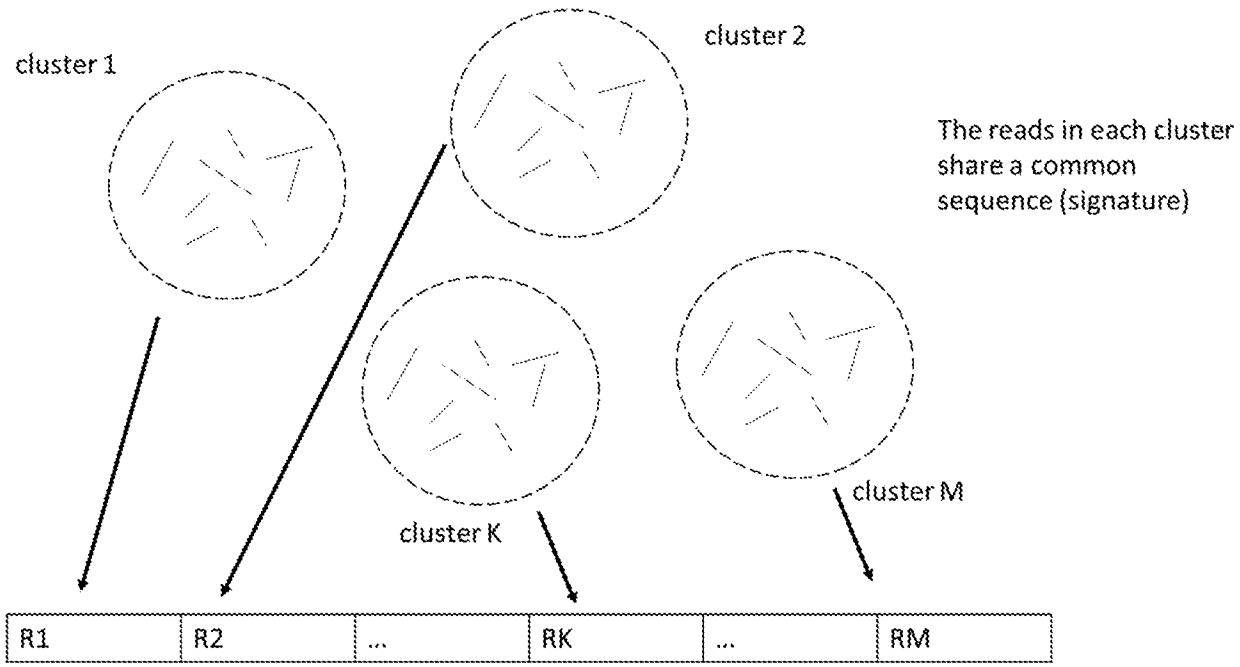
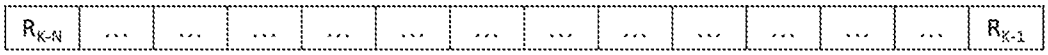


Figure 22



When encoding read K the reference is composed by reads K-1 to K-N
Appropriate padding is used when not enough reads are available.

