Abstract

In recent years, the introduction of new DNA sequencing platforms dramatically changed the landscape of genetic studies. These protocols for next-generation sequencing (NGS) are able to generate massive amounts of data, requiring the creation of new computational tools to deal with this data quickly and economically. With the development of the RNA-Seq methodology, which uses the new sequencing protocols to get information about RNA samples, the study of the transcriptome gained a new boost. Problems such as the identification of genes expression levels and alternative splicing can be solved with the assembly and the study of the transcriptome. At the same time, the use of this technology has the great advantage of allowing new biological discoveries and observations. This technology has, however, the downside of requiring a very considerable computational effort. This work aims to present a detailed study about the problem of transcriptome alignment, presenting an efficient computational solution, which requires the development of heuristics to identify splice junctions using methods and data structures for an efficient mapping.

Keywords: transcriptome, RNA-Seq, Next-Generation Sequencing, sequence alignment, splice junctions

1. Introduction

The complete genome sequencing of various organisms, including the human genome, has been in the last decade one of the centres of the attention in the scientific community. All this sequencing effort, which aims at determining which molecules constitute the deoxyribonucleic acid (DNA) results in a large amount of biological information that must be stored in databases, and is also manipulated in order to transform this data into knowledge. From this, a new area of interconnection between molecular biology, statistics and computer science has emerged, the field of Bioinformatics.

The main goal of Bioinformatics is the development of models and tools to efficiently analyze and extract information from the enormous amount of biological data currently available.

For many years, the standard methods for determining the sequence of transcribed genes, consisted in the sequencing of messenger ribonucleic acid (mRNA) using complementary DNA (cDNA) through the conventional method of Sanger [1], using Expressed Sequence Tags [2] or via microarrays [3]. Recently, a new experimental method has appeared, the RNA-Seq method, which has brought many advantages over the other conventional methods. This new method uses Next Generation Sequencing (NGS) technologies, yielding sequences of messenger RNA (mRNA) with fewer errors. This new approach produces more data from each experiment, allowing this data to be used as a direct measure of the gene expression level. RNA-Seq experiments not only capture the transcriptome, i.e., all RNA sequences present in a cell, but they also replace the conventional experiences with microarrays. One of the critical steps in a RNA-Seq experiment is the mapping of the reads, generated by NGS technologies, to a reference genome.

Currently, there are several tools developed to make the direct alignment of these fragments in a reference genome. However, due to the splicing phenomenon of eukaryotic cells, that regards a transcript that may contain parts of more than one exon, these tools may fail, since the case in which a read span to more than one exon is not covered. In order to achieve these fragments alignment, it is necessary to develop methods that can identify the splice junctions, that is, boundaries between exons and introns, thereby allowing a correct identification of the source of the transcript in question. This work aims at analysing models and methods to identify splice junctions, which is essential to the problem of aligning the transcriptome, and suggests an efficient computational solution for that.
1.1. RNA-Seq

RNA-Seq [4] is a recently developed approach to transcriptome profiling that uses deep-sequencing technologies. In general, a population of RNA (total or fractionated) is converted to a library of cDNA fragments with adaptors attached to one or both ends. Each molecule, with or without amplification, is then sequenced in a high-throughput manner to obtain short sequences from one end (single-end sequencing) or both ends (pair-end sequencing). The reads are typically 30–400 bp, depending on the DNA-sequencing technology used.

After the sequencing, the reads are mapped against a reference genome or a reference transcriptome. They can also be assembled de novo [5, 6], that is, without a reference. The main goal is to produce a transcriptional map in genomic scale with the transcriptional structure and the expression level of each gene.

RNA-Seq is probably one of the most complex next-generation applications. Expression levels, differential splicing, allele-specific expression, RNA editing and fusion transcripts constitute important information when comparing samples for disease-related studies. This brings many advantages over the existing approaches. First, unlike hybridization-based technologies, RNA-Seq is not limited to the detection of known transcripts, thus allowing the identification, characterization and quantification of new splice isoforms. In addition, it allows researchers to determine the correct gene annotation, as well as defining - at single nucleotide resolution - the transcriptional boundaries of genes and the expressed Single Nucleotide Polymorphisms (SNPs).

