GenSeg and MR-GenSeg: A novel segmentation algorithm and its parallel MapReduce based approach for identifying genomic regions with copy number variations

Rituparna Sinha, Rajat K. Pal, *Member, IEEE*, and Rajat K. De, *Senior Member, IEEE*

Abstract—Identifying intragenic as well as intergenic sequences of the DNA, having structural alterations, is a significantly important research area, since this may be the root cause of many neurological and autoimmune diseases, including cancer. Working with whole genome NGS data has provided a new insight in this regard, but has lead to huge explosion of data that is growing exponentially. Hence, the challenges lie in efficient means of storage and processing this big data. In this study, we have developed a novel segmentation algorithm, called GenSeg, and its parallel MapReduce based algorithm, called MR-GenSeg, for detecting copy number variations. In order to annotate CNVs (variants), segments formed by GenSeg/MR-GenSeg have been represented in a novel way using a binary tree, where each node is a CNV event. GenSeg considers each position specific data of whole genome DNA sequence, so that precise identification of breakpoints is possible. GenSeg/MR-GenSeg has been compared with twelve popular CNV detection algorithms, where it has outperformed the others in terms of sensitivity, and has achieved a good F-score value. MR-GenSeg has excelled in terms of SpeedUp, when compared with these algorithms. The effect of CNVs on immunoglobulin (IG) genes has also been analysed in this study.
Availability: The source codes are available at https://github.com/rituparna-sinha/MapReduce-GENSEG

Index Terms—CNV, NGS, WGS, Hadoop, MapReduce, Immunoglobulin genes.

1 Introduction

Next generation sequencing (NGS) technology [1] has become the leading platform for genome research associated with the study and analysis of copy number variations (CNVs) which are large alterations in the structure of DNA. The major reason for copy number variations, leading to complex diseases, lies in the fact that these variants can occur in the coding region, thus affecting the gene dosage. However, sometimes CNVs in non coding regions also might have some contributions in this regard. In addition, a recent study has revealed that copy number variations in the human globulin heavy chain locus lead to several infectious and autoimmune diseases [2].

NGS based variant detection algorithms make use of short reads generated by massively parallel sequencing technology, and are categorized into read pair, read depth (depth of coverage-DOC), and split read methods. Among them, DOC [3] is the most popularly used approach for variant detection. Some statistical/probabilistic model based methods include EWT (Event-wise Testing) [4], which focuses on a probabilistic model to determine genomic regions with copy number change; CNVeM [5], which works on nucleotide level data and uses expectation maximization method to detect CNV of individual samples; cnMOPS [6] and CMDs [7] work on multiple samples, where cnMOPS uses a mixture of Poisson model and CMDs is based on a between-chromosomal site correlation analysis and is used to detect recurrent copy number aberrations (RCNA); Another multiple sample based method DCC [8], obtains correlation coefficient of each genomic bin and computes the derivative of correlation coefficients and thereby finds significant derivatives to identify genome breakpoints. CNVkit [9] works with bias and GC corrected, on- and off-target bin level copy ratios, and associates a weight based on the size of the bin, the deviation of the bins log2 value in the reference from 0, and the spread of the bin in the reference.

CNV-CH [10] and CBSBR [11] are also multiple sample based CNV detection algorithms, where CNV-CH uses a convex hull based segmentation approach to detect genomic regions with abnormality, whereas CBSBR uses penalized sparse regression model using the read depth profiles. In addition, two more multi sample based algorithms, TrioCNV [12] and CanvasSPW [13], make use of a pedigree structure, and use an HMM based segmentation method to detect CNV regions. CanvasSPW, in addition to trios also processes quads, and uses EM algorithm to optimize the emission matrix.

A few very recent CNV detection tools include MSeq-CNV [14], which works with mate pair reads and uses a machine learning model based on mixture density and EM algorithm. SM-RCNV [15] is based on statistical model using frequency of variation at each location and correlation among locations across the genome. iCopyDAV

- *Rajat K. De is with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India. E-mail: rajat@isical.ac.in*
- *Rajat K. Pal is with the Department of Computer Science and Engineering, University of Calcutta, Kolkata, India.*
- *Rituparna Sinha is with the Department of Information Technology, Heritage Institute of Technology, Kolkata, India.*

[16] is another integrated platform that considers GC and mappability bias corrected RD values and performs divisive/agglomerative approaches of segmentation to detect and annotate CNVs. This method has utilized the multicore architecture in the segmentation procedure to make the process faster. DudeMl [17] uses the concept of sub window within a window with respect to a reference genome, and creates a vector of statistical parameters against sub windows, which is provided to train a random forest classifier.

All these algorithms work on NGS data, and suffer from high overhead associated with storage and processing of the massive volume of data generated through NGS technology [18] [19]. The massively parallel sequencing technology of NGS has resulted in terabytes of read data in a single execution, which is increasing exponentially [20]. Thus, the challenges to process and analyze this big data demand efficient tools and algorithms. Apache Hadoop [21] [22], the most popular distributed processing platform, can store terabytes and petabytes of data in its distributed file system, known as Hadoop Distributed File System (HDFS), where batch processing is performed under MapReduce [23] paradigm. In this regard, some of the existing tools include Genome Analysis Toolkit (GATK) [24], which is structured into traversals and walkers in order to separate accessing of the massive NGS data, from the logic specific to each analysis tool. Hadoop-BAM [25] acts as an integration layer between analysis applications and BAM files [26] stored in HDFS. These tools mainly focus on the input data access rather than the distributed variant calling approaches, which seek equal attention. HadoopCNV [27] is a scalable solution for CNV identification, which uses 32 node hadoop cluster to perform parallel processing, using both allele frequency and read depth data. However, it is unable to detect small size CNV, and is not precise in identification of CNV breakpoints.

In this study, we have developed a segmentation algorithm, called GenSeg, for detecting copy number variations in the nucleotide sequences of human DNA, based on NGS depth of coverage data. A MapReduce algorithm, called MR-GenSeg, based on GenSeg, has also been developed and implemented, under Hadoop framework, to overcome the overhead associated with processing whole genome NGS data. The majority of the CNV detection algorithms focus on target or exome sequence data. These data target at protein coding regions of the DNA, which constitute approximately 2% of the genome. As compared to whole genome sequence (WGS) data, whole exome sequence (WES) or target sequence data are cost effective, but suffer from many biases and noise, which makes WES unreliable for CNV detection [28]. Moreover, the noncontiguous nature of captured exons also makes it more error prone in detection of CNVs. All these limitations are not present in WGS data, and also mutations in introns or in both intergenic and intragenic sequences, which may affect the normal splicing of exons, are also not captured by WES. This has motivated GenSeg/MR-GenSeg to work with whole genome sequence high coverage data. Here we consider chromosomes 1, 14, 20, and 22 of two ethnic groups.

Most of the CNV detection algorithms work on a region/window based profile, where selection of the size of the window becomes an issue, and exact determination of boundaries of CNV events may not be possible. The present work considers read mapping information of each nucleotide position, where breakpoint of each small/large CNV is predicted at a high resolution. It may be mentioned here that DOC based methods generally consider read coverage, i.e., count of reads mapped to a reference location as an initial parameter. These counts suffer from many biases. Hence, in addition to normalisation, a second parameter tracking count of the number of reads that exactly match with the reference bases is also considered. Many previously used variant detection algorithms suffer from the presence of outliers, but GenSeg/MR-GenSeg is very sensitive in detecting the exact coordinates of the regions with outliers.

GenSeg has been compared with twelve popular CNV detection algorithms, where the present algorithm has outperformed the others in terms of sensitivity, having a value greater than 0.9, while specificity, precision, and F-score have been found to be in $[0.84\text{-}0.91]$. MR-GenSeg has outperformed GenSeg as well as other twelve algorithms with respect to SpeedUp, for very large data (i.e., whole genome DNA sequence data of *Homo sapiens*), where approximately 7 times SpeedUp is achieved as compared to some existing algorithms.

For analysing the variants obtained by GenSeg/MR-GenSeg, a novel tree based data structure has been implemented, where each node in the tree represents a CNV event (variant) with start and end coordinates. With the help of the designed data structure, common or rare variants are identified, with respect to different ethnic groups and diseases. Although antibodies have a major role in the central immune system, the DNA structural arrangements of immunoglobulin (IG) genes (coding for immunoglobulins) and the effect of CNVs on these IG genes are less explored. Thus, one of the objectives of the present work is to identify the effect of copy number variations in the human immunoglobulin locations, i.e., immunoglobulin (IG) locus in chromosomes 14 and 22 of the human genome. The mapping of the abnormal segments obtained by GenSeg from different samples, to IG gene sequences in chromosomes 14 and 22 has been obtained. It has been observed that some of the human immunoglobulin genes have been affected by a large region of duplication and deletion/insertion events.