Figure 23

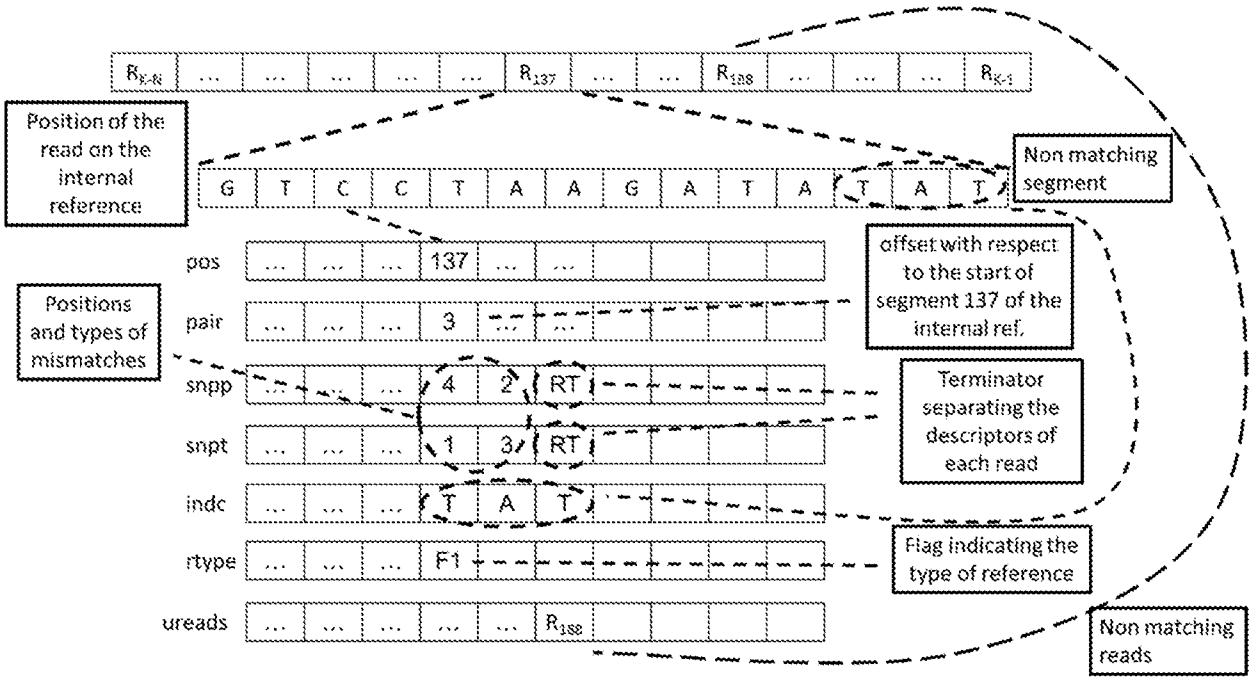


Figure 24

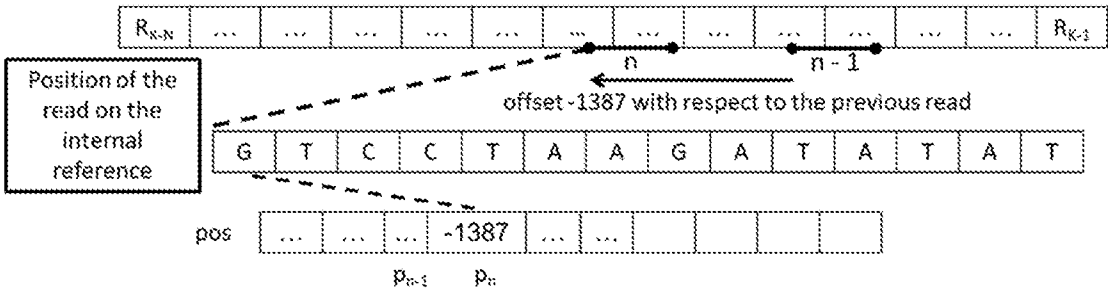
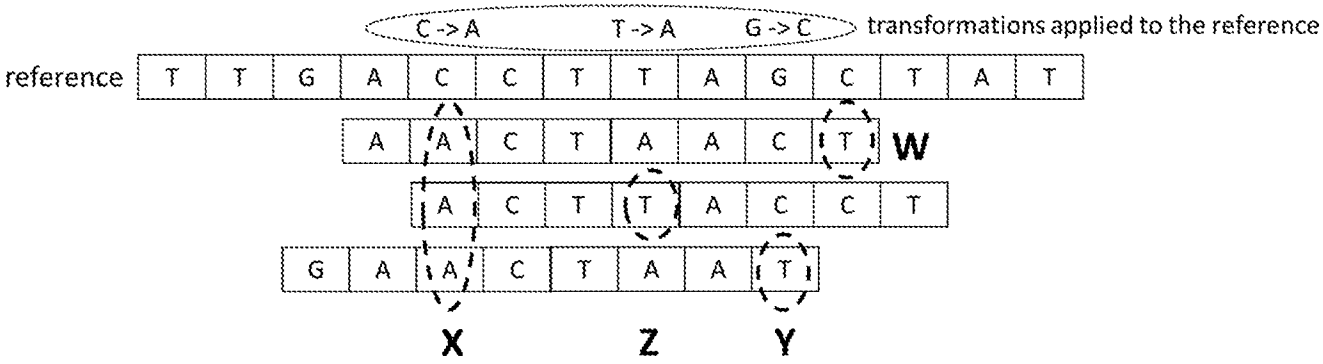


Figure 25



X = mismatch that will disappear
Y = mismatch that will not disappear but mismatch code shall change
W = mismatch that stays unchanged
Z = Newly introduced mismatch after transformation

