Combinatorial structures and NGS data in transcriptomics

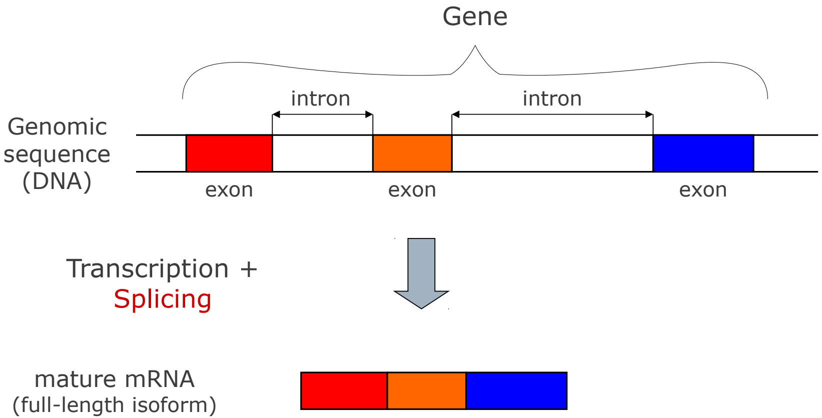
Stefano Beretta Paola Bonizzoni Gianluca Della Vedova
Yuri Pirola Raffaella Rizzi

DISCo, Univ. degli Studi di Milano-Bicocca, Milan, Italy
`yuri.pirola@disco.unimib.it`

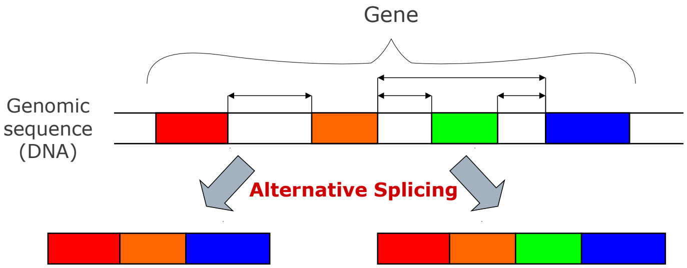
Workshop: *"Combinatorial structures for sequence analysis in bioinformatics"*
Univ. degli Studi di Milano-Bicocca, November 27, 2013

Outline

Eukaryotic Gene Structure



Alternative Splicing

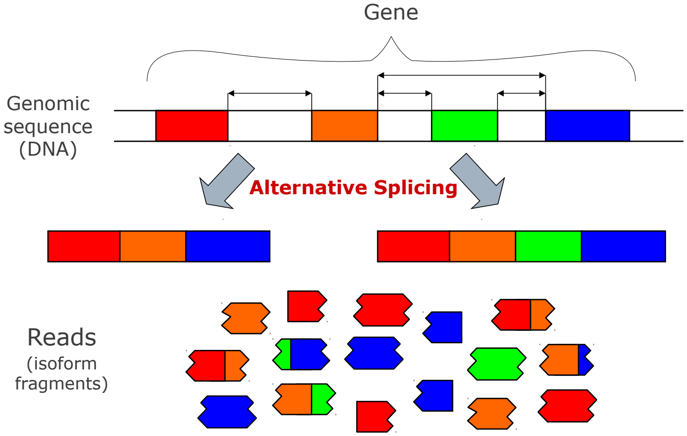


One gene, **several** distinct isoforms (**several** proteins)

- AS is widespread (95% of human genes)
- Aberrant AS events $\rightarrow$ diseases

(Pan et al., Nat Gen, 2008) and (Matlin et al., Nat Rev, 2005)

Alternative Splicing



Transcript reconstruction

Major goal

Reconstruct the set of all the original (unknown) full-length isoforms (**transcriptome**) starting from their fragments.

