New developments in KisSplice: Combining local and global transcriptome assemblers to decipher splicing in RNA-seq data

Alice JULIEN-LAFERRIÈRE[1,2,*], Gustavo AT SACOMOTO[1,2], Rayan CHIKHI[3], Erwan SCAON[3], David PARSONS[2], Marie-France SAGOT[1,2], Pierre PETERLONGO[3], Vincent MIELE[1] and Vincent LACROIX[1,2,*]

[1] Laboratoire de Biométrie et Biologie Evolutive, Université de Lyon, Université Lyon 1, CNRS, UMR5558; Villeurbanne, France.
[2] BAMBOO, INRIA Grenoble Rhône-Alpes, France
[3] Centre de recherche INRIA Rennes - Bretagne Atlantique, IRISA, Campus universitaire de Beaulieu, Rennes, France
*Corresponding authors: alice.julien-laferriere@univ-lyon1.fr, vincent.lacroix@univ-lyon1.fr

Abstract *RNA-seq is deeply changing our way to study transcriptomes. The ultimate goal is to be able to identify and quantify all RNAs present in a sample, even without any prior knowledge of the reference genome, which enables to apply this technology to both model and non model species. However, transcriptome assembly is a difficult task, in particular in the presence of alternative splicing. Two main routes have been followed so far. On the one hand, general purpose transcriptome assemblers [1,2,3] aim at reconstructing all alternative transcripts, but, in order to cope with the inherent combinatorial explosion of this problem, they introduce heuristics which lead them to output only a subset of them (the longest ones). On the other hand, local transcriptome assemblers [4] aim at cataloguing systematically and exactly all the splicing events of a gene but do not provide the full length transcripts. In this work, we propose a pipeline that combines the advantages of both. In practice, we map the output of our local assembler KisSplice [4] to the output of Trinity [1] using GEM [5] and propose a visualisation of the results using IGV [6]. We also report a major improvement of the memory performances of KisSplice upon its previous release, thanks to the integration of Minia [7] for the construction of the de Bruijn graph.*

Availability *http://kissplice.prabi.fr*

Keywords RNA-seq, alternative splicing, graph algorithms, de-bruijn graph, transcriptome, visualisation

1 The model

KisSplice is based on the fact that polymorphisms [1] in RNA-seq data create a specific pattern into a *de Bruijn graph* (DBG) associated to the reads of the datasets. For a given value of k, a DBG is a directed graph where each node represents a sequence of length k, called *k-mer*, present in the reads. There is a directed edge between two nodes if the corresponding k-mers overlap by $k-1$ nucleotides. The pattern created by polymorphisms in a DBG is called a *bubble*. It corresponds to a pair of (internally) node-disjoint paths with common endpoints.

Fig. 1 shows this non linear structure for an alternative splicing (AS) event. Here the inclusion or exclusion of S creates two paths leading from a to b. The common sites are the two extreme nodes, the shorter path corresponds to junction ab whereas the longer path corresponds to the variable part S and the junctions aS, Sb. The shorter path has a predictable length of $2k-2$ nucleotides (concatenation of the $k-1$ nucleotides covering a junction).

1. In this work, we use the term polymorphism when there is an event (at the genomic or transcriptomic level) creating variants in the transcriptome. This covers SNPs, genomic indels and alternative splicing events.

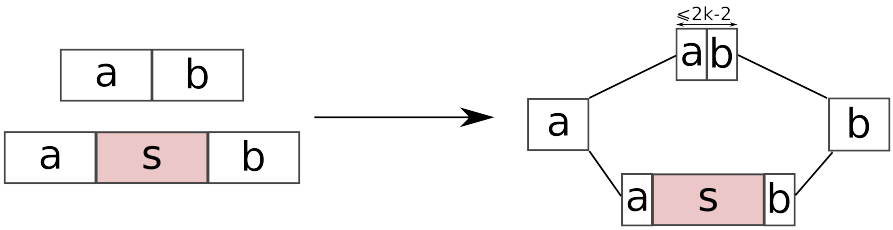

Figure 1. Example of a bubble created by an alternative splicing event.

AS bubbles may correspond to exon skipping, intron retention or alternative donor/acceptor site. They however do not cover mutually exclusive exons (the length of the shorter path is larger than $2k-2$) or alternative transcription start/end (the bubble does not close).

Other types of bubbles caused by SNPs, indels and repeats are reported by `KisSplice`, but in the present work we will focus only on AS bubbles, that is, bubbles with a shorter path of at most $2k-2$ nt and a difference of length between the two paths of at least 3nt [2].