2. Background

A new protocol for sequencing the messenger RNA in a cell, RNA-Seq, generates millions of short sequence fragments in a single run. These fragments, or reads, can be used to measure levels of gene expression and to identify novel splice variants of genes.

One of the critical steps in an RNA-Seq experiment is mapping NGS reads to the reference transcriptome. However, because transcriptomes are incomplete even when it comes to well studied species such as human and mouse, RNA-Seq analyses are forced to map to the reference genome as a proxy for the transcriptome.

When sequencing a new organism, if a sufficiently close sequence is already known, one may choose to use it as a reference and start by first mapping the reads to this reference and then determining the new sequence, by extracting the consensus from the mapping results.

In a typical application, we may have to align hundreds of millions of reads to a reference genome that can be as large as few gigabases. This is a job that cannot be efficiently achieved through standard dynamic programming procedures.

One way to speed up the read alignment task is resorting to approximate indexing techniques. A first generation of aligners was based on hash tables of k-mers. Some of them build tables of k-mers of the target sequence, whilst others index the reads, thus presumably requiring re-indexing for each new run. Recent developments in the field of compressed approximate indexes have led to a new family of alignment algorithms, which uses an enhanced suffix array [15], and BWA-SW [7], which uses a FM-index [17] to accelerate Smith-Waterman [18] alignments.

Because RNA-Seq reads are short, the first task is challenging. Current mapping strategies [7, 8, 9] include alignment procedures designed to localize Illumina, SOLiD or 454 reads directly in the genome. However, whenever an RNA-Seq read spans an exon boundary (Figure 1), part of the read will not map contiguously to the reference, which causes the mapping procedure for that read to fail.

There are other tools that manage to align RNA-Seq reads [11, 12] (for more consult [13]) but only for short reads (<100 bp) and based on the Illumina or SOLiD technologies.

All of this tools find junctions by mapping reads to the reference in two phases. In the first phase, the pipeline maps all reads to the reference genome using Bowtie [8]. All reads that do not map to the genome are set aside as initially unmapped reads, or IUM reads. Then, they use heuristics and measures values to decide where to align them.

In this project, we aim to create a new approach, based on only one phase, using TAPyR [9] and compare that performance with MapSplice [12].

2.1. TAPyR

TAPyR is a method for the alignment of pyrosequencing reads, like those produced by the 454 GS FLX platform. By focusing on this specific technology, the procedure manages to explore data characteristics to achieve improved performance over other mainstream methods. Like many of those methods, like BWA [7] or Bowtie [8], TAPyR also builds an index of the reference sequence to accelerate the alignment. It then employs a multiple seed heuristic to anchor the best candidate alignment. Contrary to other seed-based alignment tools, TAPyR’s strategy adds more flexibility by dispensing with the need of determining the number and length of the seeds beforehand. The heuristic relies on some assumptions that can be reasonably expected to hold true for re-sequencing projects based on pyrosequencing data, namely, that the op-
timal alignments are mostly composed of relatively large chunks of exact matches interspersed by small, possibly gapped, divergent regions. A banded dynamic programming is used to finish up the candidate multiple seed alignments, considering user-specified error constraints.

The choice of TAPyR has many reasons. One of them is because the number of aligners that support GS FLX pyrosequencing data is, as of today, relatively scarce compared to other technologies, most notably Illumina. Moreover, some of these tools find their origins before the advent of the new sequencing technologies were only adapted later to cope with new kinds of data, as well as some others target multiple kinds of data which is not optimized for pyrosequencing data. Given this state of affairs, it’s reasonable to think that there is still room for improvement in the realm of available aligners specifically designed for high-throughput pyrosequencing data [9].

Taking advantage from all the characteristics of this computational model, the TAPyR adaptation for the transcriptome alignment consisted in the creation of a new module that is executed in an opportune moment, reporting the existence of splice evidence, if there is any. This is done in a sequential manner and the splice signals considered by this implementation are: “GT-AG”, “GC-AG” e “AT-AC” [10].