Figure 26

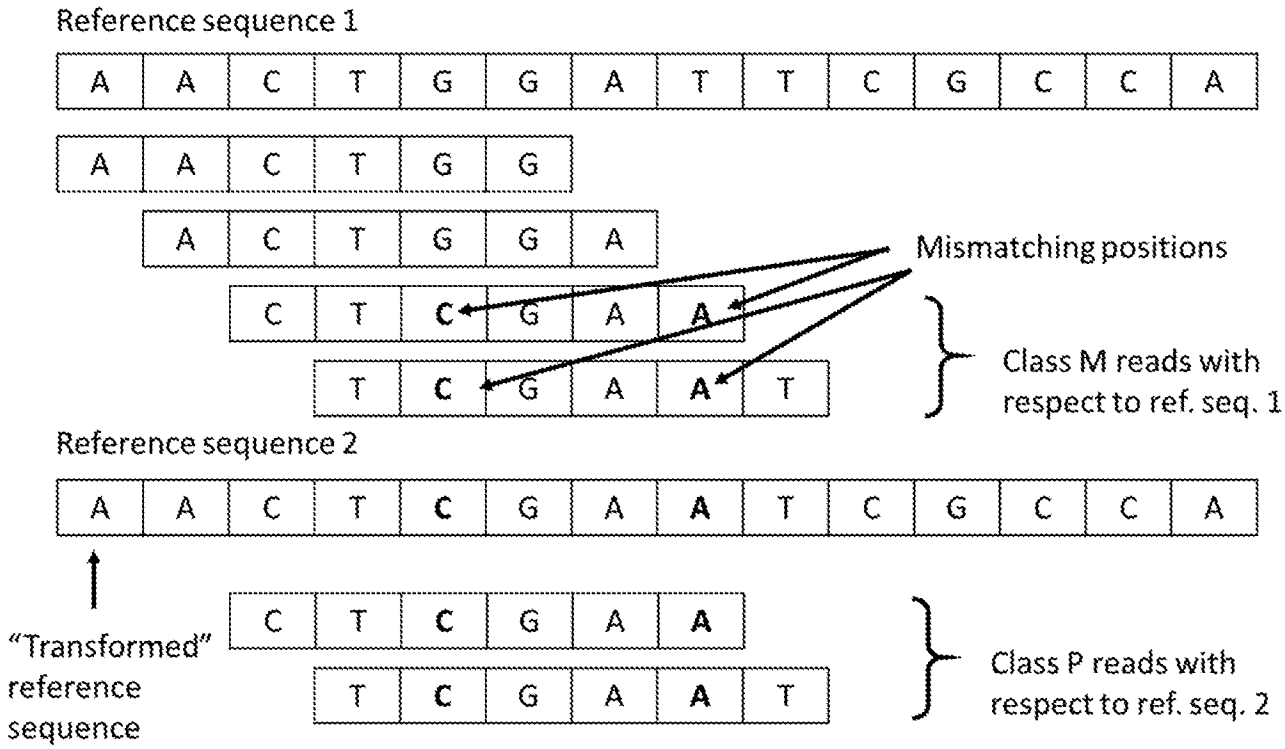


Figure 27

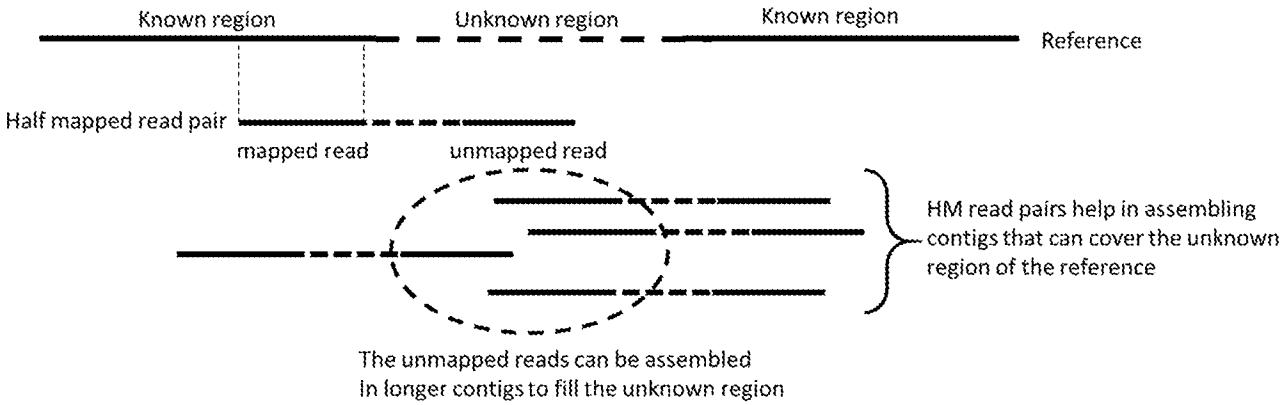


Figure 28

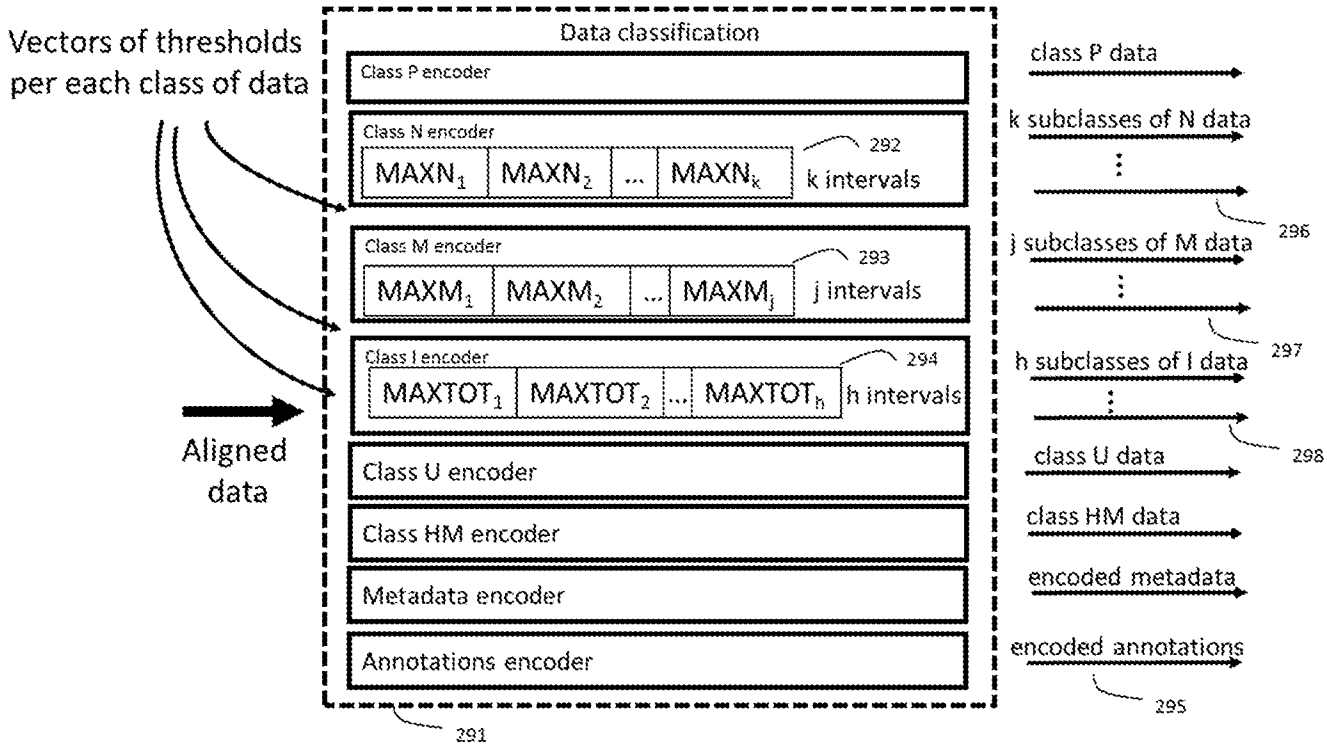


Figure 29