2 Algorithms at a glance

`KisSplice` identifies and quantifies the polymorphisms in RNA-seq data without a reference genome. A detailed description of the algorithms used in the pipeline are presented in [4]. The pipeline is composed of six main steps: de Bruijn graph construction, biconnected component decomposition, four-nodes compression, bubble enumeration, bubble filtration and classification, read coherence checking and coverage computation.

A polymorphism in the reads creates a non-linear structure in the DBG: a pair of (internally) node-disjoint paths with common endpoints. Note that, when the direction of the edges is disregarded, a bubble corresponds exactly to a simple cycle. Thus in the DBG, ignoring the directions, for any two nodes there is a bubble containing them only if they are in the same *biconnected component* (*BCC*, maximal subgraph such that there is a cycle between any two nodes). Therefore, after the DBG construction, `KisSplice` performs a decomposition into BCCs, with an appropriate graph-traversal algorithm. Since each BCC is completely independent of the others, from now on they are treated in parallel. The next step is four-nodes compression: non-branching bubbles due to SNPs and sequencing errors are detected and compressed, i.e. the two alternative paths are merged into a consensus one. Afterwards, `KisSplice` enumerates the remaining bubbles in each BCC and classifies them into four types of events: alternative splicing (AS), SNPs, small indels and approximate tandem repeats. Finally, the reads are mapped back to each bubble. A bubble is said to be *coherent* if each nucleotide is covered by at least one read. Non-coherent bubbles are discarded, the remaining bubbles are kept and the number of reads mapping to each of them is reported.

Therefore, `KisSplice` uses efficient algorithms to directly output polymorphisms present in RNA-seq data using a DBG structure. Nevertheless it does not reconstruct the full transcriptome, only outputting a context of $2(k-1)$ nucleotides for each bubble.

3 Implementation

One of the big challenges of this decade in Bioinformatics is not only to propose algorithms, but also to provide efficient and usable implementations. `KisSplice` follows this line where a specific effort has been made to make it efficient and convenient.

2. To be more precise, AS bubbles also include some indels. In human transcribed regions, 85% of indels concern 1 or 2nt [9]. On the other hand, 99% of AS events are longer than 3nt. If the purpose is really to exclude indels, at the expense of missing some AS events, we recommend to use a threshold of 10nt. Our focus here is to report most AS events. Clearly, between 3 and 10nt, the situation is ambiguous.

3.1 Efficient computational footprint

Newest versions of `KisSplice` (from `1.8.0`) integrate improvements to allow any biologist to run it without requiring large computational resources (in terms of time and space). A typical run of `KisSplice` on 100M reads requires only 5GB of RAM.

The de Bruijn graph construction is performed using `Minia` (`http://minia.genouest.org`), which is the up-to-date reference in terms of memory consumption [7,8]. The core code relies on a compact representation of the de Bruijn graph using a Bloom filter, that allows efficient implementation of graph traversal algorithms. `KisSplice` alternates sequential and parallel sections, and future releases will present a significant decrease of the sequential part. `KisSplice` is mainly implemented in C/C++ and the pipeline is driven by a high-level Python "orchestra conductor".

3.2 Pipeline combining `KisSplice` with a reference transcriptome: `KisSplice2IGV`

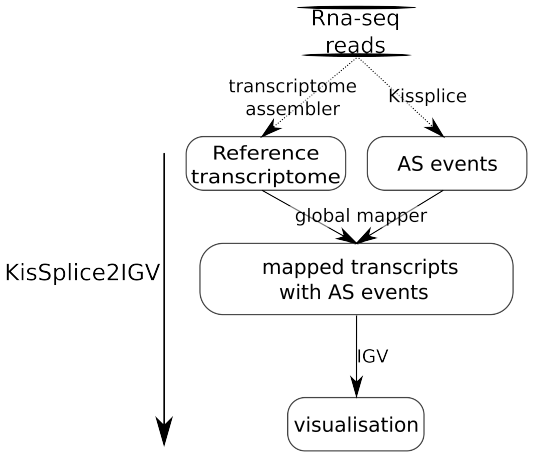

Figure 2. Pipeline proposed in the `KisSplice` suite

In this section we propose a new post-treatment of the output of `KisSplice`, combining the sensitivity of `KisSplice` to detect AS events with the longer context given by a full-length transcriptome assembler.