To summarize, the contribution of the present work is three-fold: 1. Designing and implementing a novel CNV detection algorithm, working at nucleotide resolution, overcoming several bias and noise, and thereby determining precise breakpoint with a high sensitivity, and also handling outliers present throughout the dataset. 2. Developing a parallel MapReduce version of the same algorithm, to combat the overhead associated with NGS whole genome sequence data, and introducing scalability to allow analysis of each base of multiple human genome, efficiently with respect to execution time. 3. Developing a novel tree data structure to analyse each detected CNV, with respect to ethnicity and disease, and also analysing the effect of these CNVs on human Immunoglobulin genes.

2 Methodology

In this section, we develop a variant calling algorithm, called GenSeg, which considers the coverage and quantifi-

cation values of each base coordinate, and maps them to some statistical parameters; thereby forming $2D$ inputs to GenSeg. GenSeg forms genome wide segments, and also discovers the specific coordinates of the outliers, if any, in each segment. A parallel version of GenSeg, called Map-Reduce GenSeg (MR-GenSeg), has also been developed in this article to efficiently manage the time and overhead of processing a whole genome sequence. The detailed methodology of GenSeg involves the following tasks described under respective subsections.

2.1 Estimation of nucleotide coverage and quantification values, and their normalisation

Let us consider a set of reads generated from a sample human genome through NGS technology. These reads have been mapped to a standard reference sequence, thereby obtaining the optimal positions/coordinates where they have got aligned. A read may get aligned to multiple positions, but here we have considered one such position randomly. From this information we have generated the coverage value c_i for ith nucleotide/coordinate of the reference sequence. That is, c_i is the number of reads getting overlapped to a specific ith coordinate of the reference genome. Next, for each ith coordinate, the nucleotide quantification value q_i is calculated. Out of c_i reads mapped to ith coordinate of the reference sequence, q_i is the number of reads having nucleotide exactly identical to that of the ith base of the reference sequence. Thus, the output of this step is n nucleotide coverage values c_i and n nucleotide quantification values q_i, $1 \leq i \leq n$, where n is the length of the reference sequence. High throughput sequencing gets affected by several bias, due to which normalisation of c_i and q_i values is required as described below.

a) **GC adjustment:** GC bias affects the coverage values of the low and high GC regions of a genome [29]. In other words, there lies a dependence between coverage and GC content. This bias may lead to false variant call. Thus, c_i is adjusted/normalized as

$$c_i^{(N)} = c_i \times m_1 / m_{1i}^{(GC)}, \tag{1}$$

where m_1 is the median base coverage across all coordinates of the reference genome, and $m_{1i}^{(GC)}$ is the median base coverage of all those coordinates which have identical GC% to that of ith coordinate. GC% of each ith coordinate, for this purpose, is defined as

$$d_{1i}^{(GC)} = t_{1i}^{(GC)} / t_{1i}^{(ATCG)} \times 100 \tag{2}$$

Here $t_{1i}^{(GC)}$ is the total GC count of all the reads getting overlapped at ith coordinate, and $t_{1i}^{(ATCG)}$ is the total number of nucleotides in those reads. The term $m_{1i}^{(GC)}$ in Equation (1) is computed based on $d_{1i}^{(GC)}$ values. Similarly, $q_i^{(N)}$, normalised quantification value q_i, is defined as

$$q_i^{(N)} = q_i \times m_2 / m_{2i}^{(GC)}, \tag{3}$$

where m_2 is the median base quantification across all coordinates of the reference genome, and $m_{2i}^{(GC)}$ is the median quantification of all those coordinates which have identical GC% to that of ith coordinate. In this regard, GC% of each ith coordinate ($d_{2i}^{(GC)}$) is defined as

$$d_{2i}^{(GC)} = t_{2i}^{(GC)} / t_{2i}^{(ATCG)} \times 100 \tag{4}$$

Here $t_{2i}^{(GC)}$ is the total GC count of all those reads getting overlapped at ith coordinate, and having a nucleotide at that position exactly identical to ith coordinate of the reference. The term $t_{2i}^{(ATCG)}$ is the total number of nucleotides in those reads corresponding to ith coordinates. As in the case of $m_{1i}^{(GC)}$, $m_{2i}^{(GC)}$ value in Equation (3) is calculated using $d_{2i}^{(GC)}$ values.

b) **Generating standard score of adjusted coverage and quantification values:** Coverage and quantification values are further standardised with z-scores as described below. For coverage, z-score is defined as

$$z_i^{(cov)} = \frac{(c_i^{(N)} - \mu_{cov})}{\sigma_{cov}}, 1 \leq i \leq n \tag{5}$$

Here μ_{cov} is the mean of the $c_i^{(N)}$ values over all the coordinates and σ_{cov} is the standard deviation. Similarly, z-score for quantification is defined as

$$z_i^{(quan)} = \frac{(q_i^{(N)} - \mu_{quan})}{\sigma_{quan}}, 1 \leq i \leq n \tag{6}$$

The term μ_{quan} is the mean of the $q_i^{(N)}$ values obtained over all the coordinates and σ_{quan} is the standard deviation.

These scores also put data onto the same/smaller scale. Thus, each of the normalised values of nucleotide coverage and quantification, represented by $z^{(cov)}$ and $z^{(quan)}$, constitute a 2D point for a base. Thus, for each sample genome of length n, n such 2D points have been generated.

2.2 Generating genome wide segments

GenSeg considers the above n (2D) points as input to generate a set of genome wide segments. For this purpose, GenSeg starts with ith (initially $i = 1$) base/coordinate value and finds all such jth successive points whose Euclidean distance $(dist_{ij})$ from ith point is less than or equal to sim_i, where sim_i is defined as

$$sim_i = \sum_{j=i+1}^{i+minw} dist_{ij}/minw, 1 \leq i \leq n \tag{7}$$

The term $minw$ is a user defined constant representing the minimum number of consecutive coordinates/bases considered to form a segment. In this work, we are dealing with copy number variations (CNVs), which are large blocks of duplications or deletions of DNA segments, and are of the order from kilo-bases to mega-bases. Thus, we have set $minw$ in the interval [0.5, 1] kb. Here, ith coordinate is considered as the start of a segment and all successive neighbours become part of the segment.

The above process is continued for all successive points, until a dissimilar point is encountered. For a jth point with

$dist_{ij} > sim_i$, GenSeg compares the distance of that point with the existing members of the segment. If the dissimilar point becomes a neighbour of any one of the members in the segment starting at i, the point becomes a member of the segment. This process is continued, and we stop only when we get a point, which is dissimilar to ith point as well as dissimilar to all the existing points in the segments, originating at ith coordinate. In other words, a point is considered dissimilar, if and only if it is neither a neighbour of the point at ith coordinate nor a neighbour of any of the neighbours of ith point. If $(maxout + 1)$ such dissimilar points are obtained, we backtrack to the first encountered dissimilar point and mark the position as a breakpoint, i.e., the starting of another new segment. Here, $maxout$ is a user defined constant, representing maximum number of coordinates to be considered as outliers, and are set to a value in $[0.5, 1]$ percentage of $minw$. The same procedure is repeated to find the next breakpoint. If the number of successive dissimilar points is less than or equal to $maxout$, we consider these dissimilar points as outliers and continue forming the segment without break. The pseudocode of GenSeg is given in Algorithm 1.

GenSeg output a set of DNA segments, of which some segments are treated as normal segments and some other segments have been generated due to either duplication or deletion event. The algorithm has been executed several times altering the value of the parameter $minw$, thereby altering value of the parameter $maxcount$, within an interval already mentioned above. Execution of GenSeg multiple times is aimed at considering the minimisation of the objective function S (to form compact segments) defined as

$$S = \sum_{j=1}^{k} \sum_{i=1}^{n} (||\mathbf{x}_{ij} - \mathbf{c}_j||^2), \tag{8}$$

where k is the number of segments, n is the total number of data points, and $||\mathbf{x}_{ij} - \mathbf{c}_j||$ is the distance measured between a data point (x_{ij}) and the cluster centre (c_j).

Algorithm 1 GenSeg: Segmentation algorithm to detect DNA regions with CNV event

INPUT: Set of all $z_i^{(cov)}$ and $z_i^{(quan)}$ values, where i is each coordinate of the genome, forming the input data. $maxout$ - maximum number of coordinates considered to form outliers within a single segment. The value is set in the range [0.5, 1] percentage of $minw$.
OUTPUT: A list of segments of the whole genome sequence, which has been detected by CNV event.

```
for each ith coordinate, where 1 ≤ i < n do set outcount = 0 //i holds the coordinate of the start breakpoint of any segment, and outcount represents the number of consecutive outliers.
    for i + 1 ≤ j ≤ n, do
        if dist_ij ≤ sim_i then increment j by 1
        else
            Set track = j //track holds the position of the first encountered dissimilar point.
            for each kth coordinate, where i + 1 ≤ k ≤ j − 1 and dist_kj > sim_i do //comparing the acquired dissimilar point at j, with the existing members of the segment, originating at i.
                Increment k by 1.
            end for
            if k equals j and outcount ≤ maxout then
                Increment j and outcount by 1 //signifying the dissimilar point at j, may be an outlier, within the segment originating at i.
            else if k equals j and outcount > maxout then
                Backtrack to the first encountered dissimilar point, set (track − 1)th coordinate as a breakpoint of the segment originating at i. Set i = track; set j = 0 //start of a new segment.
            else
                Continue the segment originating at i, hence increment j by 1.
            end if
        end if
    end for
end for
```

2.3 Multiple genome analysis using a tree data structure

The execution of GenSeg on multiple samples have generated a set of potential variants, containing one or more functional genes, or intragenic or intergenic sequences which might affect the expression of genes. In order to have some insight into the variants, we have designed a novel tree data structure that has enabled us to find whether the target variant of a particular sample is common or rare.