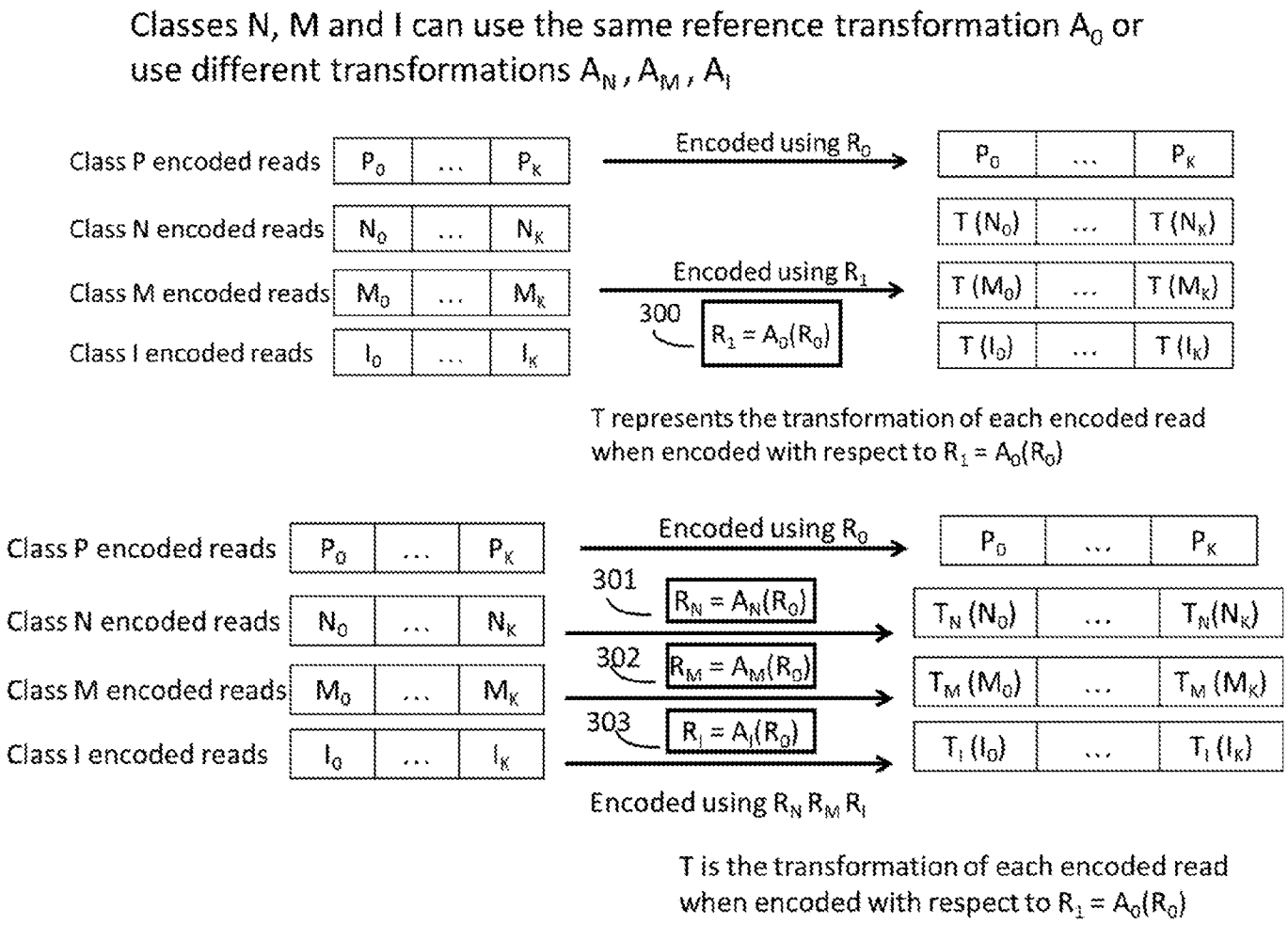


Figure 30

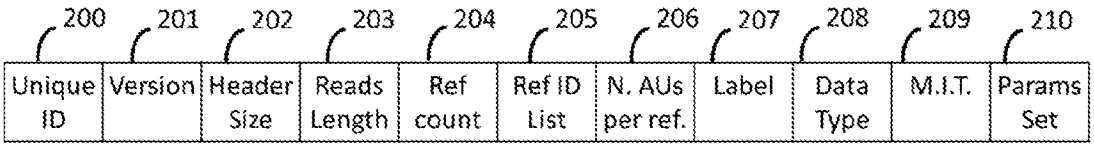


Figure 31

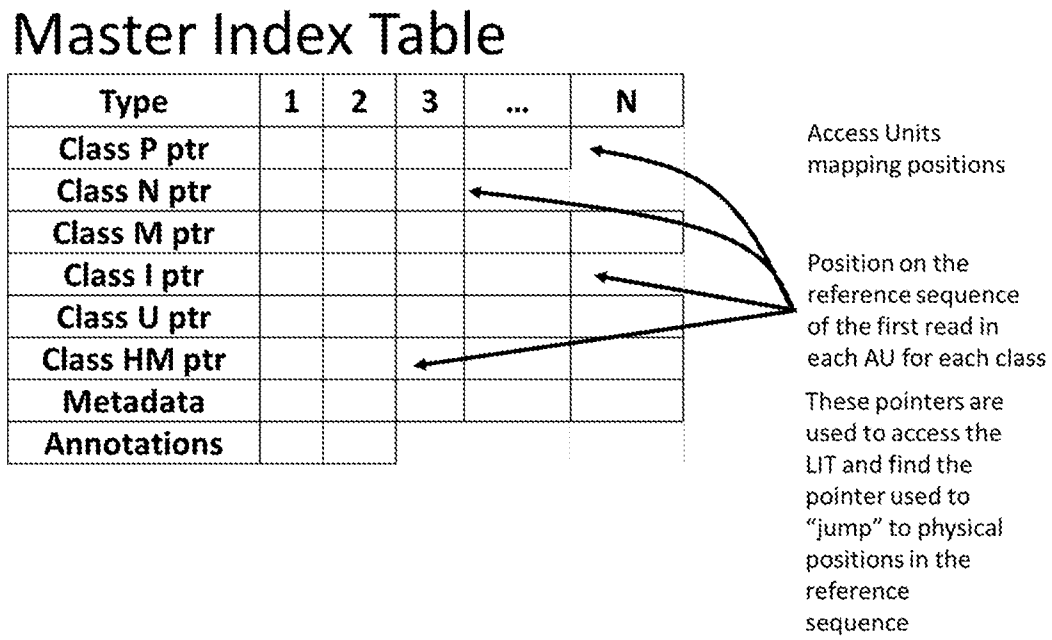


Figure 32

