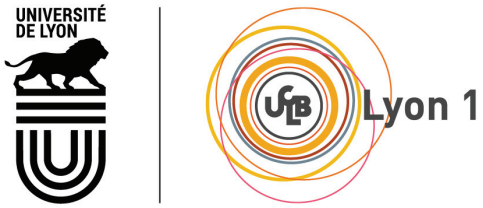
De novo algorithms to identify patterns associated with biological events in de Bruijn graphs built from NGS data

Leandro Ishi Soares de Lima



N° d'ordre NNT : 2019LYSE1055

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

École Doctorale ED01
E2M2

Spécialité de doctorat : Bioinformatique

Soutenue publiquement le 23/04/2019, par :
Leandro Ishi Soares de Lima

De novo algorithms to identify patterns associated with biological events in de Bruijn graphs built from NGS data

---

Devant le jury composé de :

Bonizzoni Paola, Full Professor, Università Degli Studi di Milano-Bicocca Rapporteure
Iqbal Zamin, Senior Researcher, EMBL-EBI Rapporteur

Brochier-Armanet Céline, Full Professor, Université Claude Bernard Lyon 1 Examinatrice
Foissac Sylvain, Junior Researcher, GenPhySE INRA Toulouse Examinateur
Naldi Maurizio, Associate Professor, Università di Roma "Tor Vergata" Examinateur
Varré Jean-Stéphane, Full Professor, Université de Lille Examinateur
Sagot Marie-France, Senior Researcher, LBBE, INRIA Rhône-Alpes Directrice de thèse
Italiano Giuseppe, Full Professor, Università di Roma "Tor Vergata" Co-directeur de thèse
Lacroix Vincent, Associate Professor, Université Claude Bernard Lyon 1 Co-encadrant de thèse, Invité

UNIVERSITE CLAUDE BERNARD-LYON 1

Président de l'Université	**M. le Professeur F. FLEURY**
Président du Conseil Académique	M. le Professeur H. BEN HADID
Vice-Président du Conseil d'Administration	M. le Professeur D. REVEL
Vice-président du Conseil Formation et Vie Universitaire	M. le Professeur P. CHEVALIER
Vice-président de la Commission Recherche	M. F. VALLÉE
Directeur Général des Services	M. A. HELLEU

COMPOSANTES SANTE

Faculté de Médecin Lyon-Est - Claude Bernard	Directeur: M. le Professeur J. ETIENNE
Faculté de Médecine et de Maeutique Lyon Sud Charles Mérieux	Directeur: Mme la Professeure C. BURILLON
Faculté d'Odontologie	Directeur: M. le Professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur: Mme la Professeure C. VINCIGUERRA
Institut Techniques de Réadaptation	Directeur: M. le Professeur MATILLON
Département de Formation et Centre de Recherche en Biologie Humaine	Directeur: Mme la Professeure A-M. SCHOTT

COMPOSANTES ET DEPARTEMENTS DE SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur: M. F. De MARCHI
Département Biologie	Directeur: M. le Professeur F. THEVENARD
Département Chimie Biochimie	Directeur: Mme C. FELIX
Département Génie Electrique et des Procédés	Directeur: M. Hassan HAMMOURI
Département Informatique	Directeur: M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur: M. le Professeur G. TOMANOV
Département Mécanique	Directeur: M. le Professeur H. BEN HADID
Département Physique	Directeur: M. le Professeur J-C PLENET
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur: M. Y.VANPOULLE
Observatoire des Sciences de l'Univers de Lyon	Directeur: M. B. GUIDERDONI
Ecole Polytechnique Universitaire de Lyon 1	Directeur: M. le Professeur E.PERRIN
Ecole Supérieure de Chimie Physique Electronique	Directeur: M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur: M. le Professeur C. VITON
Ecole Supérieure du Professorat et de l'Education	Directeur: M. le Professeur A. MOUGNIOTTE
Institut de Science Financière et d'Assurances	Directeur: M. N. LEBOISNE

Acknowledgements

First and foremost I thank my family for being patient with me on these four years of PhD. My parents, which had to get used to the distance, as it was not easy for any of the parties to travel between Brazil and France, and my wife, Heloísa. Heloísa was very patient to me in some periods, mainly close to deadlines, where I did some working nights and weekends. Moreover, although the decision to come to France and do a PhD was a joint one, Heloísa was really kind to move, join, and support me through the whole PhD. However, I can't help feeling this was somewhat selfish, as she would not be able to work and progress in her career in France due to her VISA type. Yet, she was always by my side, supporting me in the good, bad, and "meh" days of the PhD. She was the most present person in my whole PhD, although not technically involved in it. It is hard to put into few words how important and essential you were during all these years, so I hope a "I love you" is able to express all my feelings!

I am also very grateful to my supervisors, Marie-France Sagot, Giuseppe Italiano, and Vincent Lacroix. This thesis was possible because of your support and directions. I have learned a lot through you, not only about how to do research, but also how to collaborate with a diverse range of researchers, ranging from undergraduate students to senior researchers, coming from different backgrounds, which is essential in bioinformatics. I also thank you for all your patience during these four years. I acknowledge that it is not easy to mentor students. Sometimes I've been stubborn, or I would bother you for some minimal issues, or follow several paths that would not lead to results, in general I was not very optimistic, etc... This would lead to several meetings, sometimes making you repeat something that you had mentioned to me previously. So I thank you a lot for all the patience you had with me during these periods. But that is all about learning how to do research I guess, which usually does not correspond to a linear and predictable path from the problem to the results. I am very grateful for the approach you've taken to direct me, as it allowed me to be headed to the right direction, while keeping most of low-level details for me to solve, and some of the literature for me to explore, research, and find. I now feel confident to work on research projects due to you and your supervision.

My feeling of gratitude also extends to my non-supervisors collaborators which directed some works (alphabetical order): Laurent Jacob and Rayan Chikhi, which did not have the responsibility of mentoring me, but from which I have learned due to their attention and patience. And also to the other co-first authors in some papers (alphabetical order): Blerina Sinaimeri, and Magali Jaillard, which was a pleasure to work with, and from who I learned more about theoretical computer science, statistics, and GWAS. I also thank all the other collaborators in published works (alphabetical order): Alex van Belkum,