Most of the classical approaches:

- . . . are *reference-based*

  But the genomic sequence might be:

  - unavailable/unfinished
  - mutated wrt that of the individual under study

- . . . would like to *reconstruct the full-length transcripts*

  But fragments are very *short*, transcripts are very *long*
  $\rightarrow$ long-range predictions are unreliable

Assessment of transcript reconstruction methods for RNA-seq

Tamara Steijger[1], Josep F Abril[2,11], Pär G Engström[1,10,11], Felix Kokocinski[3,11], The RGASP Consortium[4], Tim J Hubbard[3], Roderic Guigó[5,6], Jennifer Harrow[3] & Paul Bertone[1,7–9]

We evaluated 25 protocol variants of 14 independent computational methods for exon identification, transcript reconstruction and expression-level quantification from RNA-seq data. Our results show that most algorithms are able to identify discrete transcript components with high success rates but that assembly of complete isoform structures poses a major challenge even when all constituent elements are identified. Expression-level estimates also varied widely across methods, even when based on similar transcript models. Consequently, the complexity of higher eukaryotic genomes imposes severe limitations on transcript recall and splice product discrimination that are likely to remain limiting factors for the analysis of current-generation RNA-seq data.

High-throughput sequencing instruments necessitate a shotgun approach for all but the shortest target molecules. Full-length representation of most cellular RNAs from sequencing data requires computational reconstruction of transcript structures. The majority of such programs infer transcript models from the accumulation of read alignments to the genome[1–4]; some take approaches are relatively adept, along with more challenging areas for future improvement.

RESULTS

We evaluated a total of 25 transcript reconstruction protocols, basing our analysis on alternate parameter usage of 14 software packages on RNA-seq data sets for three species (**Supplementary Fig. 1**, **Supplementary Table 1** and **Supplementary Note**). Programs were run by the original developers, with the exception of Cufflinks, iReckon and SLIDE. So that we could assess the ability of each method to interpret transcript expression from RNA-seq data without prior knowledge of gene content, programs were run without genome annotation, aside from iReckon and SLIDE, which require such information. Performance was benchmarked relative to the subset of annotated exons to which RNA-seq reads mapped (coverage of ≥1 read pair per 100 bp) and their corresponding transcripts (Online Methods).

Identification of annotated features

We first assessed the degree to which gene components

A different view of the problem

Sometimes transcript reconstruction is an overkill

A less ambitious aim:

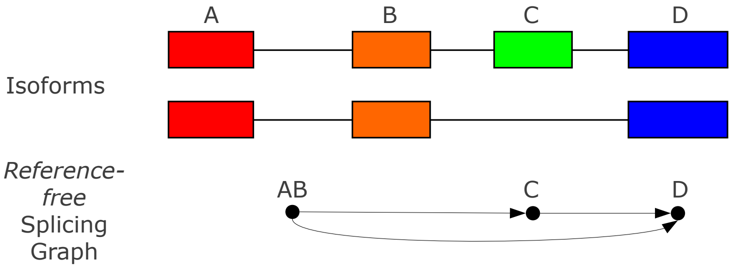
> Reconstruct a graph structure describing the AS events occurred in a sample (without the reference genome).

$\rightarrow$ **Splicing Graphs**!

(Reference-free) Splicing Graphs

(Reference-free) Splicing graphs:

Vertices:	*blocks*
Vertex labels:	*block*'s nucleotide sequence
Edges:	two *blocks* consecutive in some isoform



Similar concepts: (Heber *et al.*, Bioinf, 2002), (Lacroix *et al.*, WABI, 2008), . . .

The computational problem

Problem: **Splicing Graph Reconstruction (SGR)**

Input: the set R of all the l-long substrings of the isoforms.

Output: a “minimum-length” splicing graph G that is **compatible** with R (i.e., the set R_S of all the l-long substrings of the labels of paths of G is equal to R)

Strict formulation under *ideal* assumptions in order to highlight the intrinsic limits of SGR

Negative results

Question

Is the reconstructed splicing graph always equal to the real splicing graph?

