POMP: a powerful splice mapper for RNA-seq reads

Subrata Saha

Department of Computer Science and
Engineering
University of Connecticut, Storrs, CT
subrata.saha@engr.uconn.edu

Sanguthevar Rajasekaran^{*}

Department of Computer Science and
Engineering
University of Connecticut, Storrs, CT
rajasek@engr.uconn.edu

ABSTRACT

RNA Sequencing (RNA-seq) based on next-generation sequencing (NGS) technology enables transcriptome analyses of entire genomes at a very high resolution. Due to limitations of the sequencing technology the reads are very short and erroneous. As a consequence it is a very challenging task to accurately map RNA-seq reads onto the genome and identify the splice junctions. There are two shortcomings in some of the existing best-known algorithms for identifying splice junctions: 1) the junction boundaries are predicted within a large range, 2) they take a long time and 3) the predictions of splice junctions are inaccurate. In this article we propose a novel algorithm POMP to accurately detect the splice junctions considering all the shortcomings of the existing algorithms as stated above. It generates accurate candidate splice junctions utilizing expected properties around the splice junctions. These candidates are used as examples to train a learner. The model learnt by the learner can be used to identify splice junctions. Extensive experiments were done considering whole human genome splicing events. Experimental results show that POMP is indeed a more effective and efficient algorithm compared to the other state-of-the-art algorithms in terms of sensitivity, specificity, and runtime.

Categories and Subject Descriptors

J.3 [LIFE AND MEDICAL SCIENCES]: Biology and genetics; H.3.3 [Information Search and Retrieval]: Clustering, Search process, Selection process;

General Terms

Algorithms, Experimentation

^{*}Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
BCB'16, October 2–5, 2016, Seattle, WA, USA.
Copyright 2016 ACM 978-1-4503-4225-4/16/10 ...\$15.00.
<http://dx.doi.org/10.1145/2975167.2975210>.

Keywords

Gapped Alignment, Machine Learning, Next-Generation Sequencing (NGS), Powerful Splice Mapper (POMP), Representatives, Splice Junctions (SJs), Support Vector Machines (SVMs).

1. INTRODUCTION

RNA Sequencing (RNA-seq) refers to the use of next-generation sequencing technologies to sequence complementary DNA (cDNA) in order to get information about a sample's RNA content. In genetics a cDNA is a DNA synthesized from a messenger RNA (mRNA) template in a reaction catalyzed by the reverse transcriptase and DNA polymerase enzymes. It is often used to clone eukaryotic genes in prokaryotes. RNA-seq generates millions of short sequence fragments or reads in a single run within a very brief period of time. The reads obtained from this can then be aligned to a reference genome in order to construct a whole-genome transcriptome map and thus can be used to measure levels of gene expression and to identify novel splice variants of genes. But to fully utilize such reads one needs to be able to accurately align the sequencing reads over the intron boundaries in a reference genome. This mapping problem represents a significant challenge given the small read lengths and inherent high error rates.

One of the main purposes of RNA-seq technology is to detect genome-wide splicing events efficiently and accurately. In molecular biology and genetics, splicing is a modification of a RNA molecule after transcription, in which introns (the parts of DNA sequence that are spliced out) are removed and exons (the parts of DNA sequence after splicing) are joined. This is needed for the typical eukaryotic messenger RNA before it can be used to produce a correct protein through translation. We can define and formulate the problem of interest as follows: Splice junctions are points on a DNA sequence at which superfluous DNA is removed during the process of protein synthesis in higher organisms. The problem posed is to recognize, given a sequence of DNA, the boundaries between exons and introns, i.e., donor or acceptor sites. It is more formally stated as: Given a position in the middle of a window of DNA sequence elements (nucleotides), decide whether this is an intron-exon boundary, exon-intron boundary or not and align the reads accordingly.

In this article we propose a new algorithm called POMP to detect genome-wide splicing events. It incorporates representatives from the clusters of highly similar reads and thus reduces the number of initially unmapped reads dramatically. It enhances the chances of detecting splicing events

accurately and precisely. It also reduces the run time drastically. After getting the candidate splice junctions (SJs) from gapped alignment, we extract useful features inherent in the alignment and genomic sequence to group candidates into two classes, namely, accurate and inaccurate with a very high level of confidence. Non-linear Support Vector Machine (SVM) is subsequently employed for binary classification.

The rest of this paper is organized as follows: Some related works are discussed briefly in Section 2. A description of our proposed algorithm is presented in Section 3. The performance of the algorithm and experimental results are presented in Section 4. Some discussions are provided in Section 5. Finally Section 6 concludes the paper.