Until now, TAPyR could only align reads that had a direct match in a reference genome. Now, with the creation of a new module to identify splice junctions, TAPyR can also align RNA-Seq reads. This means it’s able to align reads that span over more than on exon.

Later will be explained how that aspect was implemented (see Approach).

2.2. MapSplice

The goal of MapSplice [12] is to find the exon splice junctions present in the sampled mRNA transcriptome, and to determine the most likely alignment of each mRNA sequence tag to a reference genome. Each tag corresponds to a number of consecutive nucleotides read from an mRNA transcript, where the length of the tag is determined by the protocol and the sequencing technology.

MapSplice operates in two phases to achieve its goal. In the “tag alignment” phase, candidate alignments of the mRNA tags to the reference genome G are determined. Tags with a contiguous alignment fall within an exon and can be mapped directly to G but tags that include one or more splice junctions require a gapped alignment, with each gap corresponding to an intron spliced out during transcription. Since multiple possible alignments may be found, the result of this phase is, in general, a set of candidate alignments for each tag.

In the “splice inference phase”, splice junctions that appear in the alignments of one or more tags are analysed to determine a splice significance score based on the quality and diversity of alignments that include the splice. For more about MapSplice see [12].

3. Approach

One of the initial goals of this project was to put TAPyR in a pipeline for transcriptome alignment to see how much improvement could be found in the performance. After this, heuristics would be developed, allowing to take advantage from that improvement. That was not possible [13].

Not being able to embed TAPyR in a pipeline, the next option was to create a new and innovative approach - to have a tool that, by an user option, would align DNA data or RNA data. Here, we break the pattern of creating a pipeline to align RNA-Seq data, and address this problem to a sequential workflow.

For TAPyR to be able to align RNA data in a sequential manner, some changes had to be made in the original algorithm. That changes are highlighted with “⇒”.

The reads processing is done the following way:

• We start from one of the ends of the read in question and then we search for each character, one by one, in the index of the reference. When we reach a character where the substring without that character exists in the index, but with that character no longer exists exists anymore, we define that substring as a seed (Figure 1).

• Then, we jump that character and we start the construction of a new seed.

• This process is repeated until we reach the other end of the read.

Once this processing is done, we get all the occurrences, i.e., all the positions in the reference sequence of the reads, proceeding with their sort by position. From this, we get a sorted list of occurrences that is analysed in order to identify the seeds that appear in the same order and with similar distances in both the read and the reference genome. ⇒ If the reads are from RNA-Seq, when the seeds don’t have a similar distance in both read and reference, we may have an intron between them, so
we’ll look for splice evidence in the read. The distance that separates the seeds is analysed and if it is in the range of an intron size (defined by the user or the default values - from 50bp to 500000bp), we execute the splice junctions detection module.

The set of occurrences that satisfy those conditions form chains (Figure 2), meaning, potential candidates of read occurrences in the genome. The chains whose sum of seed’s lengths is the highest, are selected and a dynamic programming is done in the spaces between the seeds so the number of errors can be identified. If that number, from a chain, is less than the maximum number of errors, the chain position is written in a SAM [14] file.

If the reads are from RNA data, and we identified a splice junction, the dynamic programming isn’t done and we immediately indicate the existence of an intron in the SAM file with the code “N”, as well as the size the size of the space between those seeds.

One characteristic from this implementation is that the chains don’t need to contain all the seeds, allowing to have just some of them. For example, one read may have originated three seeds, but the chain only contains the fist and the third. This has the consequence that we may have to resort to dynamic programming in the space where the second seed should be.

When sequencing RNA, if an intron appears before or after one missing seed, we can’t align that read because we would have to do dynamic programming in both extremes to see in which side the characters of the read would match.

If one read does not have any error, it gives origin to only one seed and for that will be just one chain for each occurrence in the genome.

This is the TAPyR workflow in a general view (Figure 3). Next we will explain how the splice junction detection module works.