Amandine Rey, Audric Cologne, Benjamin Istace, Camille Marchet, Clara Benoit-Pilven, Corinne Da Silva, Cyril F. Bourgeois, Didier Auboeuf, Emilie Chautard, Gustavo Sacomoto, Helene Lopez-Maestre, Hélène Touzet, Jean-Baptiste Claude, Jean-Marc Aury, Louis Dulaurier, Marie-Pierre Lambert, Maud Tournoud, Pierre Mahé, Roberto Grossi, Romeo Rizzi, Ségolène Caboche, Sophie Terrone, Vicente Acuña, and Vincent Miele. And many others with some ongoing works or interesting discussions (alphabetical order): Alex Digenova, Antoine Limasset, Arnaud Mary, Camille Sessegolo, Gabriela Paludo, Mariana Ferrarini, Said Sadique Adi, and all the members of the ASTER consortium. Special thanks to Claire Sauer, Marina da Graça, and Florence Bouheddi for simplifying a lot the bureaucracy. Of course most of the colleagues I have worked with turned into dear friends, but I would like also to thank all the other friends I had made during the PhD (alphabetical order): Alice Julien Laferrière, Carol Quinteros, Catherine Michel, Cecilia Klein, Christian Baudet, David Parsons, Delphine Parrot, Eric Cumunel, Yishu (Hélio) Wang, Irene Ziska, Katiane Rocha, Laura Urbini, Laurent Bulteau, Marianne Borderès, Martin Wannagat, Mattia Gastaldello, Nicolas Homberg, Nina Paffoni, Pedro Lealdino, Ricardo Andrade, Scheila Mucha, Susan Higashi, Taneli Pusa, and Xavier Domingo.

My deepest thanks to everyone mentioned up to here (I hope I did not forget anyone!), in particular the ones working in LBBE/Lyon, which composed my second, but most present family in these four years. Thanks for all the laughs, coffees/teas, nights out, beers, movies, etc!

I thank Université Claude Bernard Lyon 1, Università degli Studi di Roma "Tor Vergata", and the Erable team for accepting me as a PhD student. I would like to thank specially the Espace Ulys service from Université de Lyon, which made a big difference in my stay in France. Many bureaucratic issues were solved easily due to Espace Ulys, and they made us feel very welcome in France, through their actions such as events to understand the French culture, culinary, habits, etc, as well as events for the PhD students' families. This service makes a huge difference in the life of foreigner PhD students, and Espace Ulys did an awesome job everytime we needed to contact them. I also thank the Pôle administratif LBBE and the Direction de la Recherche et des Etudes Doctorales (D.R.E.D.) for making the bureaucracy of a PhD less heavy.

Lastly, I would like to thank the "Conselho Nacional de Desenvolvimento Cientifico e Tecnologico – CNPq" and the "Ministério da Ciência, Tecnologia e Inovação - MCTI" from Brazil, which funded this PhD through the "Ciência sem Fronteiras - CsF" program. I also thank the french people in general for being warm and open to foreigners.

TITRE en français

Algorithmes de novo pour l'identification de motifs associés à des événements biologiques dans les graphes de De Bruijn construits à partir de données NGS

RESUME en français

L'objectif principal de cette thèse est le développement, l'amélioration et l'évaluation de méthodes de traitement de données massives de séquençage, principalement des lectures de séquençage d'ARN courtes et longues, pour éventuellement aider la communauté à répondre à certaines questions biologiques, en particulier dans les contextes de transcriptomique et d'épissage alternatif.

Notre objectif initial était de développer des méthodes pour traiter les données d'ARN-seq de deuxième génération à l'aide de graphes de De Bruijn afin de contribuer à la littérature sur l'épissage alternatif, qui a été exploré dans les trois premiers travaux. Le premier article (Chapitre 3, article [77]) a exploré le problème que les répétitions apportent aux assembleurs de transcriptome si elles ne sont pas correctement traitées. Nous avons montré que la sensibilité et la précision de notre assembleur local d'épissage alternatif augmentaient considérablement lorsque les répétitions étaient formellement modélisées. Le second (Chapitre 4, article [11]) montre que l'annotation d'événements d'épissage alternatifs avec une seule approche conduit à rater un grand nombre de candidats, dont beaucoup sont importants. Ainsi, afin d'explorer de manière exhaustive les événements d'épissage alternatifs dans un échantillon, nous préconisons l'utilisation combinée des approches *mapping-first* et *assembly-first*. Étant donné que nous avons une énorme quantité de bulles dans les graphes de De Bruijn construits à partir de données réelles d'ARN-seq, qui est impossible à analyser dans la pratique, dans le troisième travail (Chapitre 5, articles [1,2]), nous avons exploré théoriquement la manière de représenter efficacement et de manière compacte l'espace des bulles via un générateur des bulles. L'exploration et l'analyse des bulles dans le générateur sont réalisables dans la pratique et peuvent être complémentaires aux algorithmes de l'état de l'art qui analysent un sous-ensemble de l'espace des bulles.

Les collaborations et les avancées sur la technologie de séquençage nous ont incités à travailler dans d'autres sous-domaines de la bioinformatique, tels que: études d'association à l'échelle des génomes, correction d'erreur et assemblage hybride. Notre quatrième travail (Chapitre 6, article [48]) décrit une méthode efficace pour trouver et interpréter des *unitigs* fortement associées à un phénotype, en particulier la résistance aux antibiotiques, ce qui rend les études d'association à l'échelle des génomes plus accessibles aux panels bactériens, surtout ceux qui contiennent des bactéries plastiques. Dans notre cinquième travail (Chapitre 7, article [76]), nous évaluons dans quelle mesure les méthodes existantes de correction d'erreur ADN à lecture longue sont capables de corriger les lectures longues d'ARN-seq à taux d'erreur élevé. Nous concluons qu'aucun outil ne surpasse tous les autres pour tous les indicateurs et est le mieux adapté à toutes les situations, et que le choix devrait être guidé par l'analyse en aval.

Les lectures longues d'ARN-seq fournissent une nouvelle perspective sur la manière d'analyser les données transcriptomiques, puisqu'elles sont capables de décrire les séquences complètes des ARN messagers, ce qui n'était pas possible avec des lectures courtes dans

plusieurs cas, même en utilisant des assembleurs de transcriptome de l'état de l'art. En tant que tel, dans notre dernier travail (Chapitre 8, article [75]), nous explorons une méthode hybride d'assemblage d'épissages alternatifs qui utilise des lectures à la fois courtes et longues afin de répertorier les événements d'épissage alternatifs de manière complète, grâce aux lectures courtes, guidé par le contexte intégral fourni par les lectures longues.

MOTS-CLEFS en français
ARN-seq, Lectures courtes, Lectures longues, Épissage alternatif, Graphes de De Bruijn, Bulles, Études d'association à l'échelle des génomes, Correction d'erreurs, Assemblage hybride.

Title in english
De novo algorithms to identify patterns associated with biological events in de Bruijn graphs built from NGS data

---

Abstract in english
The main goal of this thesis is the development, improvement and evaluation of methods to process massively sequenced data, mainly short and long RNA-sequencing reads, to eventually help the community to answer some biological questions, especially in the transcriptomic and alternative splicing contexts.