As shown in Fig. 2, we use a transcriptome assembler (`Trinity`, `Oases`, ...) to infer a reference transcriptome. Alternatively, any reference transcriptome, possibly assembled using other datasets, can be used. The alternative splicing (AS) events reported by `KisSplice` are then mapped to this reference transcriptome using the GEM software [5]. A bubble is considered mapped if at least one path maps to the transcripts. The gapped alignment of the other path can always be deduced from the alignment of the first path. For visualisation, we represent both.

In the case of complex events, involving more than 2 alternative transcripts with a common splice site, `KisSplice` will report all pairs of splice site choices, that is, all pairs of node-disjoint paths in the graph. For instance, if the 4 exons A, B, C and D can be combined in 3 transcripts ABCD, ACD and AD, we will report all pairwise AS events (ABCD Vs ACD, ABCD Vs AD and ACD Vs AD). A path may therefore belong to several bubbles. For instance, the path AD will be present in two bubbles. In the visualisation, we represent it only once. Finally we estimate the coverage of each variant using the read count obtained with `KisSplice` and compute the *reads per kilobase per million mapped read* (*RPKM*), using the following formula $RPKM = RC/((L+r-2k+1)RD)$, with L the fragment length, r the read length, k the k-mer size and RD the read depth. The rationale for the coefficient $(L+r-2k+1)$ comes from the fact that only $L-r+1$ reads can fit in a fragment of length L, while $r-1$ reads can overlap this fragment on each side. Since only reads overlapping the fragment with a sufficient length should be accounted for (at least k nt, to ensure that the read covers the variable region of the bubble), $k-1$ are discarded on each side, hence $L-r+1+2(r-1)-2(k-1) = L+r-2k+1$. We represent the alignments using a colour scale depending on the RPKM : the darker the colour is, the more expressed is the gene. An example of the results obtained is shown in Fig. 3.

This post-treatment scheme is implemented as a stand-alone tool `KisSplice2IGV` available on the `KisSplice` website. `KisSplice2IGV` allow direct visualization of the `KisSplice` output along with the results of a transcriptome assembler (or any reference transcriptome obtained independently).

4 Calling alternative splicing events in human

We test our pipeline on RNA-seq data from human. Even though we do not require a reference genome to run `KisSplice`, we chose to present the results in the case where one is available because it allows to both discuss the results more in depth and show that `KisSplice` can also be used when a reference genome is available.

We used two sets of reads from the Illumina Body Map 2.0 Project. They consist of 32 M reads from human brain and 39 M reads from liver. Both `KisSplice` and `Trinity` were run with default parameters ($k = 25$). `Trinity` found 52804 components (i.e. genes or gene fragments), out of which 4784 were predicted to have alternative transcripts (both resulting from alternative splicing or alternative transcription). In 3227 cases, `Trinity` outputs only one transcript, while `KisSplice` reported at least one AS event. As we had previously shown in [4], `Trinity` is indeed less sensitive than `KisSplice`and tends to understimate alternative splicing.

On the other hand, there are also some AS events that we know `KisSplice` fails to report. They correspond to AS events in genes flanked by repeats. If the same repeats are shared by many transcribed regions, these AS events are "trapped" in very large biconnected components, for which the enumeration does not terminate. Our current strategy to recover these cases is to increase k and thereby solve more repeats. Clearly, an increase of k leads to a loss a sensitivity. Combining the results obtained for several values of k is therefore a promising strategy, that we did not implement yet.

Fig. 3 is such an example where `Trinity` assembled only one long transcript whereas `KisSplice` detected 3 AS events. Fig. 4 shows the same example, but mapped to the reference genome. Clearly, when `KisSplice` is applied to a non model species, this second visualisation cannot be obtained, but we use it here to explain more in depth the results we obtain.