3.1. Splice Junction Detection

The function that identifies the splice junction receives six arguments: the genome, the read, the position where the gap in the read starts, the position where it ends, the position where the the gap in the genome starts and the position where it ends. Then we calculate the distances between both gaps - the gapSizeInRead and the gapSizeInGenome. We may have two cases (Figure 4):
• **Case 1** - gapSizeInRead = 0.

• **Case 2** - gapSizeInRead > 0.

Case 1:

- The algorithm will only verify if there is any evidence of splicing in the genome. Starting in the first position of the gap in the genome, it looks for one of the splice signals and if there is one, then he looks for the match in the final position of the gap in the genome. Having both signals, the algorithm reports that there is evidence of splicing.

Case 2:

- When gapSizeInRead > 0, it means that some part of the read was not aligned, so we must know if it has a match in one of the sides of the genome.
  - We begin in the start position of the gap in the genome looking for all the matches between the read and the genome.
  - When we have the first mismatch or if there are no more characters to align, if gapSizeInRead > 0, we try to continue the alignment, but now starting from the position where the gap in the genome ends from the right to the left.
  - When the alignment is finished, we look for splice signals and if they exist we report that there is evidence of splicing.

If after these two cases no reports of splicing could be found, we do a last verification. TAPyR constructs their seeds from the left to the right in the read, and because of that, it may happen that the second seed from that same read may have characters that belong to the first read. That is, the first characters from the second seed have exact matches in two places in the genome: in the end of the first seed and where the second seed has its match. Because this takes place several times, we notice that if that small amount of characters were aligned as being part of the first seed we would find evidence of splicing in the new places of the start and the end of the gap in the genome.

To solve this, the algorithm tries to align the first characters of the second seed next to the first one in the genome. The alignment is done until we find the first mismatch and then we look for the splice signals. If they exist, we report the splicing evidence.

For all the situations mentioned above, when we look for splice signals we must consider three cases (Figure 5):

- **Case 1** - The splice junctions take place in the position where the gap in the genome starts and ends.
- **Case 2** - The splice junctions take place in one position on the left where the gap in the genome starts and ends.
- **Case 3** - The splice junctions take place in one position on the right where the gap in the genome starts and ends.

Case 1 it’s always the first to be executed because it’s where the major part of the splice signals are. If this case fails we test cases 2 and 3.

Why do we do this? Because when we have the pairs “GT-AG” and “GC-AG” both first character of the first dinucleotide and second character of the second dinucleotide are the same. Because of that, we may have a problem when in the genome we have something like ...GGTXXXXXXXXXAG... and if the second seed has a “G” for the first character.

Despite being obvious that this read comes from two different exons (the splice signals are there), the splice junction isn’t identified. This can be extended to a general case creating some kind of neighbourhood, with just one position as a gap, where the splice signals may be present. We can justify this arguing that when a read is separated in seed, for each time that we have a mismatch, we jump one character and this character can also may be a match or an indel. With this we cover all the cases without putting at stake the quality of the alignment.

### 3.2. Advantages

This implementation provides many advantages that are crucial to a good performance. The first great advantage, and this has a great impact in the running time, is the fact that this is the first approach to analyse the set of reads just once, unlike the other tools for transcriptome alignment that have two phases and re-process the reads that weren’t aligned.

Other major aspect is that, like some of the most competitive tools in this area, TAPyR doesn’t require any splice junctions model, any annotation or any junctions library. This allows to discover new splice sites without any influence.

The way this problem was studied and implemented, resulted in a simple but promising tool.
Because the splice junctions detection consists in one module that is called only when necessary, this does not change the normal workflow of TAPyR while giving us a great flexibility in case we want to change the module. Modifying or adding something to the module requires the editing of a single file. We can take another advantage from this - TAPyR is the first accessible tool able to align both DNA and RNA data. If you want to align RNA data you just have to indicate that you wish to do so in the execution command line.