Initially, a binary tree is formed with the set of segments (variants) obtained from multiple samples. Each node in the tree represents a segment (variant) associated with two coordinates- start and end. An interval/segment to be inserted into the tree is considered as a new node represented by [$n1$, $n2$], where $n1$ and $n2$ being start and end coordinates. The new node is compared with an existing node of the tree considered as a current node with coordinates [$x1$, $x2$]. The left subtree of a node contains only those nodes whose end coordinates are less than the start coordinate of the concerned node. The right subtree of a node contains only those nodes whose start coordinates are greater than the end coordinate of the node under consideration. Here, each node contains data: a) Sample-id, representing the sample from which the segment has been obtained; b) Left and Right pointers pointing to left and right subtrees, respectively; c) A flag indicating whether the current node corresponding to the segment [$x1$, $x2$] has an overlap with the new node. Initially, the value of the flag is zero and gets increased by one whenever a segment (new node) overlaps with the current node. Finally, the value of flag variable gives us the number of segments got overlapped with the current node; d) Two search key values representing the start and end coordinates of the segment; e) Minlength, indicating the length of overlapping between a pair of nodes. In order to insert a node representing a segment/interval, we start with the root node. A node corresponding to an interval to be inserted can lie in one of the six positions corresponding to the following six different cases. **Case 1:**

Completely left of current node with which the node to be inserted is compared; **Case2:** Completely right of current node with which the node to be inserted is compared; **Case 3:** Fully overlapped with the current node's interval; **Case 4:** Partially overlapped with the current node, such that start coordinate of the new node to be inserted is less than the start coordinate of the current node, but end coordinate of the new node lies between the start and end coordinates of the current node; **Case 5:** Partially overlapped with the current node, such that the end coordinate of the new node to be inserted is greater than the end coordinate of the current node, but start coordinate of the new node lies between the start and end coordinates of the current node; **Case 6:** Partially overlapped with the current node, such that start coordinate of the new node is less than the start coordinate of the current node and end coordinate of the new node is greater than the end coordinate of the current node.

In Case 1 or 2, the node gets inserted as the left or right subtree, respectively. For Case 3, we increment the flag variable. In Cases 4 and 5, where the node/interval has partly overlapped with the current node's interval, we break the interval such that the non-overlapping part is set as the interval of the new node, and the node is inserted either to the left or right subtree accordingly. For Case 6, we break the interval into three, such that one just fits into the interval of the current node, another one is on the left of the current node, and the third one lies on the right subtree of the current node. Thus two new nodes are created. A flag with value zero indicates that the interval is disjoint. The flag variable gets incremented (Case 4, 5, and 6) only when the new segment overlaps with the current segment by at least 5% (in bp) of the length of the current segment, tracked by Minlength parameter. The steps of inserting a node is described in Algorithm 2.

Analyses: The tree described above is a dynamic data structure; hence, a new sample data can be inserted into the tree without disturbing the existing one. The shape of the tree is different if the initial root is different. However, any rare segment, if any, can be identified by the tree, even if the shape of the tree gets changed with a different root. Based on a node's flag value, some observations are as follows: a) flag = 0 - variant is rare; b) flag = total number of samples - variant is common among population (all samples); c) flag = number of samples corresponding to an ethnicity - variant is common over an ethnic group; d) flag = other nonzero value - variant may be common in a particular disease (existing among the samples overlapping with target variant). The details of the analyses are provided in the Supplementary Material.

Algorithm 2 CreateTree: This algorithm generates the tree structure with the objective to find all common and rare segments among the tested samples.

INPUT: All the detected variants obtained by GenSeg on multiple samples.
OUTPUT: CNV events categorized as common or rare with respect to ethnicity or diseases.

For each new node (segment) to be inserted in the tree
if tree is empty **then**
    insert the new node as a root node.
else
    make the current node having coordinates [$x1$, $x2$] as the root node.
    while current node ! = NULL **do**
        if $n2 < x1$ **then** update parent node = current node and current node = root of the left subtree. Set track = 1.
        else if $n1 > x2$ **then** update parent node = current node and current node = root of right subtree. Set track = 2.
        else if $x1 <= n1 <= x2$ and $n2 >= x2$ **then** update new node = $[x2+1, n2]$. Also update parent node = current node and current node = root of right subtree and set track = 2. Increment parent node.flag by 1.
        else if $x1 <= n2 <= x2$ and $n1 <= x1$ **then** update new node = $[n1, x1-1]$. Also update parent node = current node and current node = root of left subtree and set track = 1. Increment parent node.flag by 1.
        else
            Create two new nodes with coordinates $[n1, x1-1]$ and $[x2+1, n2]$. For the first node, update parent node = current node and current node = root of the left subtree, and set track = 1, and for the other node, update parent node = current node and current node = root of the right subtree. Set track = 2 and increment parent node.flag by 1.
        end if
    end while
    if track = 1 **then** insert the new node as left child of parent node **else** as right child.
    end if
end if

2.4 Parallel MapReduce algorithm for segmentation

In order to overcome the computational overhead of GenSeg, a parallel version of GenSeg, called MR-GenSeg, under the MapReduce paradigm, has also been developed in this article. MR-GenSeg has a map phase and a reduce phase. The initial input to the map phase (Algorithm 3) is a set of $\langle key, value \rangle$ pairs with genomic coordinates of the base as *key*, and *value* being the combination of base coverage and base quantification values. After processing these $\langle key, value \rangle$ pairs, the output obtained are of three categories as follows.

1.) $\langle K1, V1 \rangle$, where the key $K1$ is the combination of start coordinate of a segment, and a *flag* (other than previous *flag*) being set to 'B'. $V1$ represents a value that comprises a list of points (coordinates of bases) present in the segment already formed. This category of $\langle key, value \rangle$ pairs represents all the segments lying on the borders, i.e., the flag 'B' in the key represents a border segment. In order to process a huge volume of data, MapReduce divides the work into a set of independent jobs running in parallel, and therefore, tracking of border segments is required for future processing.

2.) $\langle K2, V2 \rangle$, where the key, $K2$ is the start coordinate of a segment and *flag* being set to 'N', representing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2020.3000661, IEEE/ACM Transactions on Computational Biology and Bioinformatics

noise data. The value $V2$ represents coordinate of each noise point present in the segment. Thus, for m noise points in a segment, we get m $\langle key, value\rangle$ pairs. For all these $\langle key, value\rangle$ pairs, *key* remains the same, but *value* represents the coordinates of the outliers present within the segment, which starts at *key.start*.

3.) $\langle K3, V3\rangle$ corresponds to the segments, where $K3$ is the start coordinate of a segment along with a *flag* being set to 'Y', indicating a significant segment. A significant segment is formed if the total number of coordinates is greater than $minw$. $V3$ indicates the end coordinate of the segment.

Algorithm 3 MR-GenSeg - Map task: In this phase, segmentation is performed and three categories of $\langle key, value\rangle$ pairs are obtained. One of them represents all border segments, second one representing all the remaining abnormal segments, and the third category represents outliers.

INPUT: A set of $\langle key, value\rangle$ pairs of the form $\langle coordinate, basecoverage, basequantification\rangle$, where *key = coordinate* and *value = base coverage, base quantification*. Each of these $\langle key, value\rangle$ is represented by K.
OUTPUT: Set of $\langle key, value\rangle$ pairs $\langle K1, V1\rangle$, $\langle K2, V2\rangle$ and $\langle K3, V3\rangle$ representing key values with respect to border segments, noise data, and segment with CNV event respectively.