Our initial objective was to develop methods to process second-generation RNA-seq data through de Bruijn graphs to contribute to the literature of alternative splicing, which was explored in the first three works. The first paper (Chapter 3, paper [77]) explored the issue that repeats bring to transcriptome assemblers if not addressed properly. We showed that the sensitivity and the precision of our local alternative splicing assembler increased significantly when repeats were formally modeled. The second (Chapter 4, paper [11]), shows that annotating alternative splicing events with a single approach leads to missing out a large number of candidates, many of which are significant. Thus, to comprehensively explore the alternative splicing events in a sample, we advocate for the combined use of both mapping-first and assembly-first approaches. Given that we have a huge amount of bubbles in de Bruijn graphs built from real RNA-seq data, which are unfeasible to be analysed in practice, in the third work (Chapter 5, papers [1, 2]), we explored theoretically how to efficiently and compactly represent the bubble space through a bubble generator. Exploring and analysing the bubbles in the generator is feasible in practice and can be complementary to state-of-the-art algorithms that analyse a subset of the bubble space.

Collaborations and advances on the sequencing technology encouraged us to work in other subareas of bioinformatics, such as: genome-wide association studies, error correction, and hybrid assembly. Our fourth work (Chapter 6, paper [48]) describes an efficient method to find and interpret unitigs highly associated to a phenotype, especially antibiotic resistance, making genome-wide association studies more amenable to bacterial panels, especially plastic ones. In our fifth work (Chapter 7, paper [76]), we evaluate the extent to which existing long-read DNA error correction methods are capable of correcting high-error-rate RNA-seq long reads. We conclude that no tool outperforms all the others across all metrics and is the most suited in all situations, and that the choice

should be guided by the downstream analysis.
RNA-seq long reads provide a new perspective on how to analyse transcriptomic data, since they are able to describe the full-length sequences of mRNAs, which was not possible with short reads in several cases, even by using state-of-the-art transcriptome assemblers. As such, in our last work (Chapter 8, paper [75]) we explore a hybrid alternative splicing assembly method, which makes use of both short and long reads, in order to list alternative splicing events in a comprehensive manner, thanks to short reads, guided by the full-length context provided by the long reads.

Keywords in english
RNA-seq, Short reads, Long reads, Alternative splicing, de Bruijn graphs, Bubbles, Genome-wide association studies, Error-correction, Hybrid assembly.

Resumé en français

L'objectif principal de cette thèse est le développement, l'amélioration et l'évaluation de méthodes de traitement de données massives de séquençage, principalement des lectures de séquençage d'ARN courtes et longues, pour éventuellement aider la communauté à répondre à certaines questions biologiques, en particulier dans les contextes de transcriptomique et d'épissage alternatif. Bien que l'objectif soit le développement de méthodes basées sur les graphes de De Bruijn (Chapitres 3, 6 et 8, articles [48, 75, 77]), la conception ou l'amélioration de telles méthodes est également liée à des études théoriques (Chapitre 5, articles [1, 2]) et à des analyses méthodologiques détaillées (Chapitres 4 et 7, articles [11, 76]). Les études théoriques permettent d'améliorer les méthodes en fournissant de nouveaux résultats, sous forme de théorèmes, d'algorithmes ou de structures de données, qui peuvent être directement appliqués ou adaptés à des problèmes pratiques. Les analyses de méthodes permettent d'évaluer de manière critique les méthodes actuelles, identifier leurs points forts et faibles, fournir à la communauté des informations détaillées sur le fonctionnement de chaque outil et sur les problèmes à résoudre.

Notre objectif initial était de développer des méthodes pour traiter les données d'ARN-seq de deuxième génération à l'aide de graphes de De Bruijn afin de contribuer à la littérature sur l'épissage alternatif. En tant que tel, le premier article (Chapitre 3, article [77]) montre que, bien que les répétitions soient moins nombreuses et plus courtes dans les données d'ARN-seq que dans les données d'ADN-seq, elles peuvent toujours créer des problèmes pour les assembleurs de transcriptome si elles ne sont pas traitées correctement. Les assembleurs de transcriptome peuvent donc être améliorés en modélisant explicitement et formellement les répétitions. Nous introduisons un modèle formel pour représenter les répétitions dans les données d'ARN-seq et exploitons ses propriétés pour déduire une caractéristique combinatoire de sous-graphes associés aux répétitions. Nous montrons que la sensibilité et la précision de notre assembleur local d'épissage alternatif augmentent considérablement lorsque les répétitions ont été formellement modélisées, avec un algorithme qui les évite implicitement. De plus, nous montrons également que l'exploration de la topologie du sous-graphe autour d'un transcript peut donner des indications sur son niveau de confiance, sa qualité, la difficulté de l'assemblage, etc. Ces informations peuvent être aussi utiles que des informations de lecture et de couverture pour les assembleurs et les évaluateurs de transcriptome. Le deuxième travail (Chapitre 4, article [11]) montre que l'annotation de l'épissage alternatif avec une seule approche conduit à rater un grand nombre de candidats, dont beaucoup sont exprimés différentiellement selon deux conditions et qui ont pu être validés expérimentalement. Ces événements ne doivent pas être exclus de l'analyse car ils peuvent jouer un rôle central dans la ques-

tion biologique étudiée. Nous avons donc plaidé en faveur de l'utilisation combinée des approches *mapping-first* et *assembly-first* pour l'annotation et l'analyse différentielle de l'épissage alternatif à partir de jeux de données RNA-seq.

Après ces deux premiers travaux, nous avons eu plusieurs idées sur la façon d'améliorer notre méthode actuelle. Parmi ces différentes idées, certaines n'ont pas fonctionné comme prévu et ont été abandonnées, certaines sont encore en cours de développement en collaboration avec d'autres membres de l'équipe (celles-ci sont exposées dans les perspectives de la thèse), et certaines ont été publiées. Dans ce manuscrit, nous ne décrirons que les travaux que nous avons pu publier ou que nous sommes sur le point de soumettre. Un de ces travaux (Chapitre 5, articles [1,2]) décrit un résultat théorique sur la manière de représenter efficacement et de manière compacte l'espace des bulles via un générateur des bulles. Ce générateur peut être trouvé en temps polynomial et nous montrons également que nous pouvons décomposer n'importe quelle bulle du graphe en bulles du générateur en un nombre polynomial d'étapes. L'exploration et l'analyse des bulles dans le générateur sont réalisables dans la pratique et peuvent être complémentaires aux algorithmes de l'état de l'art qui analysent un sous-ensemble de l'espace des bulles. Pour le moment, ce travail reste en grande partie théorique, mais nous avons quelques preuves de concepts suggérant qu'il peut avoir une bonne application biologique, bien que des travaux supplémentaires soient nécessaires sur cette partie.