At last, the integration of the new module on TAPyR doesn’t change its complexity, which is linear. In TAPyR, the construction of the index is linear to the size of the reference; the search in the index is linear to the size of the read; the alignment algorithm, using seeds, is linear to the size of the reads. The splice junctions detection module doesn’t add any complexity because it just verifies the distances between the seeds that TAPyR already calculated. In the worst case, the module has complexity of $O(n)$, where $n$ is the gapSizeInRead.

4. Results

When developing any engineering project, the results have always a preponderant role. Besides this, the solution must be efficient, economic and portable. Given the context of this work, the efficiency is important because we are dealing with a massive amount of data that has to be dealt with care and the less effort we have to put into it, the better. Being economic means that we must use the smallest possible amount of resources in terms of time and space. Like any other program, this must be portable and accessible, being also possible to execute it in any computational system.

At biological level, it’s important to obtain credible results that might help the comprehension of the problem’s domain. In this specific case, the detection of the transcripton boundaries revealed itself of extreme importance to the identification of transcripts and events of alternative splicing that may be associated with diseases, etc.

In the next subsections all the elements involved in the tests made to the new version of TAPyR are described in detail. This tests were made in a 64 bits machine with an Intel Core i7 - 3620QM 2.3 GHz, with 6GB of RAM.

4.1. Data

TAPyR is more efficient when performed with reads with at least 100bp, so it makes sense use this size of reads to test the new version. To be easier to compare the performance of TAPyR and to find a good balance between TAPyR and Mapsplice, the dataset used in the tests was extracted from the Mapsplice’s website.

The reference genome we used is the chromosome 20 from the human genome in the fasta format with 63.7MB. The set of reads are from the Illumina platform and contain about 9999991 reads with 100bp each.

Although TAPyR has been designed to align reads from 454 technology, if the reads have at least 100bp, its behaviour is good, so this dataset gives us the possibility to make a direct comparison between this new version of TAPyR and Mapsplice.

4.2. Performance

The evaluation criteria were: size of the index, number of aligned reads, error statistic and running time.

TAPyR was tested with the following properties:
- **Size of the Index** = 70.2 MB
- **Minimum Identity Percentage (MIP)** = 80% and 90%
- **Minimum Intron Size** = 50bp
- **Maximum Intron Size** = 500000bp

The results were:

**Case 1** - MIP = 80%; Mode = DNA:
- **Number of Aligned Reads** = 902006 de 999991 (90.201%)
- **Running Time** = 15 seconds (62718.954 reads/s)
- **Error Rate** = 1 error/27bp

**Case 2** - MIP = 80%; Mode = RNA:
- **Number of Aligned Reads** = 996879 de 999991 (99.689%)
- **Running Time** = 12 seconds (80931.613 reads/s)
- **Error Rate** = 1 error/116bp

**Case 3** - MIP = 90%; Mode = DNA:
- **Number of Aligned Reads** = 745499 de 999991 (74.551%)
- **Running Time** = 16 seconds (61708.794 reads/s)
- **Error Rate** = 1 error/93bp

**Case 4** - MIP = 90%; Mode = RNA:
- **Number of Aligned Reads** = 989172 de 999991 (98.918%)
- **Running Time** = 12 seconds (80931.613 reads/s)
- **Error Rate** = 1 error/133bp

This statistic data was given by TAPyR in the end of each running. From these results we can observe that the new module of splice junctions detection improves the performance of TAPyR giving more accuracy at the alignment level.

Even with a high percentage of alignment in Case 1, we can conclude that the quality of the alignment
may be compromised because the number of errors allowed is large and the error statistic shows that (1 error per 27bp). With that same percentage of identity, but in RNA mode, in Case 2 we can see that there was a considerable improvement in the alignment percentage and the error rate was less than in DNA mode (1 error per 116bp).

The major and the most significant difference between the two versions is observed when we increase the identity percentage to 90%. Looking at Case 3 we can see that the number of reads aligned dramatically decreased but so did the error rate when comparing with Case 1 (both in DNA mode). In RNA mode, there isn’t a big difference between Case 2 and Case 4. There were fewer reads aligned but so the error rate was equally less too.