1: Set $start$ = min(K.coordinate) and $stop$ = max(K.coordinate)
2: **for** $i \leftarrow start$ to $stop$, $ct \leftarrow 1$, $d \leftarrow 0$, $outcount \leftarrow 0$, $count \leftarrow 0$ **do**
3: **for** $j \leftarrow i+1$ to $stop$, **do**
4: **if** breakpoint at $K[j]$.coordinate is achieved **then**
5: if $i = start$, generate key value pair $\langle K1, V1\rangle$ of the form $\langle K[i].coordinate, B, a\rangle$ where key $\leftarrow \langle K[i].coordinate, B\rangle$ and value $\leftarrow \langle a\rangle$, and a is a list of points in the segment formed from i to $j-1$, set i to j and break. Here B represents border segments.
6: if $i! = start$, generate key value pairs $\langle K3, V3\rangle$ of the form $\langle K[i].coordinate, Y, K[j-1].coordinate\rangle$, where key $\leftarrow \langle K[i].coordinate, Y\rangle$ and value $\leftarrow \langle K[j-1].coordinate\rangle$.
7: **end if**
8: **end for**
9: **if** j is $stop$ + 1 **then**
10: Generate a key value pair $\langle K1, V1\rangle$ of the form $\langle K[i].coordinate, B, a\rangle$, where key = $K[i]$.coordinate, B and value $\leftarrow \langle a\rangle$, where a is a list of points in the segment formed from i to $j-1$. Break from loop.
11: **end if**
12: **end for**
13: Generate key value pairs $\langle K2, V2\rangle$ of the form $\langle K[i].coordinate, N, K[x].coordinate\rangle$, where key $\leftarrow \langle K[i].coordinate, N\rangle$ and value $\leftarrow \langle K[x].coordinate\rangle$ for each noise point x in the segment.

The map phase output are accumulated from the mapper processes executing in parallel, and are grouped and sorted according to their key values. These sorted output are provided to the reduce phase as input. Since map phase works on input splits, processing of segments lying on the border is required, which is done in the reduce phase of MR-GenSeg. For any two border segments obtained from different mapper processes, the reduce task first checks whether the segments are consecutive, and then a consecutive checking is done to find out whether the points constituting two segments are similar. If both the conditions are satisfied, these two border segments are merged. The output of reduce phase are all $\langle key, value\rangle$ pairs of type 'Y' and 'N', and all the border segments, which either get merged, or remain unchanged. In the reduce phase, the outliers within border segments are also tracked. The procedure of reduce phase is given in Algorithm 4.

Algorithm 4 MR-GenSeg - Reduce task: In this phase processing of all the border segments is executed.

INPUT: Sorted set of $\langle key, value\rangle$ pairs obtained from mapper.
OUTPUT: $\langle key, value\rangle$ pairs of type 'Y' or 'N', and all the border segments, which either gets merged, or remain unchanged. Here 'N', represents noise data and 'Y', represents a significant segment with CNV.

1: Initialise list L.
2: Read next key value pair $\langle K, V\rangle$, where key = coordinate, type, if present, else go to step 12.
3: **if** type is B **then**
4: value $\leftarrow a$ where a is the list of elements in a border segment.
5: Append a to list L. Go to step 2.
6: **else**
7: **if** type is Y or type is N **then**
8: perform segmentation on L (if non-empty) as described in MR-GenSeg - Map phase.
9: Empty list L. Go to step 2.
10: **end if**
11: **end if**
12: Perform segmentation on L if non-empty.

3 Results and Discussions

All NGS data (obtained under Illumina platform) used in this study are binary alignment ('.bam') files and have been obtained from 1000 genomes project via the FTP site (ftp://ftp.1000genomes.ebi.ac.uk/) hosted at EBI. The samples are of chromosomes 1, 14, 20, and 22 of *Homo sapiens*, where the sample Cell-Type is B-Lymphocyte and Tissue Type is Blood. The samples considered are two family trios, the details of which are provided in the Supplementary Material.

3.1 Analysis of simulated data

In this work, a human DNA reference sequence of chromosome H (=1, 4, 20, and 22) has been considered. All unknown bases represented by N's have been removed. The parameters considered for different experimental conditions are as follows: a) The length L (in bp) of the random region selected from the reference sequence is considered as 5000, 100000, 600000 and 4000000; b) The number (n) of samples considered is 100; c) Position (P) represents the position

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2020.3000661, IEEE/ACM Transactions on Computational Biology and Bioinformatics

or coordinate at which a CNV has to be introduced. The value is randomly chosen in 100 trials; d) Variant length (S) is the size (in bp) of each segment having CNV. Its value has been set in the interval $[50, 5000]$; e) Copy number (C) represents the number of times a segment of length S has been duplicated or deleted. The value of C is set to 0, 1, 3, and 4; f) H represents the chromosome number of the reference sequence, whose values are 1, 14, 20, and 22; g) Coverage G, representing the number of times a particular base has been covered by independent reads, where the value of C has been considered as 2X, 10X, 15X, and 30X.

Five random regions each of length L has been selected. In order to simulate a diploid genome, each region has been duplicated to get a region of length $2L$, which has acted as the reference sequence. Now, n test samples have been generated from the above diploid genome by the following approach. To create a test sample, firstly we have generated a copy of the reference genome and then introduced copy number variation at a randomly selected position P. In order to simulate a duplication event, a region from P to $P+S$ from the reference sequence has been chosen and duplicated $C-2$ times representing a copy number $C > 2$. In this approach, the decision to implant a copy number variation has been made with a coin tossing event. To simulate a deletion event, a region from position P to $P+S$ has been selected from the previous human DNA reference sequence of chromosome H, and has been inserted at the same position P and position $P+L$ of the generated reference sequence. Thus, the newly inserted sequence is only present in the reference and absent in the test sample, which has simulated a deletion event with copy number 0. In order to simulate a deletion event with copy number 1, the above region has been inserted in the reference sequence at position P and position $P+L$, as well as in the sample at position P or position $P+L$.

Here each experimental condition represents the combination of values of the parameters listed above. Simulating NGS Illumina technology, reads of size 30 bp have been generated from each test sample. Reads have been generated using a sampling process with probability p. The value of p has been determined from a unimodal distribution function, obtained using median GC-count of all possible 30 mers of respective chromosomes. Sampling has been continued till a particular coverage is obtained. This has introduced GC bias in the read generation process. Next, reads have been aligned to the reference sequence using a standard alignment algorithm, and normalised base coverage and quantification have been provided as input to algorithm GenSeg.

An instance of the base coverage values with respect to reference sequence of length 5000 bp and 100000 bp, are represented in Fig. 1A) and Fig. 1B) respectively. The coverage considered here is 15X. At the location with coordinate 500, a duplication of length 3500 bp has been introduced (in Fig. 1A)), whereas two duplication events and two deletion events have been introduced at different coordinates of the reference sequence (in Fig. 1B)). Here, the mean base coverage of the deletion events is 1.45, whereas the mean base coverage of the duplication events is 98.86. The length of duplication events are 11000 bp and 9000 bp, whereas deletion events are of length 13000 bp and 8000 bp. Figure 1C) represent the base quantification obtained with respect to 4000000 bp reference sequence having 30X coverage. Two duplication events of length 3000 bp and 700000 bp, and one large deletion event of length 1000000 bp have been introduced. The mean base coverage of the deletion events is 2.9, whereas the mean base coverage of the duplication events is 210.

MR-GenSeg has also been executed on the above simulated data. It has been observed that the performance of MR-GenSeg has not improved remarkably for small size data, since parallel execution introduces some overhead. However, as data size increases, its performance has improved much. Figure 1D) represents the execution time of GenSeg and MR-GenSeg. For an instance where reference sequence length is 5000 bp, it is observed that GenSeg took 5.72 seconds, whereas MR-GenSeg has taken 11.13 seconds. However, for reference sequences of length 100000 bp, 600000 bp, and 4000000 bp respectively, the performance of MR-GenSeg has improved significantly.

3.2 Analysis of real data

GenSeg and MR-GenSeg have been executed using NGS whole genome DNA sequence (WGS) high coverage data involving chromosomes 1, 14, 20, and 22. The motivation behind considering these chromosomes lies in the fact that the human immunoglobulin (IG) genes are present in the loci $14q32.33$, $2p11.2$, and $22q11.2$ of the human genome. IG genes code for immunoglobulins that are antigen receptors [2]. The effect of copy number variation in the IG locations is not well explored, in spite of the major role of IG genes in the central immune system. Moreover, since in this study a parallel implementation of GenSeg is executed, chromosome 1, being the largest human chromosome, has been considered. MR-GenSeg has been executed on a Hadoop cluster with one master and three data nodes.

For variation in the input over ethnicity, one may refer to Fig. 2A) and Fig. 2B). The figures represent the base coverage values of the samples corresponding to Utah (CEU) and Yoruba (YRI) groups. For the same samples, the distribution of base quantification values are represented using a box plot in Fig. 3A). It has been observed that 75% of the values are less than 60, and the values above the upper whisker are very high. It is also observed from the figures that the difference between lower quartile and upper quartile is very small, indicating that majority of the data are not very dispersed. For chromosome 1 and chromosome 20, the mean base quantification values have been found to be approximately 50 and 25 respectively, and 75% of the position specific coverage obtained has been 150 and 111 respectively, with the highest values of 8000 and 3000 respectively. However, in sample NA19240, the highest coverage has been found to be as low as 300.