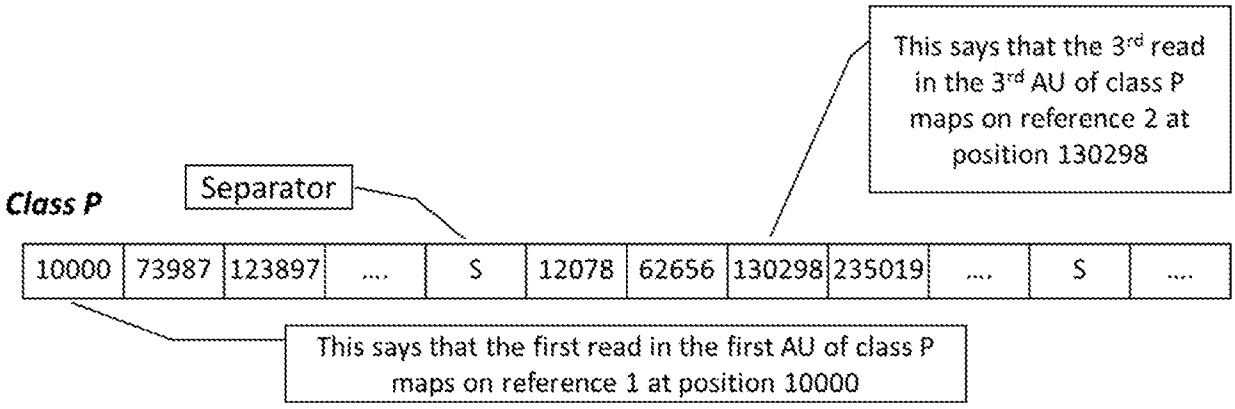


Figure 33

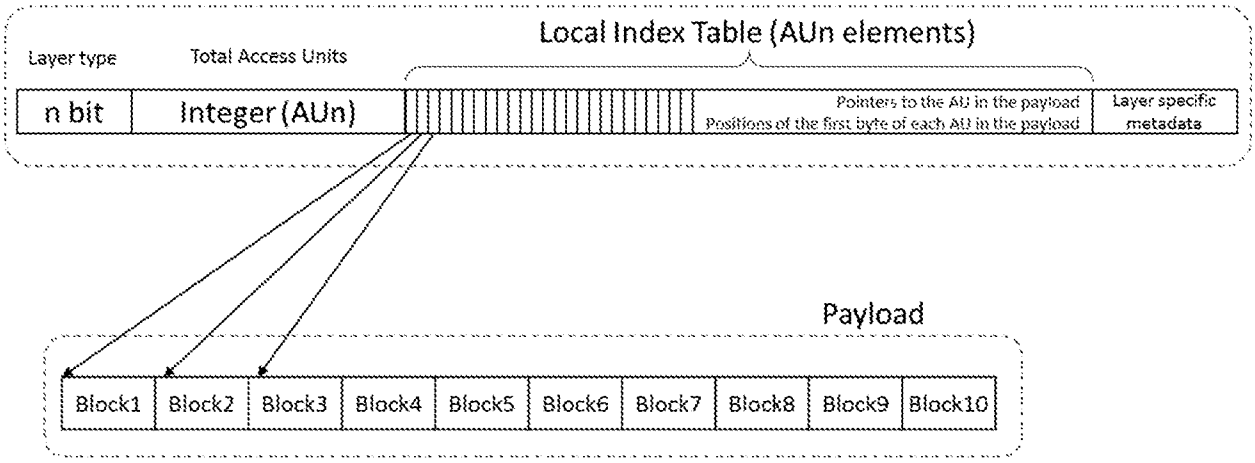


Figure 34

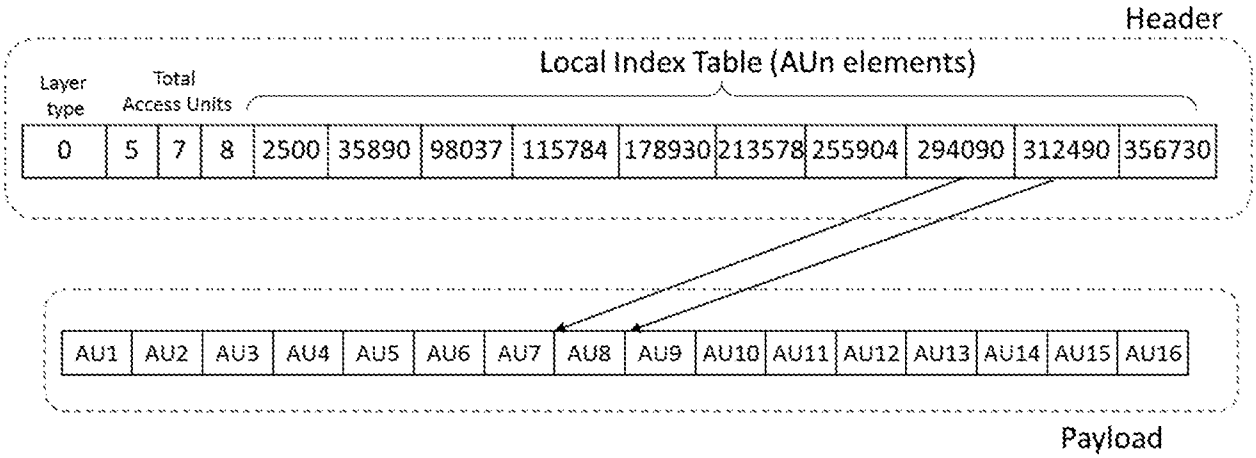


Figure 35