Par ailleurs, les collaborations et les avancées sur la technologie de séquençage nous ont incités à travailler dans d'autres sous-domaines de la bioinformatique, tels que: études d'association à l'échelle des génomes, correction d'erreur et assemblage hybride. Nous avons collaboré avec Magali Dancette et Laurent Jacob, et combiné notre expérience sur les graphes de De Bruijn et le développement méthodologique avec leurs connaissances sur l'association génotype à phénotype sur des populations bactériennes, afin de concevoir une méthode efficace pour trouver et interpréter des *unitigs* fortement associés à un phénotype, en particulier la résistance aux antibiotiques (Chapitre 6, article [48]). Notre méthode rend les études d'association à l'échelle des génomes plus facile à utiliser pour les panels bactériens, surtout ceux qui contiennent des bactéries plastiques. Ces génomes peuvent être trop différents pour être alignés sur une référence, même au sein d'une seule espèce, rendant difficile la description de leur variation génétique. Au lieu de travailler avec des k-mers comme les approches précédentes, nous travaillons avec des *unitigs*, des descripteurs aussi polyvalents que les k-mers, et qui permettent de capturer des variants génétiques allant de polymorphismes locaux aux insertions de longs éléments génétiques mobiles, mais pas redondants et plus faciles à interpréter. Nous proposons un *framework* graphique afin de réduire l'écart d'interprétabilité entre les approches basées sur les k-mers, et les approches basées sur les SNPs et les gènes.

En outre, nous avons suivi les communautés génomiques et transcriptomiques, et nous avons déplacé notre attention vers les technologies de séquençage de troisième génération (lectures longues), en réfléchissant aux moyens d'évaluer et d'intégrer ce type de données dans nos modèles. L'un des principaux problèmes du séquençage à lecture longue est qu'il est actuellement freiné par les taux d'erreur élevés qui affectent les analyses telles que l'identification des isoformes, les frontières des exons, les cadres de lecture ouverts et la création de catalogues de gènes. En collaboration avec les membres du projet ASTER, un projet visant à développer des algorithmes et des logiciels d'analyse des données de

séquençage de troisième génération, nous avons commencé par analyser à quel point les méthodes actuelles, généralement adaptées à la génomique, peuvent corriger les données de Nanopore RNA-seq (Chapitre 7, article [76]). Nous évaluons donc neuf outils de correction d'erreur d'ADN hybrides et non-hybrides de l'état de l'art dans le contexte de la correction de lectures de séquençage d'ARN longues. Nous rapportons non seulement les métriques classiques de correction d'erreur, mais également l'effet de la correction sur les familles de gènes, la diversité des isoformes, le biais vers l'isoforme principal et la détection du site d'épissage. Nous trouvons que les outils de correction d'erreur à lecture longue développés à l'origine pour l'ADN conviennent également à la correction des données de séquençage d'ARN, notamment en termes d'augmentation de la précision des paires de bases. Cependant, les chercheurs doivent être avertis que le processus de correction perturbe la taille des familles de gènes et la diversité des isoformes. Ce travail fournit des indications sur les outils de correction d'erreur à utiliser (ou non), en fonction du type d'application.

Nous pouvons dire que les lectures longues permettent d'étudier les transcripts intégralement, car elles peuvent les séquencer du début à la fin, alors que les lectures courtes conviennent mieux aux approches d'assemblage local. En effet, les lectures longues pourraient également être utilisées pour étudier des événements locaux, tels que l'épissage alternatif. Cependant, cette approche présente deux problèmes principaux: son coût élevé, qui fait qu'uniquement une fraction du transcriptome, principalement les isoformes hautement exprimés, sont couverts par de lectures longues, et son taux d'erreur élevé. Bien que le séquençage à lecture longue soit actuellement peu profond et pas aussi complet que le séquençage à lecture courte pour décrire les événements d'épissage alternatif, ils sont capables de décrire la structure complète des ARNs messagers, ce qui est difficile ou impossible, dans certains cas, avec des lectures courtes. Le séquençage complet d'un transcript donné fournit un guide pour assembler les événements d'épissage autour du transcript. Dans ce dernier travail (Chapitre 8, article [75]), nous explorons une méthode hybride d'assemblage d'épissage alternatif qui utilise des lectures à la fois courtes et longues afin de répertorier les événements d'épissages alternatifs de manière complète, grâce aux lectures courtes, guidé par le contexte intégral fourni par les lectures longues. Nous attirons l'attention sur le fait que ce travail est toujours en préparation.

Globalement, le fil principal que nous avons suivi au cours de cette thèse a été le développement et l'amélioration de méthodes de traitement de données séquencées à l'aide de graphes de De Bruijn afin de contribuer à la littérature sur l'épissage alternatif (Chapitres 3 et 8, articles [75, 77]). Nous avons également abordé ce problème général à partir d'une perspective théorique (Chapitre 5, articles [1, 2]) et analytique (chapitre 4, article [11]). Les collaborations (Chapitres 6 et 7, articles [48, 76]) nous ont dévié de ce fil principal, mais nous avons toujours eu au moins un aspect principal en commun. Je crois qu'avoir des déviations par rapport au fil principal dans un doctorat est vraiment sain, car il permet aux étudiants de doctorat, qui sont introduits à la science, d'acquérir une connaissance d'autres domaines, ce qui pourrait aider à ouvrir des portes dans la période post-doctorale.

En ce qui concerne la période post-doctorale, mon point de vue général sur le contexte scientifique dans lequel cette thèse est placée est que, sauf si un problème spécifique nécessite des lectures courtes, la communauté scientifique concentrera ses efforts sur le

traitement des lectures longues afin de résoudre les questions biologiques. Je crois que les lectures longues finiront par prendre la relève, et l'assemblage de transcriptomes pourrait même ne plus être nécessaire, et l'assemblage de génomes sera simplifié avec des lectures plus longues et plus précises. Cependant, il faudra quelques années pour que les technologies à lecture longue atteignent cet état, donc à court terme (5-10 ans, ou même plus), je pense que les méthodes capables d'utiliser efficacement les lectures courtes et longues définissent l'état de l'art. En ce qui concerne l'épissage alternatif et les variations transcriptomiques en général, il est très utile de pouvoir décrire complètement la structure d'un transcript par une lecture longue, puisque les exons peuvent être parfaitement phasés. L'identification précise des isoformes, des limites d'exons et des cadres de lecture ouverts peut encore poser problème avec le taux d'erreur élevé, mais des protocoles comme PacBio Iso-seq traitent déjà cela de manière native, et créent des *Reads of Insert* très précis. Certainement, le taux d'erreur et le coût de Nanopore et de PacBio vont diminuer, et le débit va augmenter dans les prochaines années, permettant ainsi de telles applications. Le protocole de séquençage d'ARN direct de Nanopore permettra également d'étudier les variations transcriptomiques sans les biais et les artefacts créés par l'étape de synthèse de l'ADN complémentaire, permettant une meilleure compréhension de la variation transcriptomique réelle dans une cellule.
Un autre scénario possible est que le séquençage Illumina reste très compétitif même après plusieurs années, principalement en réduisant le coût par base et en augmentant le débit, justifiant ainsi son utilisation même lorsqu'une question biologique peut être résolue en ne séquençant que des lectures longues. Cela impliquera des ensembles de données encore plus grands que ceux que nous avons aujourd'hui, et la demande de méthodes hybrides efficaces qui utilise des structures de données succinctes augmentera, avec une large fraction étant probablement basée sur des graphes de De Bruijn ou de ses variantes.
Une opinion plus générale sur la bioinformatique à l'issue de cette thèse est que ce domaine nécessite des compétences diverses, difficiles à maîtriser, allant des mathématiques et de l'informatique théorique, jusqu'à la biologie et les techniques expérimentales. La modélisation mathématique, les analyses critiques, l'implémentation efficace de logiciels et la manipulation correcte des techniques expérimentales ne sont pas des tâches faciles, et je crois que des collaborations efficaces sont essentielles pour produire de bons travaux et apprendre de nouveaux concepts.