After execution of GenSeg, we have obtained a list of variants for each sample. The parameter $minw$ has been set to a value in the interval [500, 1000], i.e., a segment containing at least 500 base pairs has been considered, although segments as small as 50 bp can also get well detected by GenSeg. The parameter $maxout$, i.e., maximum number of coordinates considered to form outliers in a single segment, has been set to a value in $[0.5, 1]$ % of $minw$. Each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2020.3000661, IEEE/ACM Transactions on Computational Biology and Bioinformatics

execution of the algorithm has a new parameter setting, and the objective function as represented in Equation (8), is evaluated. It has been observed that the segments obtained using NA12878 have a high number of duplication events, and other samples in different chromosomes generally have many duplication events with a few deletion events. However, samples NA19240 and NA19238 belonging to YRI family trio generally possess deletion events in chromosome 14, with a few duplication events present in the parent (NA19238) in the beginning of the chromosome. NA12878 has very large duplication events in chromosome 1 and chromosome 22. GenSeg is able to detect CNVs of varying sizes, as reflected in Fig. 3B). Both small and large CNVs have been detected by the present algorithm. However, most of the CNVs have been detected in the range of 2kb–500kb. In NA12878, NA12891, and NA12892, the majority of the variants detected is of size (2kb–200kb), (0.5kb–1kb), and (201kb–500kb) respectively.

The results obtained after execution of GenSeg/MR-GenSeg on the data sets have been validated with the Database of Genomic Variants (DGV). We have observed that segments identified as variants by the present algorithms have overlapped the most with the variants listed in the database, compared with the other algorithms. Other than this, the algorithms have found many novel variants that are not listed in the DGV. The novel variants detected by the algorithms, in the region [16000000, 17000000] of chromosome 22 of sample NA12878, have been listed in Table 1. The detected variants have mostly overlapped with EWT. Some rare variants also overlapped with CNV-CH, CNVeM, cnMOPS, DudeML and iCopyDAV.

TABLE 1
The novel variants obtained by GenSeg/MR-GenSeg in chromosome 22 of sample NA12878 in the region [16000000, 17000000]. The Length column represents the size of each detected variant and the last column represents the methods with which the detected variants have overlapped.

Start Coordinate	End Coordinate	Length (bp)	Overlap
16303178	16304610	1432	EWT, CNV-CH
16390484	16391954	1470	EWT, CNV-CH, DudeML,
16697840	16732622	34782	EWT, cnMOPS, CNVeM
16736689	16738068	1379	EWT, CNVeM,
16742135	16847840	105705	cnMOPS, CNV-CH, EWT, DudeML, iCopyDAV

TABLE 2
The common and rare variants obtained by execution of GenSeg on chromosome 14 of all samples in the region 19312049–32547087. The first two columns denote the start and end coordinates of a variant. The third column represents the samples containing the variants, and also characterizes the variant as rare or common.

Start Coordinate	End Coordinate	Samples
19312049	19313144	NA12878 (Rare)
20281460	20282642	NA19238 (Rare)
20374993	20389257	All Samples (Common)
20213344	20214522	Common in YRI Family
20647501	20880854	NA12878 (Rare)
21175287	21679062	NA12878 (Rare)
22343408	23685190	NA12878 (Rare)
26590957	26592389	All Samples (Common)

In this work, the novel data structure designed has been applied to analyse the detected variants. Table 2 considers a region [19312049, 32547087] of all the samples in chromosome 14. The variants detected in this region by executing GenSeg on each of the samples is denoted in the first two columns, and the third column specifies the sample/samples in which it has been found. Sample NA12878 have many rare variants in this region, whereas there is a variant in [20213344, 20214522] belonging only to families of YRI ethnicity. Variants in [20374993, 20389257] and [26590957, 26592389] are common variants. Similarly, in chromosome 20, some of the variants that occur in particular ethnicity are [685901, 1537601] and [2910901, 2978401] present in CEU family only, whereas [3032581, 32239010] present only in Utah ethnicity.

3.3 Inference on IG Genes

In this work, we have analysed the effect of the variants on human immunoglobulin (IG) genes, so as to understand the relationship between IG genes and its association with diseases. The heavy and light chains of IG genes lie in chromosomes 14 and 22, the details of which are provided in the Supplementary Material. We have mapped the abnormal segments obtained by GenSeg from different samples, to the IG gene sequences on chromosomes 14 and 22. It has been observed that some of the human immunoglobulin genes have been altered by a large region of duplication and deletion/insertion events. Gene IGLV6-57 in chromosome 22 has been altered by a duplication event in all the samples, except sample NA19240 which has undergone a small deletion event in that genomic region. Gene IGLJ3 in the same chromosome has been affected by a deletion event in Utah (CEU) ethnic group. No abnormality in this gene has been observed in samples NA19238 and NA19239 belonging to Yoruba (YRI) ethnic group. However, a similar deletion event of size 1170013 bp has been observed in sample NA19240. The gene set (IGLC1, IGLL1, IGLL5, IGLV3-21, and IGLV4-3) in chromosome 22, present in the molecular location lying in [21700001, 23100000] with a cytogenic location of $22q11.22$, has been affected only in Utah (CEU) ethnic group. Yoruba (YRI) group has no abnormality in this genomic region. Considering IG genes present in chromosome 14 of all the samples, it has been observed that IGHV3-71 has been affected by duplication event, while IGH and IGHG2 have been affected by deletion event. Other IG genes in chromosome 14, have not been affected by any kind of variations in any sample. Table 3 gives the details of IG genes that have got affected by either duplication or deletion event, for both chromosomes 14 and 22.

3.4 Comparative study

The performance of GenSeg has been compared with some popular DOC based algorithms for detection of copy number variations. The algorithms are EWT [4], CNV-CH [10], CNV-eM [5], CNVkit [9], and cnMOPS [6]. In addition, GenSeg has also been compared with some recent CNV detection algorithms, viz., iCopyDAV [16], DCC [8], SM-RCNV [15] and some recent machine learning based methods,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2020.3000661, IEEE/ACM Transactions on Computational Biology and Bioinformatics

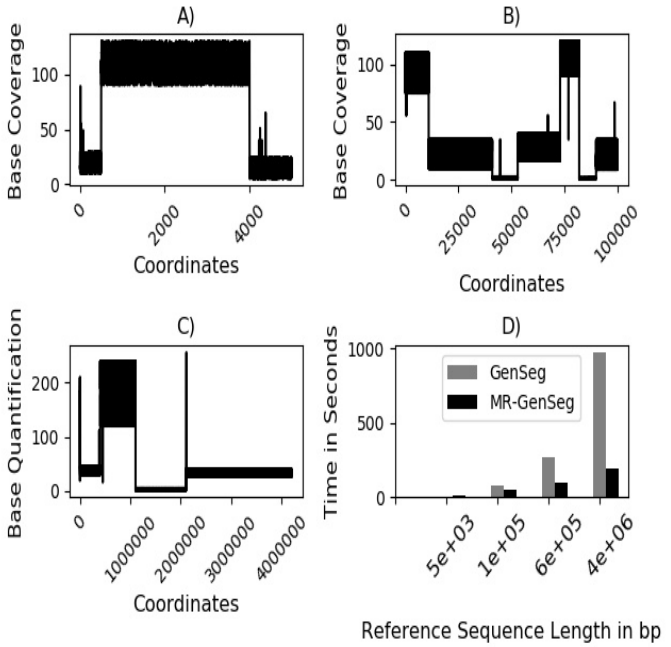

Fig. 1. A) Base coverage obtained via mapping to a reference sequence of length 5000 bp. B) Base coverage obtained via mapping to a reference sequence of length 100000 bp. The x-axis represents the coordinates of the reference sequence, and the y-axis represents the nucleotide base coverage obtained at 15X coverage. C) Base quantification obtained on a reference of 4000000 bp, having 30X coverage. D) The execution time of GenSeg and MR-GenSeg executed on a sample mapped to a reference sequence of length 5000, 100000, 600000, and 4000000 bp, respectively. The y-axis represents the time in seconds.

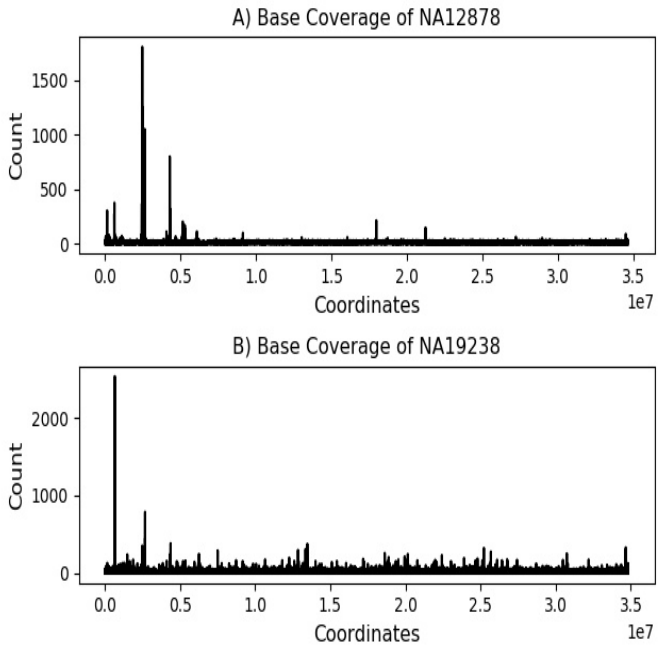

Fig. 2. A) Base coverage values obtained from human chromosome 22 of sample NA12878, belonging to CEU. B) Base coverage values of chromosome 22 of sample NA19238, belonging to YRI ethnicity. Here x-axis denotes the genomic coordinate of the reference sequence and y-axis depicts the base coverage values. The coverage considered here is 30X.