2. RELATED WORKS

Many methods and algorithms have been devised to detect genome-wide splicing events. There are two basic procedures for RNA-seq reads alignment without employing gene annotation. Those are *exon inference* and *gapped alignment*. TopHat [1] follows the first approach where it maps the reads to the reference genome using Bowtie [2] and then clusters all the Initially Mapped (IM) reads. IM reads are those reads that can be completely mapped to the reference without being split into several parts. TopHat calls the resulting clusters as islands and each island represents putative exon regions. To recognize potential splice junctions, TopHat first enumerates all the canonical donor and acceptor sites within the island sequences. After enumeration, it considers all pairing of these sites that could form canonical (GT-AG) introns between neighboring (but not necessarily adjacent) islands. Each possible intron is then checked with the IUM reads using seed-and-extend strategy. IUM reads are those reads that cannot be completely mapped to the reference without being split into several parts. For each of possible splice sites, a seed is formed by appending a small amount of sequence upstream of the donor and downstream of the acceptor sites. Any read containing the seed is checked for a complete alignment to the exons on either side of the possible exons. This approach is proved to be very accurate since it infers exons with a reasonably high coverage. But TopHat may fail to detect splice junctions in a number of cases. For example, when the transcript is located in a region with a low sequencing depth, there might not be enough reads that straddle the junction for easy detection. As another example, TopHat may fail when the exon length is shorter than the read length.

SpliceMap [3] and several other methods [4], [6], [5] and [7] follow the second approach, namely, gapped alignment. There are four main steps in the SpliceMap algorithm. These are half-read mapping, seeding selection, junction search and paired-end filtering. At first each read is divided into halves and SpliceMap maps both halves of the read to the reference genomic sequence with a high probability. This step is done using any existing short alignment tool such as SeqMap [8] or ELAND. If a half-read is mapped exactly within the genomic sequence, it is called a mapped hit and dubbed as seeding. For each seeding, the alignment on the reference genome is then extended base by base to find the splicing point. It subsequently tries to find a partner splicing point that provides a perfect match for the corresponding residual sequence of the original read. Paired-end information of reads is used to filter out false positives. This mapping and extension approach is not an effective way to handle reads

with sequencing errors, since the call rate will be low especially when the expression level is low. Also, SpliceMap is not able to effectively deal with reads that can be mapped to multiple locations. They simply ignore the hits that are too close together. Another paper [9] proposes an algorithm that is similar to SpliceMap. In the alignment procedure every read is split into two non-overlapping parts or “anchors” that are aligned separately. The two anchors are then extended as long as they still match the reference sequence. If a splice junction is located between the gaps of the two anchors, there is a high probability that the two anchors correspond to two exons in the DNA sequence. TopHat2 [10] adapts both the strategies where short reads are mapped using *exon inference* and long reads are mapped using *gapped alignment*.

3. METHODS

Our algorithm POMP runs in two phases. At the first phase it detects initially unmapped reads (IUMs) and constructs representatives from those IUMs. The representatives are then used to find the candidate splice junctions (SJs) by employing gapped alignment procedure. In the second phase we prune the candidate splice junctions to reduce the false positives with the help of a novel machine learning technique. In this context our algorithm can be thought of as a self-learning algorithm. We first generate a set of candidate splice junctions. These candidates are then further processed to get a set of highly accurate splice junctions. These form the positive examples for the learning step. We also generate a set of negative examples. These examples are then used to train the learner. The model learnt by the learner can then be used to identify splice junctions. Our learning is based on Support Vector Machines (SVMs). Details of our algorithm are provided next.

3.1 Finding candidate splice junctions

This phase has three sub-phases. In the first sub-phase we find a set of unmapped reads. Since the coverage of RNA-seq reads are very high, we can build clusters of similar reads. A representative string for each cluster can then be formed by taking the consensus from the overlapping reads. These representatives are naturally longer than the original reads. The bases in the representatives are also highly accurate. This is exactly what is done in the second sub-phase. The representatives are then aligned onto genome using our gapped alignment algorithm.

3.1.1 Generating initially unmapped reads

At first we align the reads onto the genomic sequence G . Any alignment tool can be used to perform this step. We used Bowtie alignment tool to perform this task within a predefined Hamming distance d , where $2 \leq d \leq 3$. The value of d depends on the length of the read. The reads which are aligned onto G within a Hamming distance d are called Initially Mapped (IM) reads and they form the set IM . The rest of the reads are called Initially Un-Mapped (IUM) reads and form the set IUM . Reads in IUM are of two types. The first type of reads in IUM could not be aligned since they span two or more exons in mRNA. The second type of reads in IUM cannot be aligned within a Hamming distance of d due to (more than d) errors. The

first type of reads play a key role in detecting the splice junctions.