Contents

Chapter 1

Introduction

The objective of this chapter is to give the reader a complete, but succinct overview of the works performed during this thesis. We start by describing, in Section 1.1, the aim of this thesis. We then proceed by presenting, in Section 1.2, the most important points of the papers produced in this thesis. We finish the description of each paper with its main messages. The objective of this section is to help the readers to decide which chapters might interest them. As such, we want to be concise, giving only an overview of each paper, but we also try to be more detailed than the paper's abstract. For the sake of brevity, and since the focus of this chapter is to describe the produced papers, we are skipping here the basic definitions in biology, computer science and bioinformatics. The formal, comprehensive, and detailed definitions can be found in Chapter 2. Sometimes, the text in the papers also assumes that the readers know beforehand some definitions, as papers are occasionally not self-contained, since their focus are a specific extension of the literature. We therefore advise the readers to check out Chapter 2 also in search of definitions not properly defined in the papers. We further take the liberty of considering as publications works that are not yet peer-reviewed, *i.e.*, still submitted or under review (those correspond to Chapter 7, paper [76]). We also note that one of the works here presented, Chapter 8, paper [75], is still in preparation, although sometimes we use the term paper or publication to refer to it. We finish by describing the outline of this thesis in Section 1.3.

1.1 Aim and development of this thesis

The main goal of this thesis is the development, improvement and evaluation of methods to process massively sequenced data, mainly short and long RNA-sequencing reads, to eventually help the community to answer some biological questions, especially in the transcriptomic and alternative splicing contexts. Although its focus is the development of methods based on de Bruijn graphs (Chapters 3, 6 and 8, papers [48, 75, 77]), the conception or improvement of such methods is also related to theoretical studies (Chapter 5, papers [1, 2]) and comprehensive method analyses (Chapters 4 and 7, papers [11, 76]). Theoretical studies allow the improvement of methods by providing new results, in form of theorems, algorithms, or data structures, that can either be directly applied or adapted

to fit the model of a practical problem. Method analyses allow to critically evaluate current methods, identifying their strong and weak points, providing the community details on how each tool performs and which problems should be further addressed.

Our initial objective in this thesis was to develop methods to process second-generation RNA-seq data through de Bruijn graphs to contribute to the literature of alternative splicing. As such, the two first papers I got involved in this PhD contributed to a better understanding, from my part, of the state-of-the-art on alternative splicing events identification, from an algorithmical (Chapter 3, paper [77]) and an analytical (Chapter 4, paper [11]) point-of-view. Although this first contact was tough for me, mainly because I had a computer science background, and just an introduction to bioinformatics and biology, I was fortunate enough to work with people that would patiently teach me (sometimes basic) concepts in biology and bioinformatics. Working on [77] allowed me to understand better the history and the current state of KisSplice, a method developed in our team to find alternative splicing events from RNA-seq data without a reference genome by enumerating bubbles in a de Bruijn graph, which has been conceived some years before the start of this PhD [14, 107–109]. In complement to this, I was also working on [11] as a second author, which provided me a more practical view of the field: how KisSplice and other tools, implementing different approaches, can be combined to annotate and analyse alternative splicing events from RNA-seq datasets.

After these two first works, we had several ideas on how to improve our current method. Among these different ideas, some did not work as expected and were abandoned, some are still being developed in collaboration with other members of the team (those are exposed in the perspectives of the PhD), and some were published. In this manuscript, we shall describe only the works that we were able to publish, or that we are close to submitting. One such work (Chapter 5, papers [1,2]) describes how to better explore the set of all bubbles in a de Bruijn graph in search of alternative splicing events that are hard to find (*i.e.* unfeasible in practice) by the current KisSplice algorithm. For now, this work remains largely theoretical, but we have some proofs of concepts hinting that it can have a nice biological application, although further work on this part has to be done. Personally, it was enriching for me to work on a purely theoretical paper after the two first papers. I can say that these three first works were a nice introduction to the diverse competences a mixed field such as bioinformatics requires, as they covered methodological development, methods analyses, and theoretical studies.

Further, collaborations and advances on the sequencing technology encouraged us to work in other subareas of bioinformatics, such as: genome-wide association studies, error correction, and hybrid assembly. We collaborated with Magali Dancette and Laurent Jacob, and combined our experience on de Bruijn graphs and methodological development with their knowledge on genotype-to-phenotype association on bacterial populations, to conceive an efficient method to find and interpret unitigs highly associated to a phenotype, especially antibiotic resistance (Chapter 6, paper [48]). I am grateful for this collaboration, as it introduced me to genome-wide association studies and antibiotic resistance, which can be a possible direction to follow in future works.

Next, we followed the genomic and transcriptomic communities, and shifted our attention to third generation (long reads) sequencing technologies, thinking on ways to evaluate and integrate this type of data in our models. In collaboration with members

of the ASTER project, a project with the purpose of developing algorithms and software for analysing third-generation sequencing data, we started by analysing how well current methods, usually tailored for genomics, can correct Nanopore RNA-seq data (Chapter 7, paper [76]). I believe this is an important question to answer, as this type of data is becoming widely used, but the high error rate affects downstream analyses, and options for the error-correction of RNA-seq long reads remain very limited.

Moreover, we also started to develop a method to explore PacBio Iso-seq and Illumina data in order to integrate long and short reads in the search for alternative splicing events in a reference-free context (Chapter 8, paper [75]). When alternative splicing events are of interest, sequencing only long reads might not be enough, as these technologies are currently unable to dig as deep in the transcriptome as short reads. As such, this is a first step on working on a hybrid alternative splicing assembler, but this paper is still in preparation.