TABLE 3
IG genes that have got affected by some copy number variations. The third and fourth columns of the table represent the exact region of the gene, which have got affected by duplication (Dupl) or deletion (Del) event. Here S1, S2, S3, S4, S5, and S6 correspond to samples NA12878, NA12891, NA12892, NA19238, NA19239, and NA19240 respectively. Y in the first column and last row, refers to the genes IGLL1, IGLV1-44, IGLV2-5, IGLV3-21, IGLV4-3, IGLL5, IGLV3-19, and IGLV3-25.

Gene Name	Chrom	Start Coordinate	End Coordinate	Type	Samples
IGHV3-71	14	20213344	20389257	Dupl	All
IGH	14	70017307	70021111	Del	All
IGHG2	14	106331549	106365772	Del	All
IGLV6-57	22	16175319	16855450	Dupl	S1, S2,S3 S4, S5
IGLV6-57	22	17045349	17046533	Del	S6
IGLJ3	22	23200584	23240415	Del	S1, S2 S3, S6
IGLC1, Y	22	21820448	21903742	Del	S1, S2

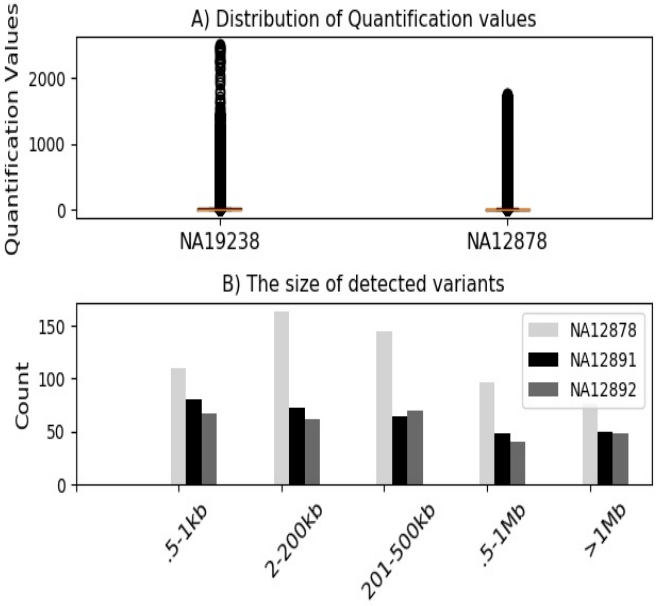

Fig. 3. A) Distribution of the base quantification values obtained from human chromosome 22 of samples NA19238 and NA12878 belonging to YRI and CEU ethnic groups. B) Size of CNVs detected in the CEU family trio.

viz., TrioCNV [12], CanvasSPW [13], and DudeML [17]. All these algorithms execute serially, and work on read depth input. The performance of MR-GenSeg has been compared with a similar MapReduce based CNV detection tool, called HadoopCNV [27]. The details of the parameters considered for these algorithms are provided in the Supplementary Material.

GenSeg has outperformed cnMOPS, DCC, CNV-CH, iCopyDAV, TrioCNV, SM-RCNV and CanvasSPW, in determining precise coordinates of breakpoints of a region with variant. It has been observed that CNV-CH and DCC have performed poorly in presence of frequent outliers, and DCC performs weakly in detecting rare segments. Besides, CNV-CH is computationally expensive for large input data set. On the other hand, CNVeM predicts CNVs at base level, similar to GenSeg; however, CNVeM has performed poorly with respect to execution time for NGS high coverage WGS human DNA sequence data. TrioCNV, CNVkit, SM-RCNV, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2020.3000661, IEEE/ACM Transactions on Computational Biology and Bioinformatics

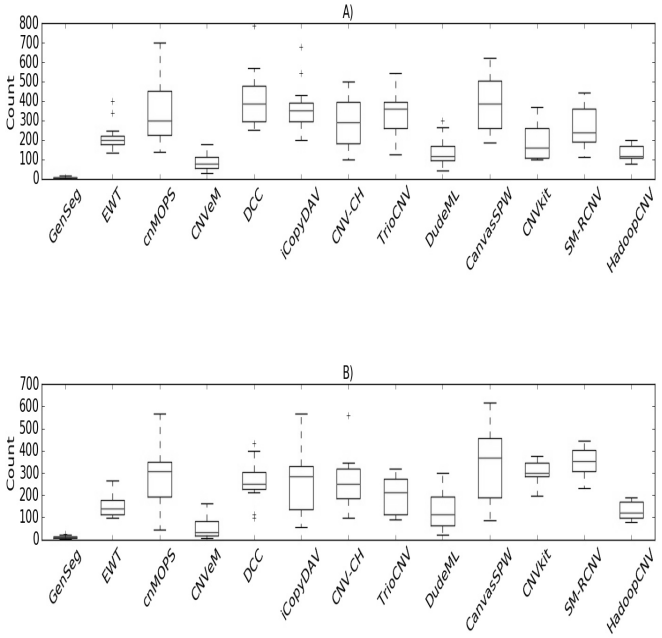

Fig. 4. A) Breakpoint error with respect to 30X coverage, for all the algorithms with which GenSeg has been compared, on simulated data B) Breakpoint error with respect to 15X coverage for all the algorithms.

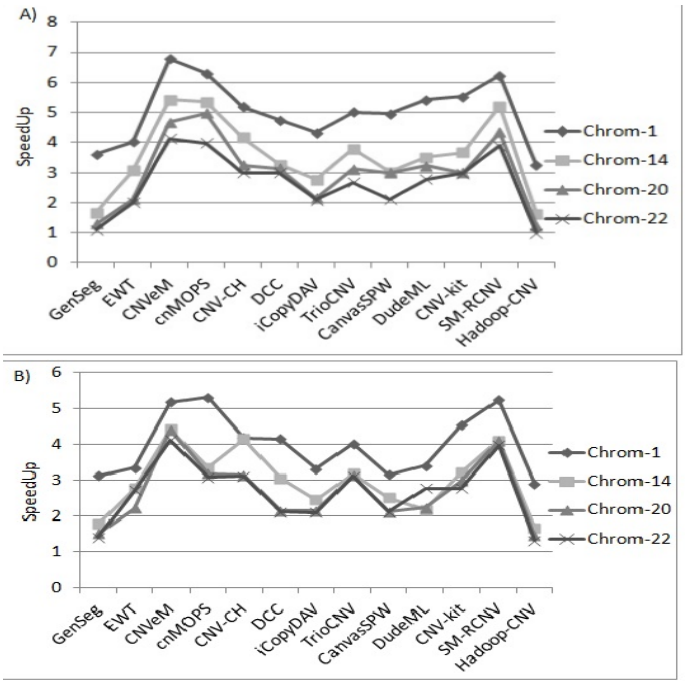

Fig. 6. A) Performance in terms of SpeedUp of GenSeg, EWT, CNVeM, cnMOPS, CNV-CH, DCC, iCopyDAV, TrioCNV, CanvasSPW, DudeML, CNVkit, SM-RCNV, and HadoopCNV, with that of MR-GenSeg (executed on a 4 node cluster), on all the 4 chromosomes, at 30X coverage. B) SpeedUp of all the algorithms, with respect to MR-Genseg, on all the chromosomes at 15X coverage.

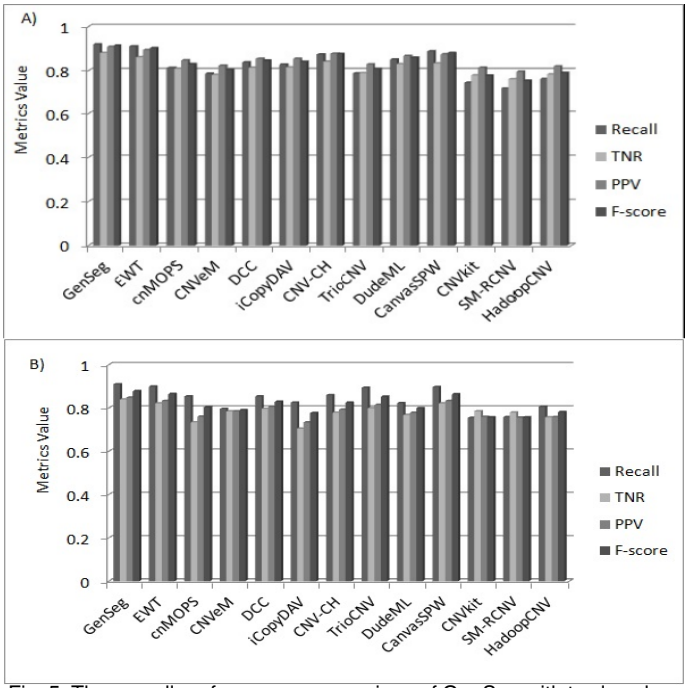

Fig. 5. The overall performance comparison of GenSeg with twelve algorithms in terms of sensitivity (recall), specificity (TNR), precision (PPV), and F-score, for chromosome 1, at 30X coverage A) The performance metrics with respect to sample NA12878. B) The performance metrics with respect to sample NA12891.