In Figure 6 we have an example in a region of the reference genome, about the difference in the number of reads aligned in DNA and RNA mode. The images were generated by TAPyR and the represented sequence corresponds to the interval of 275000 and 365000 position. In the first image, we have the TAPyR alignment in DNA mode, where there are 992 aligned reads and in the second image we have the TAPyR alignment in RNA mode where there are 1280 aligned reads. The data about the number of aligned reads was also given by TAPyR.

If we look carefully at the images and compare them, we can see that even more important than align more reads, the RNA mode alignment has more read coverage (green lines) and so we can identify the exonic regions even better.

The main conclusion is that this new version of TAPyR, with the option of RNA data alignment, brings out more capacity and flexibility to align both genomes and transcriptomes. The number of reads aligned is very good and the running time is far better than the existing state of the art software.

5. Discussion

MapSplice is, nowadays, one of the most competitive tools to align transcriptome [12] so we decided to compare the number of reads aligned and the running time of MapSplice with the ones by TAPyR.

MapSplice uses Bowtie in a first phase, which is to do the direct alignment of the read and only then it tries to align the other ones, factoring them in smaller pieces. This process takes more time than TAPyR because MapSplice has to execute a second process to the set of reads.

For the same data used in the previous section, with an index generated by Bowtie with 72.1MB, the performance of TAPyR was:

- **Number of Aligned Reads** = 985955 de 999991 (98.59%)
Running Time = 2 minutes e 38 seconds

Although the number of the aligned reads is close to the one of TAPyR in RNA mode, the running time is much worse. One possible cause for this is because this version of TAPyR is a sequential one: we only process the reads once and even when we call the splice junctions detection module we don’t have to do new calculations or any others processes because we use the information already known by the main algorithm.

These facts bring a new impulse to the alignment of the transcriptome because, as it will be much faster align transcriptomes even in large genomes and/or with a large amount of reads.

6. Conclusions

Since the beginning of this work, the importance of the transcriptome sequencing have been often present. With the NGS technologies and the evolution of computational tools, all sequencing problems have been evolving very fast. However there is always room for improvement in the computational branch.

The main goal of this work was to improve the area of transcriptome alignment. Even though there are already tools for the identification of splice junctions, their alignment is still a bit slow and they can’t align a considerable amount of reads.

The module developed for the identification of splicing events allows TAPyR to enter, with a competitive profile, in the group of algorithms for aligning RNA data. This new approach breaks the idea that we have to separate the reads in two groups, aligned and unaligned, in order to achieve their alignment and that one single tool can do both things - align DNA and RNA - without compromising each others performance. The running time is better when compared to other methods and even the number of reads aligned is at least as good as the ones performed by other applications.

This can have a major impact when sequencing very large transcriptomes and if it comes to insert this algorithm in some pipeline of transcriptome assembly. The final result of this project is positive.

6.1. Future Work

Transcriptome alignment is just one of the very steps in the big world that transcriptomics is. A number of improvements can be made in order to measure the quality of the alignment, which is not being done at the moment. One possible approach to change that is using the read coverage in a certain region to be sure that we are in the presence of a splicing event or not.

Beyond the little improvements that can be made to the module that identifies the splicing events, another use could be given to TAPyR - the assembly of transcriptomes. There are already some tools able to do that [19], using tools similar to the one developed in this work, and with the results we obtained here, this version of TAPyR probably would have good results in an application like that.

This area is, without a single doubt, fascinating and has a great impact in the molecular biology and genetic worlds. The stagnation of improvements in this area will hardly happen any time soon.

Acknowledgements

I wish to thank most of all to my supervisor, Alexandre Francisco, for all the support and attention given in the last year. He never gave up on me and here is the result of that belief.

I want also to thank a lot to Francisco Fernandes for all the help when I needed most.

And finally to my family, my friends and the researchers Paulo Fonseca e Ana Teresa Freitas for the support.

Funding

This work was partially supported by FCT through the project TAGS PTDC/EIA-EIA/112283/2009.

References