Overall, the main thread we followed during this thesis was developing and improving methods to process sequenced data using de Bruijn graphs to contribute to the alternative splicing literature (Chapters 3 and 8, papers [75, 77]). We tackled this general problem also from a theoretical (Chapter 5, papers [1,2]) and an analytical (Chapter 4, paper [11]) perspective. Collaborations deviated us from this main thread, but we always had at least one main aspect in common (Chapter 6, paper [48] describes a method based on de Bruijn graphs, and Chapter 7, paper [76], describes an evaluation of methods to correct RNA-seq long reads). I do believe that having some deviations from the main thread in a PhD is really healthy, as it allows PhD students, who are being introduced to science, to have a knowledge of other fields and areas, which might help to open doors in the post-PhD period.

1.2 Publications

In the following, we introduce the papers and manuscripts produced in this PhD.

1.2.1 Playing hide and seek with repeats in local and global de novo transcriptome assembly of short RNA-seq reads

The first paper [77] explored the fact that repeats are an underestimated problem in *de novo* transcriptome assembly, creating ambiguities and confusing assemblers if not addressed properly. This happened in the method developed in our team, KisSplice [107], conceived to enumerate alternative splicing (AS) events in a *de-novo* context. The KisSplice algorithm was improved in [14], and later in [108]. However, even the improved algorithm is not able to enumerate all bubbles corresponding to AS events in a de Bruijn graph. There are certain complex regions in the graph, likely containing repeat-associated subgraphs but also real AS events, where it takes a huge amount of time. In practice, the enumeration is halted after a given timeout. The bubbles trapped inside these regions are thus missed.

To address this issue, we first introduce a simple, but realistic enough, model for representing high copy-number and low-divergence repeats in RNA sequencing data. We then exploit its properties to infer that repeat-associated subgraphs contain few

compressible arcs (or many branching vertices). Based on this, we formulate the repeat identification problem in RNA-seq data, whose objective is to find for large enough subgraphs that do not contain many compressible arcs, which would correspond to repeat-associated subgraphs. We show that this problem is NP-complete for directed graphs with total degree (maximum number of in- and out-arcs in a vertex) bounded by $d \geq 3$, including, in particular, de Bruijn graphs, so an efficient algorithm for it is unlikely. However, in the specific case of a local assembly of AS events, we can implicitly avoid repeat-associated subgraphs based on our previous characterisation. More precisely, it is possible to find bubbles corresponding to AS events in a de Bruijn graph that are not contained in a repeat-associated subgraph, by restricting the search to bubbles with few branching vertices. We provide a polynomial-delay algorithm to enumerate such bubbles, and we show, through simulated datasets, that this new algorithm is significantly more sensitive and precise than the previous version of KisSplice [108], Trinity [38], and Oases [114], for the specific task of calling AS events.

Finally, we turn our focus to full-length transcriptome assembly. We argue that: i) most transcriptome assemblers are based on de Bruijn graphs and have no clear and explicit model for repeats in RNA-seq data, relying instead on heuristics to deal with them; ii) the most commonly used protocol to extract RNA yields pre-mRNA fractions around 5%. Thus, more introns than expected are sequenced, generating problems to transcriptome assemblers, particularly when several introns span members of a specific repeat family.

Within the complex parts of the graph generated by repeats, any assembler will have to choose the "right" path(s) among the many present. If the assembler decides to guess a path, it may erroneously extend a contig and create a chimeric transcript. It can also choose to be conservative by not choosing any path in complicated regions of the de Bruijn graph, and instead truncating the transcript. Although this strategy can lead to an accurate assembly, it will produce a very fragmented one, which is not desirable. Whatever the strategy (conservative or permissive), the resulting assembled transcript may be erroneous (chimeric or truncated).

It is hence important to be able to identify low-confidence transcripts, which are the ones traversing complex regions of a de Bruijn graph, in order to know that the solution presented is the result of a "difficult" choice and therefore *may* not be the right one. To identify such transcripts, we introduce the concept of Branching Measure of a transcript t, which is able to indicate if t traversed a hard-to-assemble region (*i.e.* a region with many branching vertices) in the de Bruijn graph. We then show a proof of concept of this measure by providing two examples where it was able to flag a chimeric and a truncated transcripts assembled by Trinity on real RNA-sequencing data. Finally, we show that this simple Branching Measure gives better results than Rsem-Eval [67] and TransRate [117] on both real and simulated datasets for detecting chimeras, and therefore is able to capture assembly errors missed by these methods.

Main message

The main message of this work is that, although fewer and shorter repeats are present in RNA-seq data than in DNA-seq data, they can still create problems for transcriptome

assemblers if not addressed properly. Transcriptome assemblers can thus be improved by explicitly and formally modeling repeats. Moreover, we also show that exploring the topology of the subgraph around a transcript can give some hints about its confidence level, quality, assembly hardness, etc. This information can be as valuable as read and coverage information for transcriptome assemblers and evaluators.

1.2.2 Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data

While the first paper explored a limitation of current *de novo* RNA-seq assembly methods, the second [11] focused on showing that such methods can also be applied even when a high-quality reference genome and annotations are present.

In general, there are two approaches to assemble transcripts or alternative splicing (AS) from RNA-seq data [84]. The mapping-first approaches first map the reads to the reference genome and the mapped reads are then assembled into exons and eventually transcripts. In contrast, assembly-first approaches first assemble the reads based on their overlaps. The assembled sequences (corresponding to sets of exons) are then aligned to the reference genome.

Mapping-first approaches have been the most used so far, essentially because they were the first to be developed and they initially required less computational resources. *De novo* assembly methods were also thought to be restricted to non-model species, where no (good) reference genome is available, and they seemed to be inadequate when an annotated reference genome is available.

Recent progress in *de novo* transcriptome assembly is clearly changing this view, and the argument of the heavier computational burden does not hold anymore. The application of *de novo* assembly to human RNA-seq datasets however still remains rare, although some studies have already shown its potential to detect novel biologically relevant splicing variants [25, 33]. The generalization of *de novo* assembly approaches for studying splicing in human seems to be mostly impeded by the lack of a clear evaluation of its potential interest in comparison to more traditional mapping-based approaches. In this paper, we fill this gap by performing a systematic evaluation of an assembly-first and a mapping-first approach on two RNA-seq datasets.

As a first step, we compared pipelines that we developed in parallel, namely KisSplice and FaRLine, because we could easily control their parameters. Any difference between the predictions that is solely due to a parameter setting could be fixed easily, which enabled us to obtain a precise understanding of the irreducible differences between the two approaches. Overall, we developed and adapted jointly these two pipelines in order to minimize the discrepancies that could complicate the comparison.

We found out that the mapping-first approach predicts a much larger number of events. This difference in sensitivity is due to the fact that while mapping-first approaches require that each exon junction is covered by at least one read, assembly-first approaches require overlapping reads across the entire skipped exon. Therefore, it can be anticipated that low abundant isoforms, that are covered by few reads, will be reported by mapping, but not by the assembly-first approach.