CanvasSPW have performed poorly with low coverage data, and small CNV regions remain undetected. Performance of DudeML on real data set is much less efficient with respect to time overhead. The breakpoint detection error in base pair unit at 30X and 15X, are represented in Figs. 4A) and B).

It is observed that at 30X coverage GenSeg has on an average 6 bp breakpoint error in detecting regions with CNVs, whereas HadoopCNV, CNVkit, EWT, and SM-RCNV have 132, 194, 217, and 269 bp breakpoint error. CNV-CH, TrioCNV, cnMOPS, iCopyDAV, CanvasSPW, and DCC have 301, 337, 357, 366, 388, and 417 bp breakpoint error, respectively. It may be mentioned here that the breakpoint error of HadoopCNV has been much high prior to post processing step, detailed in the Supplementary Material. However, CNVeM and DudeML have much less breakpoint error of 84 bp and 140 bp respectively. At low coverage average breakpoint error of GenSeg has been 11, whereas CNVeM, HadoopCNV, EWT, and DudeML have less breakpoint error of 56, 133, 136, and 152 bp respectively. Methods CNV-CH, DCC, iCopyDAV, and cnMOPS have obtained a high breakpoint error of 259, 261, 266, 292 respectively. However, CNVkit, CanvasSPW, and SM-RCNV performed quite poorly having a breakpoint error of 304, 333, and 347 bp. It is to be observed here that CanvasSPW has performed very poorly for both high and low coverage data.

The performance of these algorithms has been demonstrated on chromosomes 1, 14, 20, and 22 of all the human' samples considered in this work. A detected segment is considered true positive if it has at least 5 bp of overlap with the corresponding sequence in the DGV. Figure 5 represents the performance comparison of GenSeg with the

other twelve algorithms for chromosome 1, with respect to Sensitivity (Recall), Specificity (TNR), Precision (PPV) and F-score, corresponding to samples NA12878 and sample NA12891.

It has been observed that GenSeg and EWT have overlapped maximum with DGV. The performance of GenSeg is found better than all the other algorithms with the highest F-score values in all the samples considered. Sensitivity values resulted in by GenSeg are always greater than 0.9, whereas the Specificity, Precision, and F-score have been in the intervals of $[0.84, 0.88]$, $[0.84, 0.90]$ and $[0.87, 0.91]$ respectively. F-score values for EWT, cnMOPS, CNVeM, DCC, iCopyDAV, CNV-CH, TrioCNV, DudeML, CanvasSPW, CNVkit, SM-RCNV, and HadoopCNV, have been obtained in the intervals of $[0.86, 0.90]$, $[0.80, 0.82]$, $[0.79, 0.80]$, $[0.82, 0.84]$, $[0.77, 0.84]$, $[0.82, 0.87]$, $[0.80, 0.85]$, $[0.79, 0.80]$, $[0.86, 0.88]$, $[0.75, 0.77]$, $[0.74, 0.75]$, $[0.78, 0.79]$ respectively. It has been observed that similar to GenSeg, EWT, CanvasSPW and CNV-CH have also obtained a good Sensitivity in $[0.89, 0.91]$, $[0.88, 0.89]$ and $[0.81, 0.85]$, respectively. A few variants obtained by GenSeg have been considered false positive as they are not listed in DGV, although they have overlapped with the variants detected by the other algorithms. These false alarms may be considered as novel variants as listed in Table 1 for chromosome 22 in sample NA12878.

Performance comparison of MR-GenSeg with GenSeg, as well as all the other algorithms, has been performed with respect to SpeedUp, i.e., the ratio between the execution time of the compared algorithms, with that of MR-GenSeg. It has been observed that GenSeg and MR-GenSeg behave similarly with respect to time for small data, i.e., simulated data sets, having reference size less than 100000 bp. However, as reference sequence length increases, MR-GenSeg has performed much more efficiently with respect to execution time as represented in Fig. 1D). MR-GenSeg has outperformed on the real data sets as represented in Fig. 6.

SpeedUp of MR-GenSeg (executed on a 4node cluster) as compared to GenSeg, EWT, CNVeM, cnMOPS, CNV-CH, DCC, iCopyDAV, TrioCNV, CanvasSPW, DudeML, CNVkit, SM-RCNV, and HadoopCNV, on all the 4 chromosomes at 30X coverage has been reflected in Fig. 6A), whereas Fig. 6B) represents SpeedUp of MR-GenSeg as compared to the other algorithms on all the 4 chromosomes, at low coverage dataset.

The SpeedUp of MR-GenSeg with respect to chromosome 1 at 30X coverage have been high as compared to all the the algorithms. It has been greater than 6 when compared with CNVeM, cnMOPS and SM-RCNV, whereas, SpeedUp has been 5 times more when compared with CNV-CH, TrioCNV, DudeML and CNVkit. When compared with EWT, DCC, iCopyDAV, and CanvasSPW, SpeedUp has been 4 times more, however, GenSeg and HadoopCNV performed better where MR-GenSeg obtained a SpeedUp of 3.62 and 3.24 respectively. Similarly, for low coverage data (chromosome 1) also HadoopCNV and GenSeg performed better with respect to execution time, where MR-GenSeg obtained a SpeedUp of 2.9 and 3.12. Moreover, CanvasSPW, iCopyDAV, EWT, and DudeML also performed better as compared to other algorithms. However, MR-GenSeg achieved a high SpeedUp of 4 to 5 times with respect to CNVeM, cnMOPS, DCC, CNV-CH, TrioCNV, CNVkit, and SM-RCNV .

When chromosomes 14, 20 and 22 of all samples have been considered, MR-GenSeg obtained a SpeedUp in the range $[1.11, 1.77]$, $[2.00, 3.07]$, $[4.12, 5.42]$, $[3.08, 5.35]$, $[3.00, 4.16]$, $[3.00, 3.26]$, $[2.11, 2.77]$, $[2.67, 3.79]$, $[2.11, 3.01]$, $[2.78, 3.5]$, $[2.98, 3.67]$, $[3.89, 5.21]$, and $[1.10, 1.67]$, as compared with GenSeg, EWT, CNVeM, cnMOPS, CNV-CH, DCC, iCopyDAV, TrioCNV, CanvasSPW, DudeML, CNVkit, SM-RCNV, and HadoopCNV.

Since, chromosome 1 is the largest human chromosome, MR-GenSeg has performed much better with respect to execution time, and has achieved approximately 7 times SpeedUp as compared to some algorithms, however, the SpeedUp of MR-GenSeg with respect to HadoopCNV (a MapReduce approach) and GenSeg is less compared to the other algorithms. MR-GenSeg is on an average 1.52 and 3.26 times faster than HadoopCNV on chromosomes (14, 20, and 22), and chromosome (1) respectively.

4 Conclusion

GenSeg, a novel CNV detection algorithm, has been developed in this article. The algorithm efficiently works on information processed on each nucleotide position of human genome sequence. This has enabled GenSeg to detect the boundaries of each potential CNV region (Fig. 4) precisely. Since the algorithm works with whole genome DNA sequence, it is able to identify the variations in both intragenic and intergenic sequences. Thus, the scope of further research on the functions of the non-coding DNA is possible. While comparing GenSeg with twelve popular algorithms, it has been observed that the algorithm has outperformed the others with respect to Sensitivity and F-score (Fig. 5). GenSeg has also excelled in determining small as well as large variants of size greater than a few megabases (Fig. 3).

In order to overcome the issues regarding storage and processing of NGS whole genome sequence data, a parallel MapReduce based algorithm called MR-GenSeg, has also been developed on Hadoop framework. The performance of MR-GenSeg in terms of SpeedUp has been compared with all the twelve algorithms, as well as another MapReduce based CNV detection algorithm, called HadoopCNV. MR-GenSeg has achieved on an average more than 3 times faster SpeedUp, on chromosome 1 compared to HadoopCNV, and obtained approximately 7 times high SpeedUp with respect to some other serial algorithms (Fig. 6).