3.1.2 Generating representatives

In the next-generation sequencing technology millions to billions of short reads are generated in a single run within a very short period of time. Since reads are generated by fragmenting the genome in random positions, the chances of finding highly overlapped reads are very high. If we manage to cluster highly overlapped *IUM* reads and create a representative string for each of the clusters, not only will the runtime drastically reduce but also the representatives could be mapped with a high level of accuracy. This is due to the fact that the representatives are built from the consensus of the clustered reads and the representatives are longer than the RNA-seq reads. The details of this sub-phase is provided next.

1. Let $r \in IUM$ be any read from the set of initially unmapped (IM) reads. At first each read $r \in IUM$ is decomposed into overlapping substrings of a fixed length k . These k -mers are then hashed into a hash table H . Each entry (i.e., bucket) $b \in H$ represents a set of similar reads (since they share a k -mer with a high probability).
2. For each read $r \in IUM$ we collect all other reads $L(r)$ that are sufficiently overlapped with r . Specifically, the Hamming distance in the overlapping region is no more than a predefined threshold τ , where the value of τ depends on the length of the overlapping region and $2 \leq \tau \leq 3$. $L(r)$ along with r forms a cluster c of highly overlapped reads coming from the same genomic region of G . Please note that in our procedure each read $r \in IUM$ can be in only one cluster c . Let C stand for the set of all such clusters.
3. For each cluster $c \in C$ we align each r' from $L(r)$ onto r using our greedy alignment algorithm [11]. This is accomplished by aligning similar k -mers between r' and r . But there can be more than one k -mer shared by r' and r . While processing the buckets of the hash table H , we keep track of the k -mer for which the size of the overlapping region between r and r' is the longest and at the same time the Hamming distance between them is within the predefined threshold τ as described in the previous step. We align r and r' based on this k -mer. We perform a greedy alignment that does not take much time. After aligning the reads in $L(r)$ with r , we build the consensus string. This consensus string is called in our algorithm as the representative (of c).

3.1.3 Mapping representatives

Each representative is divided into two non-overlapping segments of equal size. Each segment is then mapped onto the genome using Bowtie within a predefined Hamming distance d' . Segments can be mapped in several positions and thus have the potential to detect alternate splicing events. There are three cases to be considered while mapping the segments. Consider a representative S with segments be S_1 and S_2 .

1. Segments S_1 and S_2 of the representative S can be mapped onto the genomic sequence G within a Hamming distance d' where $d' \leq 1$. It is possible that S_1 and/or S_2 can be

mapped in several positions in G . If the order of the mapping is coherent i.e., the ending position p of S_1 is followed by the starting position q of S_2 , (p, q) constitutes a candidate splice junction. The alignment of S_1 and S_2 in this case constitutes a *gapped alignment*. Let n_1 and n_2 be the number of mappings of S_1 and S_2 , respectively, that are coherent. Then there will be at most $n_1 \times n_2$ possible candidate splice junctions corresponding to S .

2. Assume that only S_1 is mapped onto G within a Hamming distance of d' where $d' \leq 1$. In this scenario we fix S_1 onto G and extend it to the right by appending one base from S_2 at a time starting from left until the number of mismatches m is within a Hamming distance of d'' . The value of d'' is dynamically calculated based on the modified length of S_2 . Let S_1 appended with this portion of S_2 be denoted as S'_1 . Let the remaining part of S_2 be referred to as S'_2 . If S'_1 ends in position p , S'_2 is then mapped onto G with a starting point that is within the range $[p, p + c]$ where c is a user defined constant. Let S'_2 start at q in the genome G . The pair (p, q) constitutes the donor and acceptor sites of the candidate splice junction.
3. This scenario is exactly the same as described above with S_1 replaced by S_2 and vice-versa. We fix S_2 onto G and extend it to the left by appending one base from S_1 at a time starting from right until the number of mismatches m is within a Hamming distance d'' as described above. If S'_2 starts at position q , S'_1 is then mapped onto G within the range $[q, -(q + c)]$ where c is a user defined constant. Let S'_1 end at p in genome G . The pair (p, q) constitutes the candidate splice junction.

3.2 Finding highly accurate splice junctions

We get candidate splice junctions (SJs) from the previous phase. But this candidate set might contain a lot of false SJs which need to be eliminated. In this phase we apply Support Vector Machine (SVM) - a well-known machine learning technique - to classify candidate SJs into two categories, i.e., accurate and inaccurate SJs with a very high level of confidence.