Having clarified that rare variants are better handled by the mapping-first approach, we decided to filter them out, in order to analyse other differences between the two approaches. After this filtering, approximately 70% of the predicted skipped exons were found by both approaches. We highlight also that some isoforms are systematically missed by one approach. Mapping-first approaches miss AS events involving: i) novel exons or novel combinations of existing exons; ii) recent paralog genes. Assembly-first approaches miss AS events involving: i) repeats; ii) complex AS events.

In a second step, we confirmed the generality of our findings by benchmarking our methods against Cufflinks [126], MISO [53] and Trinity [38], which are widely used pipelines. Overall, we found that the vast majority of AS events were predicted by FaR-Line, MISO and Cufflinks, showing that the differences between mapping- and assembly-first approaches reported above are not limited to one mapping-first approach. Finally, we also verified that KisSplice is significantly more sensitive than the most widely used *de novo* full-length transcriptome assembler, namely Trinity.

Main message

The main message of this work is that annotating alternative splicing with a single approach leads to missing out a large number of candidates, many of which are differentially regulated across conditions and that we were able to validate experimentally. Such AS events should not be discarded from the analysis, as they may play a central role in the studied biological question. We therefore advocate for the combined use of both mapping-first and assembly-first approaches for the annotation and differential analysis of alternative splicing from RNA-seq datasets.

1.2.3 On Bubble Generators in Directed Graphs

As mentioned earlier, KisSplice models alternative splicing (AS) events as bubbles in a de Bruijn Graph (DBG) built from the input reads. Theoretically, the number of bubbles in a graph can grow exponentially on the size of the graph. In practice, DBGs built from real datasets tend to be huge, usually containing millions of vertices and a prohibitively large amount of bubbles. Any algorithm that tries to be exhaustive, listing and analysing a big part of the bubble space, will certainly spend a prohibitive amount of time in real data graphs and will not be applicable. As such, algorithms that deal with bubbles in such huge graphs will either simplify the graph by removing them, or just analysing a small subset of the bubble space. KisSplice algorithms, for example, only lists bubbles with predefined constraints, which was shown to be usually associated to AS events [77, 107–109]. Such subsets may, however, not be the best representatives of the bubble space. More worrying is the fact that all the relevant events described by bubbles not satisfying the predefined constraints are lost.

In this third work [1], we thus explored some mathematical properties of the bubble space in order to find an efficient and compact description of all bubbles in a graph G, called bubble generator $\mathcal{G}(G)$. Our intuition is that the bubble generator is a suitable compressed representation of the bubble space, and exploring it might allow us to retrieve information that is lost when only a subset of the bubble space is analysed. We first define a constrained symmetric difference operator Δ, such that, given two bubbles, B_1 and B_2,

$B_1 \Delta B_2$ is defined if and only the subgraph induced by the arcs of $(A(B_1) \cup A(B_2)) \setminus (A(B_1) \cap A(B_2))$ is a bubble; otherwise, we say that $B_1 \Delta B_2$ is undefined. If $B = B_1 \Delta B_2$, then we say that we can combine B_1 and B_2 into B, or that B can be decomposed into B_1 and B_2. We show that any bubble in a graph G can be described as the combination of two or more bubbles from the generator $\mathcal{G}(G)$. Moreover, for any given directed graph G, we also introduce an algorithm to find, in polynomial time, such generator set of bubbles $\mathcal{G}(G)$, with $|\mathcal{G}(G)| \leq nm$, where n and m are the number of vertices and arcs of G, respectively. Indeed, $|\mathcal{G}(G)|$ is polynomial on the size of the graph, but we also highlight that our generator is not minimal, *i.e.* there may be bubbles $b \in \mathcal{G}(G)$ which can be obtained by combining bubbles in $\mathcal{G}(G) - b$ through Δ. Finding a minimal generator is left as open problem.

Finally, we prove that, given a graph G, any bubble B in G can be represented as a sum of $O(n^2)$ bubbles belonging to $\mathcal{G}(G)$. This decomposition can be found in a total of $O(n^3)$ time. This decomposition algorithm can be applied when one needs to know how to decompose a bubble into its elementary parts, which are the bubbles in $\mathcal{G}(G)$, *e.g.* when identifying and decomposing complex AS events [112] into several elementary AS events. The link between the bubble generator and the set of such elementary AS events, however, has still to be further studied.

This theoretical work was presented in the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2017), and published in LNCS [1]. We further extended this paper, by applying it in two different directions in the analysis of a real RNA-seq dataset. First, we employed the generator as a preprocessing step to algorithms that find bubbles, by "cleaning" from the graph all unnecessary arcs (*i.e.* arcs that do not belong to any bubble). In a sample dataset, we were able to reduce the size of the graph by around 40%. Second, we use it to find AS events in a reference-free context. In particular, in one experiment, 18.5% of the putative events from our bubble generator are hard to find using the KisSplice algorithm. Moreover, 10% of these bubbles correspond to true AS events that are missed by KisSplice, showing the applicability of the generator. However, this application should still be seen just as a proof-of-concept on the practical potential of the bubble generator or as complementary to current methods, since it remains limited for the exhaustive enumeration of AS events. The latter would require a non-trivial procedure to enumerate AS-associated bubbles by combining generator bubbles, which we still did not explore. The extension [2] has been submitted to the journal *Algorithmica*, and is presented in Chapter 5.

Main message

The main theoretical result of this work is an improvement on the literature on how to efficiently and compactly represent the bubble space through a bubble generator. This generator can be found in polynomial time, and we also show that we can decompose any bubble in the graph into bubbles of the generator in a polynomial number of steps. Exploring and analysing the bubbles in the generator is feasible in practice and can be complementary to state-of-the-art algorithms that analyse a subset of the bubble space.

1.2.4 A fast and agnostic method for bacterial genome-wide association studies: bridging the gap between k-mers and genetic events

In this fourth work [48], we changed our context from transcriptomics to genomics, but we still kept the same theoretical framework, using de Bruijn Graphs (DBGs) to model the data contained in a set of sequences. Here, we applied DBGs to Genome-Wide Association Studies (GWAS). GWAS aim at identifying associations between genetic variants and a phenotype observed in a population, and rely on a representation of the genomic variation as numerical factors. The most common approaches are based on single nucleotide polymorphisms (SNPs), defined by aligning all genomes of the studied panel against a reference genome or against a pangenome built from all the genes identified by annotating the genomes, and on gene presence/absence, using a pre-defined collection of genes. The use of a reference genome becomes unsuitable when working on bacterial species with a large accessory genome – the part of the genome which is not present in all strains. On the other hand, methods focusing on genes are unable to cover variants in noncoding regions, including those related to transcriptional and translational regulation. Moreover, some poorly studied species still lack a representative annotation. To circumvent these issues, recent studies have relied on k-mers. The presence of k-mers in genomes can account for diverse genetic events such as the acquisition of SNPs, (long) insertions/deletions and recombinations. While k-mers can reflect any genomic variation in a panel, they do not themselves represent biological entities. A k-mer representation often loses in interpretability what it gains in flexibility, and the best way to encode the genomic variation in bacterial GWAS is not yet clearly defined. Moreover, translating the result of a k-mer-based GWAS into meaningful genetic variants is not trivial.