The novel tree data structure, developed in the present work, has been used to analyse every potential variant obtained, in terms of common and rare variants. Besides, ethnicity-wise common and rare variants have been identified. This data structure can also be applied to analyse the variants with respect to a disease. The present work has considered all the IG genes and has identified all the CNV events influencing these genes (Table 3). This extends the scope of further analysis of these influenced IG genes for proper functioning of the immune system, which may further control the onset of critical diseases including cancer.

References

[1] M. L. Metzker, *Sequencing technologies-The next generation*, *Nat. Rev. Genet.*, vol. 11, 2010.

[2] C. T. Watson, and F. Breden, *The immunoglobulin heavy chain locus: Genetic variation, missing data, and implications for human disease*, *Genes Immun.*, vol. 13, 2012.

[3] A. Magi, L. Tattini, T. Pippucci, F. Torricelli, and M. Benelli, *Read count approach for DNA copy number variants detection*, *J. Bioinform.*, vol. 28, 2012.

[4] S. Yoon, Z. Xuan, and V. Makarov, *Sensitive and accurate detection of copy number variants using read depth of coverage*, *Genome Res.*, vol. 19, 2009.

[5] Z. Wang, F. Hormozdiari, W. Yang, E. Halperin, and E. Eskin, *CNVeM: Copy number variation detection using uncertainty of read mapping*, *J. Comput. Biol.*, vol. 20, 2013.

[6] G. Klambauer, K. Schwarzbauer, A. Mayr, and S. Hochreiter, *Cn.MOPS: Mixture of poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate*, *Nucleic Acids Res.*, vol. 40, 2012.

[7] Q. Zhang, L. Ding, D. E. Larson, D. C. Koboldt, M. D. McLellan, K. Chen, X. Shi, A. Kraja, E. R. Mardis, R. K. Wilson, I. B. Borecki, and M. A. Province, *CMDS: a population-based method for identifying recurrent DNA copy number aberrations in cancer from high-resolution data*, *J. Bioinform.*, vol. 26, 2010

[8] X. Yuan, J. Zhang, L. Yang, J. Bai, and P. Fan, *Detection of significant copy number variations from multiple samples in Next-Generation Sequencing data*, *IEEE T NANOBIOSCI.*, vol. 17, 2018.

[9] E. Talevich, A. H. Shain, T. Botton, and B. C. Bastian, *CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing*, *PLoS Comput. Biol.*, vol. 12, 2016.

[10] R. Sinha, S. Samaddar, and R. K. De, *CNV-CH: A Convex Hull based segmentation approach to detect copy number variations (CNV) using Next-Generation Sequencing data*, *PLoS ONE*, vol. 10, 2015.

[11] J. Duan, H. Deng, and Y. Wang, *Common copy number variation detection from multiple sequenced samples*, *IEEE T BIO-MED ENG.*, vol. 61, 2014.

[12] Y. Liu, J. Liu, J. Lu, J. Peng, L. Juan, X. Zhu, B. Li, and Y. Wang, *Joint detection of copy number variations in parent-offspring trios*, *J. Bioinform.*, vol. 32, 2016.

[13] S. Ivakhno, E. Roller, C. Colombo, P. Tedder, and A. J. Cox, *Canvas SPW: Calling de novo copy number variants in pedigrees*, *J. Bioinform.*, vol. 34, 2017.

[14] A. Malekpour, H. Pezeshk, and M. Sadeghi, *MSeq-CNV: Accurate detection of Copy Number Variation from Sequencing of Multiple samples*, *Sci. Rep.*, vol. 8, 2018.

[15] Y. Li, X. Yuan, J. Zhang, L. Yang, J. Bai, and S. Jiang, *SM-RCNV: a statistical method to detect recurrent copy number variations in sequenced samples*, *GENES GENOM*, vol. 41, 2019.

[16] P. Dharanipragada, S. Vogeti, and N. Parekh, *iCopyDAV: Integrated platform for copy number variations- Detection, annotation and visualization*, *PLOS ONE*, vol. 13, 2018.

[17] T. Hill, and R. L. Unckless, *A Deep Learning Approach for Detecting Copy Number Variation in Next-Generation Sequencing Data*, *G3-GENES GENOM. GENET.*, vol. 9, 2019.

[18] S. Samaddar, R. Sinha, and R. K. Dey, *A Model for distributed processing and analyses of NGS data under Map-Reduce Paradigm*, *IEEE ACM T COMPUT BI.*, vol. 16, 2019.

[19] A. O'Driscoll, J. Daugelaite, and R. D. Sleator, *'Big data', Hadoop and cloud computing in genomics*, *J. Biomed. Inform.*, vol. 46, 2013.

[20] R. Tripathi, P. Sharma, P. Chakraborty, and P. K. Varadwaj, *Next-Generation Sequencing revolution through big data analytics*, *FRONT LIFE SCI.*, (Online) Journal homepage: http://www.tandfonline.com/loi/tfls20, 2016.

[21] The apache software foundation, *Hadoop*, Available: https://hadoop.apache.org.

[22] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*, The Apache Software Foundation, vol. 16, 2005.

[23] D. Miner, and A. Shook, *Map-Reduce Design Patterns*, Oreilly, 2012.

[24] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. Depristo, *The genome analysis toolkit: A mapReduce framework for analyzing next-generation DNA sequencing data*, *BMC Bioinformatics*, vol. 20, 2010.

[25] M. Niemenmaa, A. Kallio, A. Schumacher, P. Klemel, E. Korpelainen, and K. Heljanko, *Hadoop-BAM: Directly manipulating Next Generation Sequencing data in the cloud*, *J. Bioinform.*, vol. 28, 2012.

[26] The SAM/BAM format specification working group, *Sequence alignment format specification*, Available: https://samtools.github.io/hts-specs/SAMv1.pdf.

[27] H. Yang, G. Chen, L. Lima, H. Fang, L. Jimenez, M. Li, G. J. Lyon, M. He, and K. Wang, *HadoopCNV: A dynamic programming imputation algorithm to detect copy number variants from sequencing data*, *bioRxiv.*, 2017.

[28] A. Belkadi, A. Bolze, Y. Itan, A. Cobat, Q. B. Vincent, A. Antipenko, L. Shang, B. Boisson, J. L. Casanova, and L. Abel, *Whole-genome sequencing is more powerful than whole-exome sequencing for detecting exome variants*, *PNAS.*, vol. 112, 2015.

[29] Y. Benjamin, and T. P. Speed, *Summarizing and correcting the GC content bias in high-throughput sequencing*, *Nucleic Acids Res.*, vol. 40, 2012.

Ms. Rituparna Sinha is currently working as an Assistant Professor in the Department of Information Technology at Heritage Institute of Technology, Kolkata. She obtained her M.Tech. degree from National Institute of Technical Teachers Training and Research, Kolkata, in the year 2008. She has 12 years of teaching experience. Ms. Sinha has authored four papers published in Journals and a book chapter. Her research interests include Computational Biology, Data Science, Distributed Databases, and Algorithms in Big Data Paradigms.

Dr. Rajat K. Pal (Member IEEE) received the B.E. degree in Electrical Engineering in 1985 from Bengal Engineering College, Shibpur, University of Calcutta, the M.Tech. degree in Computer Science and Engineering in 1988 from the University of Calcutta, and the Ph.D.degree in Computer Science and Engineering in 1996 from Indian Institute of Technology (I.I.T.), Kharagpur. Since 1994, he has been as a faculty with the Department of Computer Science and Engineering, University of Calcutta. He has also worked as the Head of the Department from 2005 to 2007 and from 2016 to 2018. He went on to become Professor with the Department of Information Technology, Assam University, Silchar, India from 2010 to 2012, on lien from the Department of Computer Science and Engineering, University of Calcutta. At present, he is working as a Professor with the Department of Computer Science and Engineering, University of Calcutta. His major research interests include VLSI design, graph theory and its applications, perfect graphs, logic synthesis, design and analysis of algorithms, computational geometry, parallel computation and algorithms. Dr. Pal has authored more than 200 technical research articles and authored and co-authored two books of international standard. Dr. Pal also holds several international patents.

Dr. Rajat K De (Senior Member, IEEE Membership Number 41260992) is a Professor of the Indian Statistical Institute, Kolkata, India. He obtained his Bachelor of Science (with Physics Major), Bachelor of Technology (Computer Science and Engineering) from University of Calcutta, and Master of Engineering (Computer Science and Engineering) from Jadavpur University in the years 1987, 1991 and 1993, respectively. Dr. De received his Ph.D. degree from the Indian Statistical Institute, India, in 2000. He was a Distinguished Postdoctoral Fellow of the Whitaker Biomedical Engineering Institute, the Johns Hopkins University, USA during 2002-2003. He visited the Department of Medicine, University of California, San Diego, under Fulbright-Nehru Academic and Professional Excellence Fellowship Program, during 2017-2018. Dr. De has authored about 100 papers published in journals, edited books and in conference proceedings. His research interest includes modelling biochemical pathways, multi-omics data analysis, pattern recognition, machine learning, deep neural networks, and data sciences.