3.2.1 Deriving initial accurate/inaccurate splice junctions

Support Vector Machine (SVM) has been developed by Vapnik et al. at AT&T Bell Laboratories [12] [13]. Kernel-based techniques (such as support vector machines, Bayes point machines, kernel principal component analysis, and Gaussian processes) represent a major development in machine learning algorithms. Support vector machines (SVMs) are a group of supervised learning methods that can be applied to classification or regression. They represent an extension to nonlinear models of the generalized portrait algorithm. The basic idea is to find a hyperplane which separates any given d -dimensional data perfectly into two classes. In brief, SVM is a supervised learning model with an associated learning algorithm. It analyzes data and recognizes patterns and is used for classification and regression analysis. Given a set of training examples, each marked with membership information to one of two categories, an SVM training algorithm builds a model. This model assigns unseen examples into one category or the other, making it a non-probabilistic binary linear classifier. To build the model

accurately we need an adequate number of training examples. In our case the candidate junctions will form the examples set. This set contains highly accurate and inaccurate SJs. Although it is very difficult to label the candidate SJs as true or false with 100% confidence without any prior knowledge, we can certainly select a small subset of true and false SJs by carefully investigating some properties around the candidate SJs.

To derive training examples for supervised classification we divide the set of representatives into three disjoint subsets. Uniquely Gapped Representatives (UGRs): the set *UGR* consists of a subset of the representatives in which each representative has only one canonical SJ (a SJ is canonical if the corresponding intron starts with GT and ends with AG). Multiple Gapped Representatives (MGRs): each representative in this subset *MGR* has more than one canonical SJs. Non-canonical Uniquely Gapped Representatives (NUGRs): each representative in this subset *NUGR* must have only one non-canonical or semi-canonical SJ (a semi-canonical SJ is a SJ where the corresponding intron starts with AT and ends with AC or starts with GC and ends with AG). We define accurate and inaccurate spliced junctions next.

1. Initial accurate splice junctions (IASJs)

Each representative in *UGR* has only one canonical SJ as described above. We retain those SJs in which there is no mismatch in either side of the gapped alignments. We also retain any SJ under the following scenario. Consider any SJ *S*. Let *A* refer to the 12 bps to the left of the donor site of *S* and *B* refer to the 12 bps to the right of the acceptor site of *S*. There could be a number of representatives that have the same SJ *S* and have *A* as a substring and each such representative has been constructed from a number of *IUM* reads. If the total number of such reads across all the representatives that have the same SJ *S* and share *A* is ≥ 5 then *S* is retained. Symmetrically, if the total number of reads across all the representatives that have the same SJ *S* and share *B* is ≥ 5 then also *S* is retained.

2. Initial inaccurate splice junctions (IISJs)

This set is acquired by selecting those candidate SJs which are supported by MGRs but not any UGRs. Those SJs that are supported by only NUGRs and whose mapping length is shorter than 12 bp are also retained. (The mapping length of any SJ *S* is the length of the shorter of *S*₁' and *S*₂'.)

3.2.2 Deriving features

Every SJ can be represented by some characteristics locally observed around it. These characteristics are called features or dimensions in any classification algorithm. The features described below may decide altogether whether a candidate SJ is accurate or inaccurate. In this context let *J(a, b)* denotes a splice junction where *a* and *b* refers the coordinates of donor and acceptor sites, respectively.

1. Exon-intron distribution

The distribution of the bases around exon-intron border in any genome *G* may give valuable information to classify a candidate SJ more accurately. We devise a novel technique to extract the hidden information around exon-intron boundary by employing *k*-mer distribution. The distribution is computed by considering highly accurate SJs from

IASJs. In this procedure a region of fixed length *l* around exon-intron boundary of a SJ is extracted from the genome. This region is then decomposed into overlapping substrings of length *k*. This process is repeated for all SJs in IASJs. We sort the *k*-mers based on the abundance and take the top 10 *k*-mers. In a similar way regions of a fixed length *l* is extracted from a candidate SJ of interest and they are decomposed into *k*-mers. If any of these *k*-mers from a candidate SJ can be found in top 10 list, we score it as 1; otherwise the score is 0.

2. Intron-exon distribution

In this procedure a region of fixed length *l* around the intron-exon boundary of a SJ is extracted. It is then decomposed into overlapping substrings of length *k*. This process is repeated for all SJs in IASJs and use the same procedure as described above.