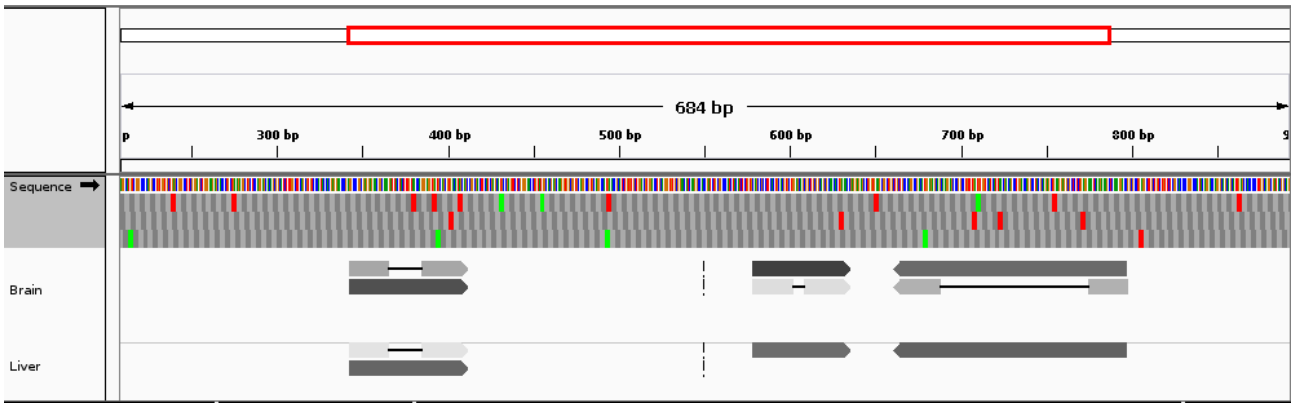

Figure 3. Visualisation of the alternative splicing events found by `KisSplice` aligned to one `Trinity` transcript with `IGV`. The first track (named sequence) shows the reference sequence (the `Trinity` transcript) and the three possible open-reading frames (ORF) for the translation. Green squares are initiation codons and red ones are stop codons. Then the two following tracks represent the alignment results, the events reported by `KisSplice`. The colour of an alignment depends on the $\log_{10}$(RPKM). The darker the more expressed. The upper track corresponds to the transcripts found in brain, the lower one to the transcripts found in liver. A thin line indicates a gap in the alignment. Notice that the strand of the `KisSplice` output is not informative.

The new splice variants discovered by `KisSplice` and missed by Trinity clearly correspond to minor isoforms (0.7 to 3 RPKM for novel splice variants versus 21 to 75 for major variant). However, even if they are minor, it is still possible to show that the last one (downstream) is tissue specific (Fisher test, p-value $= 0.00239$). The 3 events use canonical splice sites (GT-AG). In the presence of an annotated reference genome, we can further check if these variants are annotated (Fig. 4). In this case, only one out of the 3 events is annotated

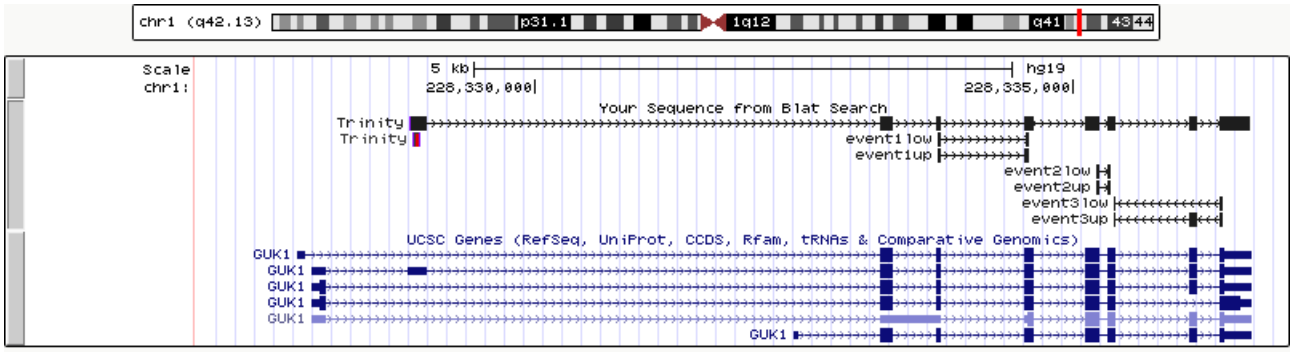

Figure 4. Both `Trinity` and `KisSplice` output mapped to the annotated reference genome. Only the third event was annotated.

in UCSC while the 2 others are only confirmed by ESTs (not shown). In the presence of a reference genome, we can also subclassify the events: one of them is an exon skipping event, while the 2 others are alternative acceptor sites.

Furthermore, two of the detected splice variants include variable regions of length not multiple of 3, hence potentially shifting the reading frame. While these events may seem strange at first sight, a more thorough inspection of the ORF (as suggested by the IGV track) reveals that there is an alternative methionine (green) downstream the AS event 1. This methionine could very well be used as an initiation codon, which would then mean that the event falls within a UTR, hence not shifting any reading frame. In this case, this hypothesis seems to be very relevant since the annotations indeed contain an alternative transcript which uses this methionine as an alternative translation start (see Fig. 5). A similar reasoning can be applied to the third event, since the length of the skipped exon is not a multiple of 3. This time, the skipped exon is located at the very end of the ORF, and it does correspond to a shift in the reading frame which ultimately yields a longer protein. These findings actually suggest that finding novel AS events whose length are not multiples of 3 could be used to improve the annotation of alternative translation start and end.