Solvable genes: reconstructed SG = real SG

Not all genes are solvable

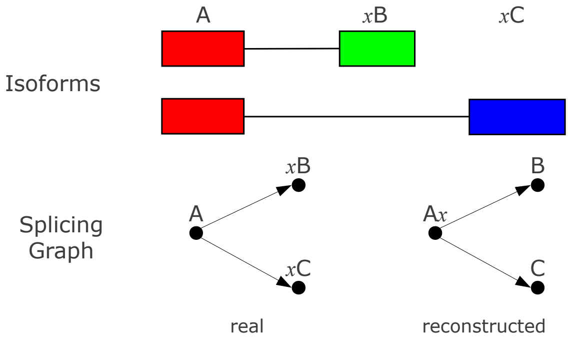
Unsolvable genes

A gene is **not solvable** if:

1. it is “ambiguous”
2. there exists an $(l-1)$-long repeat in two blocks
3. there exists an $(l-1)$-long repeat in a single block

Unsolvable genes - 1. Ambiguity

Example:

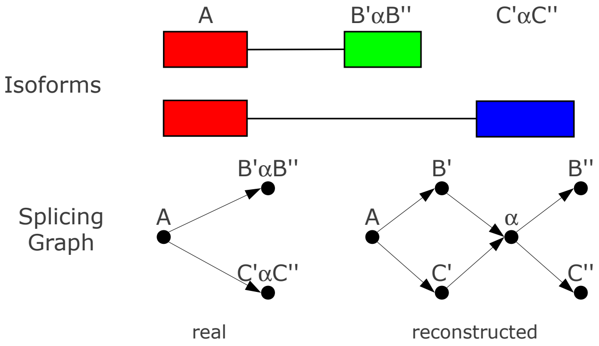


The reconstructed SG is “shorter” than the real SG...

But $R_S = R \rightarrow$ the reconstructed SG is *compatible* with R.

Unsolvable genes - 2. Repeats

Example:



(Repeats in a single block have a similar effect.)

Positive results

There exists an efficient algorithm that:

- optimally solves SGR for *"well-expressed"* genes (subclass of solvable genes)
    - not ambiguous
    - without $l/2$-long repeats
    - with blocks of length $\geq l$
- works well also for the other genes (empirical experimental evidence)

Method outline

Algorithm outline:

1. *Hash table read indexing* for read classification in unspliced/spliced/perfectly-spliced

   Keys of the hash tables are $l/2$-long suffixes/prefixes

2. *Block creation*
   Unspliced reads are assembled into blocks

3. *Edge creation*
   Perfectly-spliced reads form edges

Theoretical properties

Properties:

- Linear running time (in the size of R)
- Correct for well-expressed genes (subclass of solvable genes)

On non well-expressed genes...

Pre/post processing heuristics effectively lessen the impact of assumption violations:

- *Low coverage*
  Read enrichment for adding likely perfectly-spliced reads.
  (Work-in-progress: direct detection of “splice-junctions”)
- *Errors and SNPs* (mutations)
  Post-processing for linking to/merging with other nodes.
- *Repeats*
  If coverage is low, repeats do not heavily affect the prediction.
  (experimental evidence)

Experimental evaluation

- **Data:** synthetic datasets generated from real isoforms of benchmark genes (Guigó *et al.*, Genom Biol, 2006)
- **Comprehensive tests:**
  - ideal case
  - low coverage
  - mixed genes
  - SNPs
  - *comparison with Trinity* (Grabherr *et al.*, Nat Biotech, 2011)

Comparison with Trinity

Data: ideal case (full coverage, no errors, separated genes)

Competitor: *Trinity*, a de-novo transcript assembly

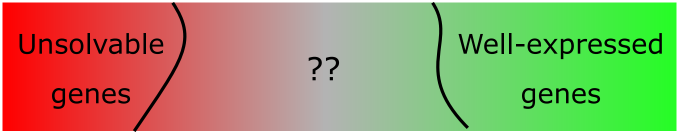
Splicing Graphs are then generated via alignment to the genome

Several transcripts cannot be aligned!!