3. Boundary coverage score

Boundary coverage score measures the average coverage depth of donor and acceptor sites together. Initially Mapped (IM) reads are used to compute this score. Since introns are spliced off from genome *G* to form mRNA, there should be a large concentration of mapped reads to the left and right regions of donor and acceptor sites of *G*, respectively. If we could formulate a meaningful feature from this observation, it should definitely help us in classifying accurate and inaccurate splice junctions. Let *R_i* be the total number of IM reads that are mapped onto the genome *G* starting from position *i*. We define coverage score of a region as the average number of reads mapped onto that region. Let a region be bounded by *l₁* and *l₂*. The coverage score of this region can be expressed as $C_{score}(l_1, l_2) = \frac{1}{l_2 - l_1} \sum_{i=l_1}^{l_2} R_i$. Score for donor and acceptor regions are defined as $C_{score}(a) = C_{score}(a - c_1, a + c_2)$ and $C_{score}(b) = C_{score}(b - c_2, b + c_1)$, respectively where *c₁* and *c₂* are user defined constants. Now the boundary coverage score for a junction *J(a, b)* is the score derived by summing these two scores, i.e., $B_{score} = C_{score}(a) + C_{score}(b)$.

4. Intron coverage score

Initially Mapped (IM) reads are used to calculate an intron coverage score. As introns are spliced out during the formation of the mRNA, it is intuitive that the average coverage depth of the intron specified by the SJ will be very low. Let *a* and *b* be the co-ordinates (i.e., positions) of the donor and acceptor sites, respectively. We calculate the intron coverage score as the average number of reads mapped onto a base in the region bounded by the interval (*a, b*) as described above. In this scenario the scores for the donor and acceptor regions are defined as $C_{score}(a) = C_{score}(a - c_3, a + c_4)$ and $C_{score}(b) = C_{score}(b - c_4, b + c_3)$, respectively where *c₃* and *c₄* are user defined constants. Now the intron coverage score for a junction *J(a, b)* is the score derived by summing these two scores, i.e., $I_{score} = C_{score}(a) + C_{score}(b)$.

5. Donor-acceptor pattern score

Almost 99% of the novel splice junctions contain canonical donor-acceptor sites (i.e. AC-GT). Very few of the junctions contain semi-canonical and non-canonical donor-acceptor sites. Based on this observation, a high score *s₁* is given to the corresponding SJ if it contains canonical donor-acceptor sites. A medium score *s₂* and a low score *s₃* are given for semi-canonical and non-canonical sites, respectively.

Table 1: Genome-wide performance evaluation for all datasets.

Dataset	Read Length	Junctions	Method	Predicted	True	False	SN (%)	SP (%)	GN (%)	CPU Hour
D1	50 bp	140,856	TopHat2	137,701	121,482	16,219	86.25	88.22	87.23	25
			POMP	131,295	122,543	8,752	87.00	93.33	90.16	17
D2	75 bp	140,856	TopHat2	147,712	130,257	17,455	92.48	88.18	90.33	17
			POMP	126,841	117,087	9,754	90.05	92.31	91.18	10
D3	100 bp	140,856	TopHat2	153,630	133,463	20,167	94.75	86.87	90.81	25
			POMP	140,084	127,855	12,229	90.77	91.27	91.02	18

Table 2: Chromosome-wide performance evaluation for dataset D1 for first 10 chromosomes.

Chromosome	Simulated Junctions	Method	Predicted	True	False	SN (%)	SP (%)	GN (%)
1	14,162	TopHat2	13,868	12,177	1,691	85.98	87.81	86.90
		POMP	13,134	12,153	981	86.15	92.71	89.43
2	10095	TopHat2	9,523	8,406	1,117	83.27	88.27	85.77
		POMP	9,235	8,524	711	84.43	92.3	88.37
3	8,223	TopHat2	7,858	7,143	715	86.87	90.90	88.88
		POMP	7,968	7,369	599	89.61	92.48	91.05
4	5,181	TopHat2	4,790	4,309	481	83.17	89.96	86.56
		POMP	4,908	4,471	437	86.29	91.09	88.69
5	6,095	TopHat2	5,734	5,293	441	86.84	92.31	89.58
		POMP	5,701	5,371	330	88.12	94.21	91.17
6	6,562	TopHat2	6,350	5,664	686	86.32	89.20	87.76
		POMP	6,312	5,859	453	89.28	92.82	91.05
7	6,830	TopHat2	7,094	5,838	1,256	85.48	82.29	83.89
		POMP	6,041	5,465	576	80.01	90.45	85.23
8	4,650	TopHat2	4,409	4,009	400	86.22	90.93	88.57
		POMP	4,134	3,919	215	84.28	94.8	89.54
9	6,330	TopHat2	6,157	5,513	644	87.09	89.54	88.32
		HASM	5,907	5,515	392	87.12	93.36	90.24
10	5,482	TopHat2	5,278	4,639	639	84.62	87.89	86.26
		POMP	4,785	4,505	280	82.18	94.15	88.17

Table 3: Genome-wide performance evaluation for Initially Accurate Splice Junctions (IASJ).