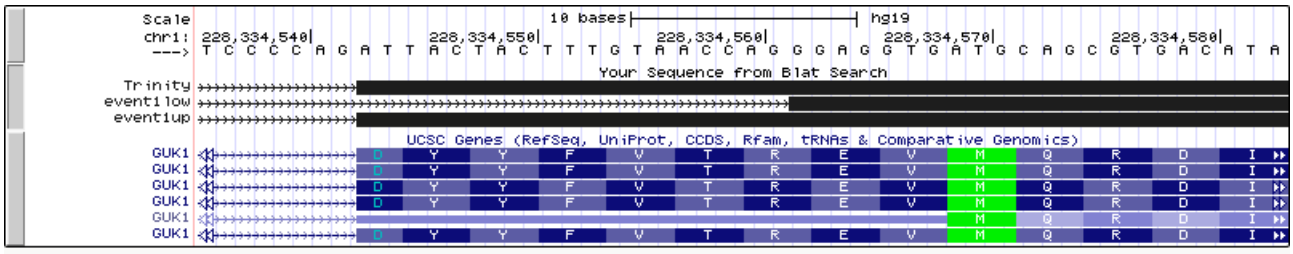

Figure 5. Zoom on the first event. In green the possible alternative start codon. The alternative start was already annotated as it can be seen in the 5th transcript.

5 Conclusion

We presented a pipeline which enables to analyse the results of `KisSplice` when a reference transcriptome is available. Parts of this pipeline are still under development to be fully automatised but our initial analysis shows that, at least for this gene, events missed by `Trinity` have canonical splice sites, are seen in the annotations or ESTs, and may modify the protein. We had come to similar conclusions in [4] but not with such a level of detail and not with the possibility to visualise the results. We hope that the possibility to visualise the results in a genome browser will help non expert users to easily use `KisSplice` for their RNA-seq analysis. Finally, `KisSplice` can also detect SNPs and indels (there were predicted SNPs for this gene, but we did not include them). In the future, we plan to automatise as well the visualisation of these other types of polymorphism. The joint analysis of both genomic and transcriptomic polymorphisms should prove very useful.

Acknowledgements

We would like to thank Gunter Roeth (Bull) for interesting discussions about code optimization, Vincent Navratil (Prabi) and Tristan Lefebure (LEHNA) for having deeply tested `KisSplice` on their datasets. This work was funded by the ANR-12-BS02-0008 (Colib'read). The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no [247273]10.

References

[1] Grabherr MG., Haas BJ., Yassour M., Levin JZ., Thompson DA., Amit I., Adiconis X., Lin F., Raychowdhury R., Zeng Q., and others., Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature biotechnology*, 29:644–652, 2011.

[2] Robertson G., Schein J., Chiu R., Corbett R., Field M, Jackman SD., Mungall K, Lee S., Okada HM., Qian JQ., and others., De novo assembly and analysis of RNA-seq data. *Nature methods*, 7:909–912, 2010.

[3] Schulz MH., Zerbino DR., Vingron M., Birney E., Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28:1086–11092, 2012

[4] Sacomoto GAT., Kielbassa J., Chikhi R., Uricaru R., Antoniou P., Sagot MF., Peterlongo P and Lacroix V., KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, 13(Suppl 6):S5, 2012.

[5] Marco-Sola S., Sammeth M., Guigo R. and Ribeca P., The GEM mapper: fast, accurate and versatile alignment by filtration. *Nature Methods*, 9:1185–1188, 2012.

[6] Thorvaldsdottir H., Robinson JT. and Mesirov JP., Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration, *Brief. Bioinformatics*, 2012

[7] Chikhi R. and Rizk G., Space-Efficient and Exact de Bruijn Graph Representation Based on a Bloom Filter. *Algorithms in Bioinformatics*, Lecture Notes in Computer Science, 7534:236-248, 2012.

[8] Salikhov K., Sacomoto G., Kucherov G., Using cascading Bloom filters to improve the memory usage for de Brujin graphs. *Arxiv preprint*, arXiv:1302.7278, 2013.

[9] Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K: dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research* 2001, 29:308–311.