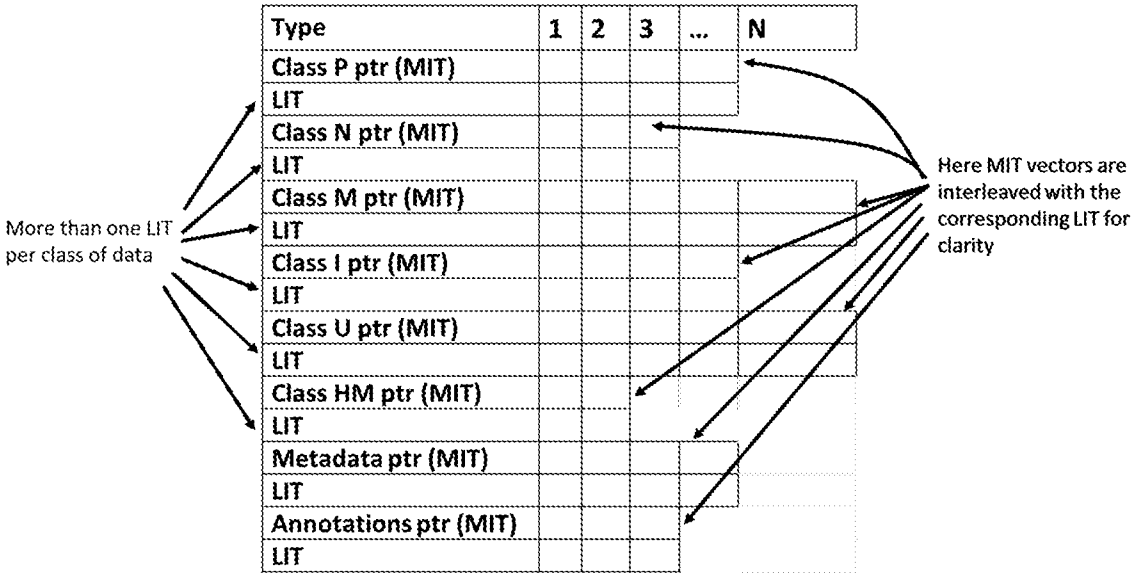

Type
1
2
3
...
N
Class P ptr (MIT)
LIT
Class N ptr (MIT)
LIT
Class M ptr (MIT)
LIT
Class I ptr (MIT)
LIT
Class U ptr (MIT)
LIT
Class HM ptr (MIT)
LIT
Metadata ptr (MIT)
LIT
Annotations ptr (MIT)
LIT
More than one LIT per class of data
Here MIT vectors are interleaved with the corresponding LIT for clarity