Dataset	Read Length	Predicted	True	False	SN (%)	SP (%)	GN (%)
D1	50 bp	103,895	99,025	4,870	70.30	95.31	82.81
D2	75 bp	107,714	103,150	4,564	73.23	95.76	84.50
D3	100 bp	113,734	108,516	5,218	77.04	95.41	86.23

6. Mismatches

It is the mean number of alignment mismatches of all the representatives mapped onto a candidate $J(a, b)$ of interest. Let $\mathcal{R}$ denote the set of representatives mapped on to $J(a, b)$. Suppose each representative contributes m_i mismatches in the gapped alignment. The mismatch score is defined as the average number of mismatches of all the representatives while aligning onto $J(a, b)$, i.e., $M_{score} = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{R}|} m_i$.

7. Shortest mapping length

It is the maximum length of the shorter side of the gapped alignment among all the representatives around a candidate junction $J(a, b)$ of interest.

8. Junction overlapping number: It is defined as the number of UGRs mapped onto a candidate junction $J(a, b)$.

3.2.3 Classifying splice junctions

At the beginning SVM builds a learning model using training examples from IASJs and IISJs. As the representatives in our algorithm are highly accurate and long compared to IUM reads, the number of accurate SJs are larger than those

(a) % Sensitivity of different algorithms

(b) % Specificity of different algorithms

(c) % Gain of different algorithms

(d) Elapsed time of different algorithms

Figure 1: Performance evaluations of different algorithms including POMP for datasets D1-D3.

of inaccurate SJs. If we consider all the SJs from IASJs and IISJs, SVM will produce a biased model. To develop an unbiased model, we randomly pick SJs from IISJs which are nearly equal in number to SJs in IASJs. After building the model we classify each of the candidate SJs as accurate or inaccurate and output accurate SJs.

4. RESULTS

The effectiveness of our algorithm POMP has been evaluated by comparing it with the currently best-known algorithm TopHat2. Although MapSplice, SpliceMap, GSNAP [16], STAR [17], and RUM [18] are some of the state-of-the-art algorithms in this domain, they all produce poorer results compared to TopHat2. So, we have compared our algorithm with TopHat2 considering it as a benchmark. The experimental results show that our proposed algorithm is indeed very effective and competitive. More details follow.

4.1 Experimental datasets

Three datasets each having 40 million reads were simulated. Read lengths of these datasets are 50, 75, and 100 bp long, respectively. Maq [14] was used to generate simulated Illumina reads from human genome hg19 with a realistic error rate of 0.02. The number of reads were also proportional to the human datasets based on UCSC Known Gene Model. Thus the test datasets are nearly similar to the real transcriptome sequencing data. Expression levels were estimated using Cufflinks [15] which was based on isoforms defined by UCSC Known Gene Models.

4.2 Experimental environment

All the experiments were done on an Intel Haswell 2.60 GHz compute node with 24 Intel Xeon E5-2690 cores and 128 GB of RAM. The operating system running was Red Hat Enterprise Linux Server release 5.7 (Tikanga). The POMP algorithm has been implemented in C++ and standard Java programming language. To compile the C++ source code we used g++ compiler (gcc version 4.6.1) with the -O3 op-

tion. Java source code is compiled and run by Java Virtual Machine (JVM) 1.8.0_31.

4.3 Evaluation metrics

We measure the effectiveness of our proposed algorithm POMP using four different metrics. These metrics are defined below.

1. Sensitivity

The fraction of the simulated junctions correctly detected. Let the number of simulated and predicted junctions be T and P , respectively. Now, the Sensitivity can be written as $SN = \frac{P}{T}$.

2. Specificity

The fraction of the true junctions among all predicted junctions. Let the number of true junction detected by the algorithm of interest be N . Specificity now can be formulated as $SP = \frac{P}{N}$.

3. Gain

It is defined as the average of sensitivity and specificity. Please note that both sensitivity and specificity are important measures. Gain is a simple way to combine these two measures. It can be written in terms of SN and SP as $GN = \frac{SP+SN}{2}$.