Alternative hypothesis	Node p-values		Arc p-values	
	Sn	PPV	Sn	PPV
Our approach > Trinity	0.228	**0.020**	0.452	0.191
Trinity > Our approach	0.839	0.990	0.640	0.870

Conclusions

- Splicing graphs can overcome some issues of transcript reconstruction
- Formal (strict/ideal) definition of SGR
- Characterization of genes:

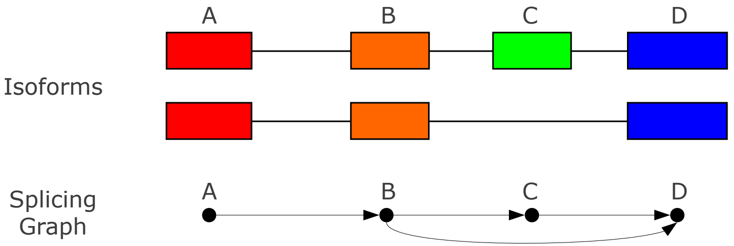


Additional Content

Splicing Graphs

Splicing graphs:

Vertices:	exons
Vertex labels:	exon's nucleotide sequence
Edges:	two exons consecutive in some isoform

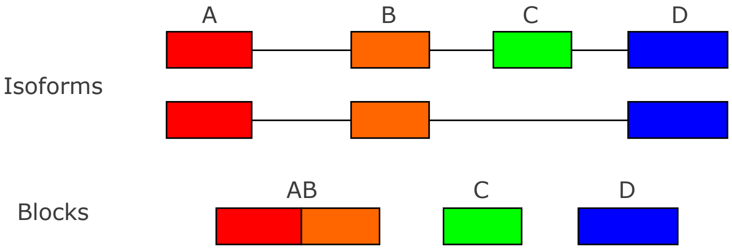


(Heber *et al.*, Bioinf, 2002), (Lacroix *et al.*, WABI, 2008), . . .

Exons and Blocks

Without the reference, we cannot detect **exons** but **blocks**.

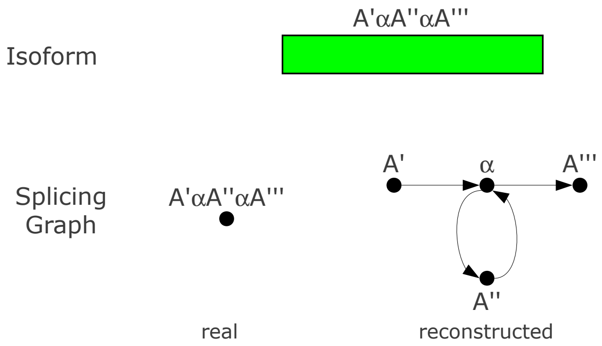
Block: composition of consecutive exons that appear together or do not appear in each isoform.



It is not a limit: AS events are *not* occurred inside blocks.

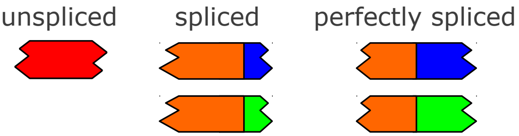
Unsolvable genes - 3. Repeats

Example:



Read classification

Reads are:



Classification?
Put each read in a hash table LH with key equal to the $l/2$-long *prefix* and in a hash table RH with key equal to the $l/2$-long *suffix*

Spliced reads: reads with the same key (in LH or in RH)

Unspliced reads	$\rightarrow$	blocks
Spliced reads	$\rightarrow$	block boundaries
Perf. spliced reads	$\rightarrow$	graph edges

Block and edge creation

Block creation:

1. Assemble chains of *unspliced* reads having an $l/2$ overlap
2. Merge overlapping chains
3. Trim block borders

Edge creation:

For each block, check in the hash table if there exists a perfectly spliced read linking it to another block.