Figure 36

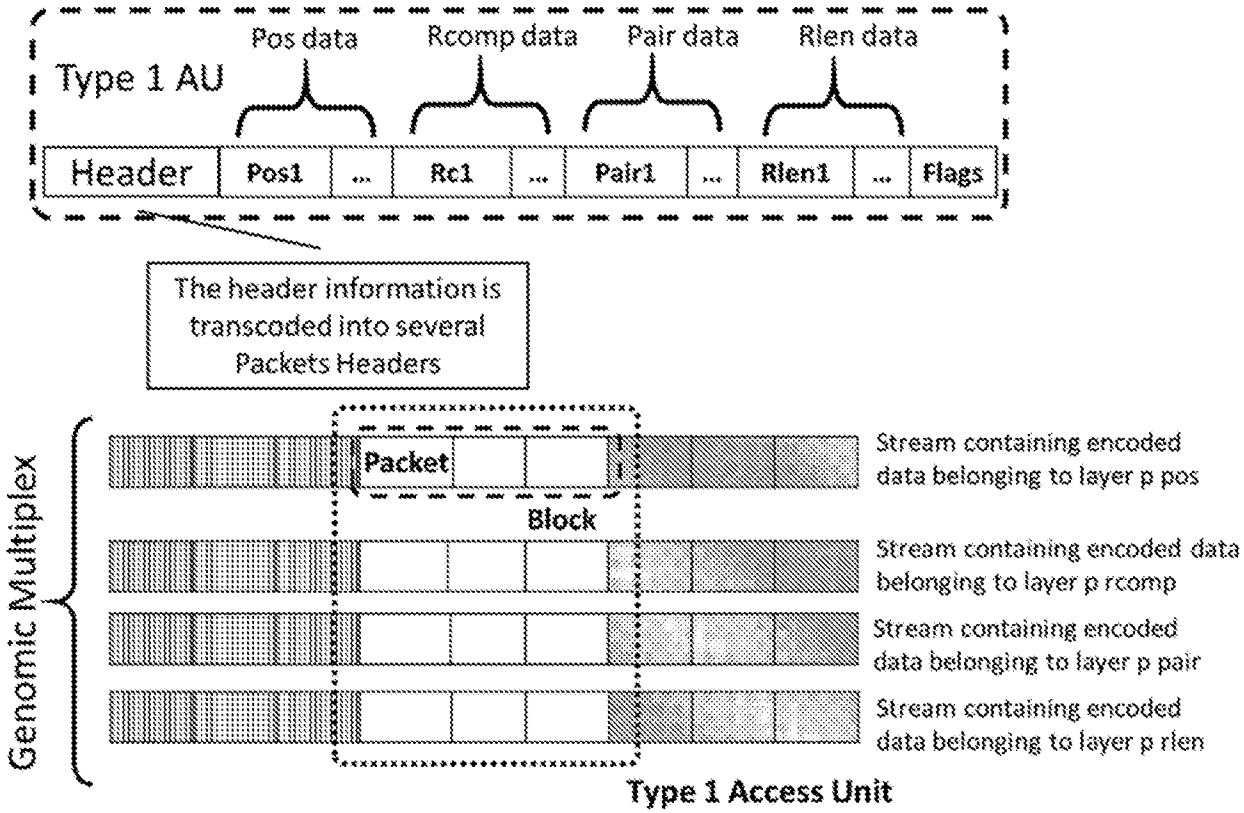


Figure 37

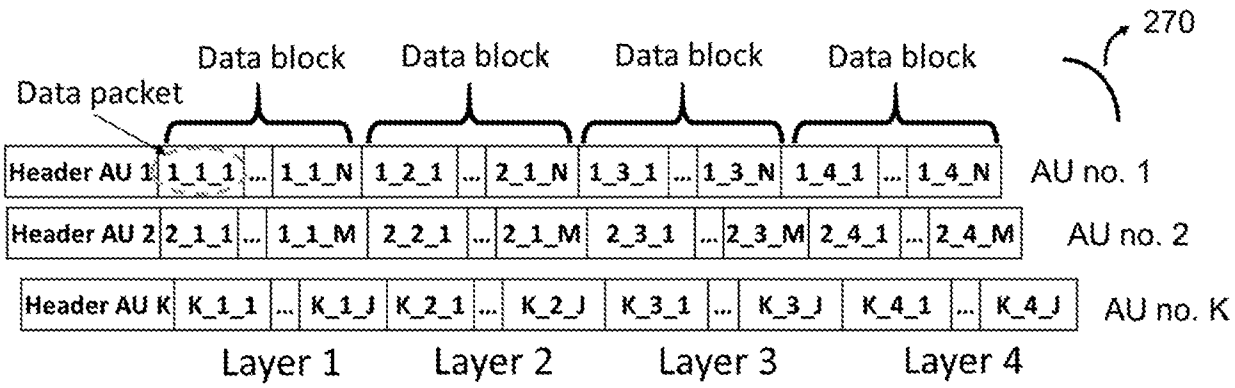


Figure 38

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 17/17842

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 7/00 (2017.01)
CPC - H03M 7/30, G06F 17/30067, G06F 17/30595, G06F 17/30961, H03M 7/3084

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History Document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History Document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History Document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2015/0227686 A1 (Sheinin et al.) 13 August 2015 (13.08.2015) entire document (especially para [0032], [0042]-[0046],	1-2, 21, 29-30, 32-33, 35-36
Y	US 2016/0191076 A1 (Leighton et al.) 30 June 2016 (30.06.2016) (para [0031], [0036], [0083])	1-2, 21, 29-30, 32-33, 35-35
Y	US 2013/0204851 A1 (Bhola et al.) 08 August 2013 (08.08.2013) (para [0009], [0078]-[0079], [0148]-[0149])	1-2, 21, 32, 35-36
Y	US 2012/0059670 A1 (Sanborn et al.) 08 March 2012 (08.03.2012) ([0044]-[0045], [0112], [0118])	2, 29-30, 32-33
Y	US 2014/0371109 A1 (McMillen et al.) 18 December 2014 (18.12.2014) (para [0068], [0257], [0269]-[0270], [0274], [0279], [0321], [0364])	29-30, 32-33, 35-36
Y	US 2004/0153255 A1 (Ahn) 05 August 2004 (05.08.2004) (para [0027], [0056], [0068]-[0070])	30, 33, 36
A	US 2008/0077607 A1 (Gatawood et al.) 27 March 2008 (27.03.2008) entire document	1-2, 21, 29-30, 32-33, 35-36

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 June 2017 (09.06.2017)	22 JUN 2017
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 17/17842

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 3-20, 22-28, 31, 34, 37-43
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.