4. Time

Measured elapsed time using total number of CPU clock cycles consumed by each of the algorithms of interest.

4.4 Outcomes

Consider the dataset D1 of 50 bp reads (please, see Table 1). The overall gains of POMP and TopHat2 are 90.16% and 87.23%, respectively. POMP is approximately 3% better than TopHat2 in this case. The sensitivity and specificity of POMP are better than those of TopHat2. The specificity of POMP and TopHat2 are 93.33% and 88.22%, respectively. POMP has an around 5% better specificity. This means that POMP not only predicts more SJs but also detects true SJs more precisely. The runtime of POMP is also very impressive. It took 17 CPU hours where TopHat2 took 25 CPU hours. We show chromosome wide experimental evaluations of dataset D1 for first 10 chromosomes. Please, see Table 2 for more details. It is evident that chromosome wide gains of POMP are better than those of TopHat2. Results for datasets D2 and D3 are shown in Table 1. The overall genome wide gains of POMP are better than those of TopHat2. POMP also took less time to detect potential splice junctions. POMP took 10 CPU hours whereas TopHat2 took 17 CPU hours for dataset D2. On dataset D3 POMP took 18 CPU hours while TopHat2 took 25 CPU hours. Please, see Figure 1 for visual details.

5. DISCUSSION

To the best of our knowledge POMP is the first algorithm in the domain of splice junction discovery which employed consensus strings (i.e., representatives) of highly overlapping reads coming from the same genomic region. It not only enhances the accuracy of the junction discovery but it also drastically reduces runtime. The limitations of the sequencing technology results in errors in the reads. The errors could be substitution(s), insertion(s) and/or deletion(s)

in a single base or multiple bases. Although the errors are being greatly reduced with the advancement of the modern technology, it is still a serious concern as of today. By identifying and correcting the erroneous bases of the reads, not only can we achieve high quality data but also the computational complexity of many biological applications can be greatly reduced. When we build a representative from a cluster of similar reads, the bases of the representative are chosen by taking the consensus. As a result the representatives are naturally error-free with a very high probability and hence the alignment can be done with a high accuracy.

We choose the overlapping regions of the reads in the clusters in such a way that the lengths of the representatives are no more than 1.2 times of the read length. This ensures that similar reads forming clusters are originating from the same genomic regions. Please note that some clusters can have only a single read. It is due to the fact that the coverage is very low in that region of mRNA. In this case a single read in a cluster forms a representative by definition. Generally representatives are longer than the read and thus splice junctions detections are very accurate. Consider the results shown in Table 3. It is based on the computation of initial accurate splice junctions. Without any statistical analysis and significant computation POMP can detect a subset of splice junctions with a very high level of accuracy. The specificity is always above 95% for all of the datasets. The overall gain for each of the datasets is also very impressive.

Our algorithm POMP detects donor sites and acceptor sites of SJs within a very high resolution. The true boundaries of SJs can be found by POMP within 20 bp resolution. On the other hand Tophat2 detects boundaries of SJs within 50-100 bp resolution. More specifically, let $J(p, q)$ denote a splice junction with p and q being the donor and acceptor sites in the genome G , respectively. Let p' and q' be the outputs of POMP corresponding to this junction. Then POMP guarantees that p' and q' will be bounded by $p - 20 \leq p' \leq p + 20$ and $q - 20 \leq q' \leq q + 20$, respectively. TopHat2 detects donor and acceptor sites of SJs very poorly within 20 bp resolution. For example on the D1 dataset TopHat2 discovers only 42 simulated SJs (out of 140,856 true SJs) within 20 bp resolution. The time complexity of our algorithm is also quite impressive. It runs 60-70% faster than TopHat2 and achieves a better accuracy.

To investigate the effectiveness of our algorithm POMP, we have done extensive experiments by considering 3 simulated RNA-seq datasets of different read lengths. The RNA-seq datasets were generated in accordance with the real transcriptome data by introducing realistic gene and error model. The abundance of the simulated reads are also in proportion to the real reads. Each dataset contains 40 million reads and aligned with the hg19 human genome. Both sensitivity and specificity are important metrics to judge the performance of any algorithm. Thus we need a measure that combines these two into a single measure. Keeping this in mind we have introduced the gain metric as the average of sensitivity and specificity. From the experimental analysis it is evident that POMP is better than TopHat2 in terms of gain and running time. TopHat2 has better sensitivities on D2 and D3 datasets but lower specificities. It indeed detects a lot of alternating splicing events but with greater false positives. On the contrary POMP is more restricted in deciding true alternating splicing events. It balances be-

tween sensitivity and specificity and thus its overall gains are better than those of TopHat2 on all the datasets.

There are two issues in the proposed algorithm that have to be further investigated. At this point (1) POMP does not exploit the advantage of paired-end reads and (2) it divides the representatives into two equal halves. If POMP could exploit the advantage of paired-end reads, the specificity would be further increased. Since in this setting we know the relative distance between two representatives a priori, the gapped alignments will be more accurate and the number of false positives will be drastically reduced. On the other hand a single representative can stretch over two splice junctions. If we divide a representative into two halves and the coverage is low in that region we might potentially miss these junctions. By combining these two ideas we can further increase the sensitivity and specificity of our proposed algorithm. We plan to work on these ideas in future.

6. CONCLUSION

In this article we have proposed a novel algorithm POMP to detect splicing events more accurately with a very short period of time compared to other state-of-the-art algorithms. At the beginning similar sequencing reads from initially unmapped reads are clustered. From individual cluster *representatives* are formed which are naturally more accurate and longer than the given set of reads with a very high level of confidence. These *representatives* are then used to detect the splice junctions. SVM is used to prune candidate splice junctions by classifying them into two classes, i.e., accurate or inaccurate. Experimental results show that POMP is indeed effective and efficient compared to other best-known algorithms in this domain.

Funding

This work has been supported in part by the following grants: NIH R01-LM010101 and NSF 1447711.

Conflict of interest statement.

None declared.

7. REFERENCES

- [1] Trapnell,C., Pachter,L. and Salzberg,S.L. (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, **25**, 1105-1111.
- [2] Langmead,B., Trapnell,C., Pop,M. and Salzberg,S.L. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.
- [3] Au,K.F., Jiang,H., Lin,L., Xing,Y. and Wong,W.H. (2010) Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res.*, **38**, 4570-4578.
- [4] Wang,K., Singh,D., Zeng,Z., Coleman,S.J., Huang,Y., Savich,G.L., He,X., Mieczkowski,P., Grimm,S.A., Perou,C.M. et al. (2010) MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Res.*, **38**, e178.
- [5] Bryant,D.W. Jr, Shen,R., Priest,H.D., Wong,W.K. and Mockler,T.C. (2010) Supersplat: spliced RNA-seq alignment. *Bioinformatics*, **26**, 1500-1505.
- [6] Dimon,M.T., Sorber,K. and DeRisi,J.L. (2010) HMMSplicer: a tool for efficient and sensitive discovery of known and novel splice junctions in RNA-Seq data. *PLoS One*, **5**, e13875.
- [7] Pan,Q., Shai,O., Lee,L.J., Frey,B.J. and Blencowe,B.J. (2008) Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nat. Genet.*, **40**, 1413-1415.
- [8] Jiang,H. and Wong,W. (2008) Seqmap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, **24**, 2395-2396.
- [9] Ameur,A., Wetterbom,A., Feuk,L. and Gyllenstein,U. (2010) Global and unbiased detection of splice junctions from rna-seq data. *Genome Biology*, **11**, R34.
- [10] Kim,D., Pertea,G., Trapnell,C., Pimente,H., Kelley,R. and Salzberg,S.L.. (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, **14**, R36.
- [11] Saha,S. and Rajasekaran,S. (2015) NRRC: A Non-referential Reads Compression Algorithm. *11th International Symposium on Bioinformatics Research and Applications (ISBRA)*, 297-308.
- [12] Vapnik,V. (1995) The Nature of Statistical Learning Theory. Berlin: Springer-Verlag.
- [13] Cortes,C. and Vapnik,V. (1995) Support vector networks. *Mach Learn.*, **20**, 1-25.
- [14] Li,H., Ruan,J. and Durbin,R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18**, 1851-1858.
- [15] Trapnell,C., Williams,B.A., Pertea,G., Mortazavi,A., Kwan,G., van Baren,M.J., Salzberg,S.L., Wold,B.J. and Pachter,L. (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol.*, **28**, 511-515.
- [16] Thomas,D.W. and Serban,N. (2010) Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, **26**, 873-881.
- [17] Dobin,A., Davis,C.A., Schlesinger,F., Drenkow,J., Zaleski,C., Jha,S., Batut,P., Chaisson,M. and Gingeras,T.R. (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15-21.
- [18] Gregory,R.G., Michael,H.F., Angel,D.P., Nicholas,F.L., Jonathan,S., Brian,P.B., Christian,J.S., John,B.H. and Eric,A.P. (2011) Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics*, **27**, 2518-2528.