The approach developed in the fourth work [48], coined DBGWAS, bridges the gap between, on the one hand, SNP- and gene-based representations lacking the right level of flexibility to cover complete genomic variations, and, on the other hand, k-mer-based representations which are flexible but not readily interpretable. In this work, we relied on compacted DBGs (cDBGs) to eliminate local redundancy, reflect genomic variations, and characterise the genomic environment of a k-mer at the population level. More precisely, we build a single cDBG from all the genomes included in the association study. From this cDBG, we build a variant matrix which represents patterns of presence/absence of unitigs (vertices of the cDBG) in each genome of the population. Each variant is then tested for association with the phenotype using a linear mixed model, adjusting for the population structure. The unitigs found to be significantly associated to the phenotype are then localised in the cDBG, and their neighbourhood are used as a proxy for their genomic environment at the population level. Subgraphs induced by such neighbourhoods are extracted, and they often provide a direct interpretation in terms of genetic events through the integration of three types of information: 1) the *topology* of the subgraph, reflecting the nature of the genetic variant, 2) the *metadata* represented by vertex size and colour, allowing us to identify which vertices in the subgraph are associated to a particular phenotype status, and 3) an optional sequence *annotation* helping to detect unitigs mapping to – or near – a known gene.

We show how such subgraphs output by DBGWAS can be read as genetic events using several antibiotic resistance phenotypes within three bacterial species of various degrees of genome plasticity: *Mycobacterium tuberculosis*, *Staphylococcus aureus* and

Pseudomonas aeruginosa. By interpreting such subgraphs, we were able to identify, in the aforementioned panels, biological events such as mobile genetic element insertions, local polymorphisms in core and accessory genes, and also in non-coding regions, which corresponded to known and unknown resistance markers.

We compared DBGWAS results to those obtained by applying the same association model to a collection of known resistance genes and SNPs [27,49], and to two other recent k-mer-based methods: pyseer [64,65], and HAWK [100]. DBGWAS recovers known variants, while suggesting novel candidates out of the range of the resistome-based approach. It is also more computationally efficient and provides more interpretable outputs than the other k-mer-based methods. We also verified that DBGWAS is scalable, being able to process large panels of 5000 clonal bacterial strains, such as *Mycobacterium tuberculosis*, in half a day, and 2500 plastic bacterial strains, such as *Pseudomonas aeruginosa*, in one day using 8 cores.

Main message

The main message of this paper is that DBGWAS makes GWAS more amenable to bacterial panels, especially plastic ones. These genomes can be too different to be aligned against a reference, even within a single species, making the description of their genetic variation challenging. Instead of working with k-mers as previous approaches, we work with unitigs, which are descriptors as versatile as k-mers, allowing to capture genetic variants ranging from local polymorphisms to insertions of large mobile genetic elements, but not redundant and easier to interpret. We provide a computationally efficient and user-friendly implementation, enabling non-bioinformaticians to carry out GWAS on thousands of isolates in a few hours. Moreover, we offer a graphical framework which helps interpret GWAS results, narrowing the interpretability gap between k-mer-based, and SNP- and gene-based approaches.

1.2.5 Comparative assessment of long-read error-correction software applied to RNA-sequencing data

Advances on the sequencing technology resulted on the maturity of third generation sequencers, *e.g.* PacBio and Oxford Nanopore. On the RNA context, these technologies are being increasingly used as they better describe exon/intron combinations, and frequently sequence full-length transcripts, thus usually eliminating the assembling step and its related problems.

However, these technologies are currently hindered by high error rates that affect analyses such as the identification of isoforms, exon boundaries, open reading frames, and the creation of gene catalogues. Due to the novelty of such data, computational methods are still actively being developed. Many methods have been conceived to correct errors in RNA-seq short reads. They no longer apply to long reads because they were developed to deal with low error rates, and mainly substitutions. A new set of methods have been proposed to correct genomic long reads. There is, however, a lack of error-correctors that are specifically designed for RNA-sequencing long reads.

In this fifth work [76], we report on the extent to which state-of-the-art tools enable the correction of long noisy cDNA Nanopore reads. We take the standpoint of evaluating

DNA long-read error-correctors on RNA-seq data, an application that was likely not considered by the authors of the respective tools. There exist two types of long-read error-correction algorithms, those using information from long reads only (*self* or *non-hybrid* correction), and those using short reads to correct long reads (*hybrid* correction). We extensively benchmark four DNA hybrid correction tools: LoRDEC [110], NaS [82], PBcR [58], proovread [43]; and five DNA self-correction tools: Canu [59], daccord [123], LoRMA [111], MECAT [131], pbdagcon [22].

We first examine basic and classical metrics of error-correction, such as number of output and mapped reads, mean reads length, number of output and mapped bases, error rate, number of detected genes, running time and memory usage. We then concentrate on the error-rate analysis, breaking it down to evaluating the different types of errors: substitutions, insertions and deletions. Since homopolymer errors are systematic in Nanopore sequencing in particular, we further investigate these separately. We then ask several questions that are specific to transcriptome applications, which are the main contribution of this work to the literature. In a first step, we investigate if error-correction perturbs the number of reads mapping to the genes and transcripts, which can affect downstream RNA-sequencing analyses relying on these numbers for quantification, differential expression, etc. We then verify if error-correction truncates the transcriptome, by perturbing gene family sizes and isoform diversity. We find that error-correctors do not strictly preserve the sizes of gene families, and that multi-isoform genes tend to lose lowly-expressed isoforms after correction, and minor isoforms are corrected toward major isoforms. The latter is more prevalent when the variation is not long, *e.g.* when alternative exons are small. We also conclude that hybrid error correction tools present a clear advantage over the non-hybrid ones for allowing correct splice sites detection. We also provide a software that enables automatic benchmarking of long read RNA-sequencing error-correction tools, in the hope that future error-correction methods will take advantage of it to avoid biases.

Main message

Current long reads RNA-sequencing technologies are hindered by high error rates that affect several transcriptomic analyses. There is a lack of error-correctors that are specifically designed for correcting such reads. We thus evaluate nine state-of-the-art hybrid and non-hybrid DNA error-correction tools on the context of correcting long RNA-sequencing reads. Overall, hybrid tools outperform non-hybrid ones, mainly because they also have access to accurate and massive Illumina datasets. However, no tool outperforms all the others across all metrics and is the most suited in all situations. The choice should be guided by the downstream analysis, and we provide recommendations to some applications, such as quantification, isoform identification, and splice site detection.

1.2.6 Assemblying local alternative splicing events from short reads guided by accurate long reads

As explored in Subsection 1.2.1, assembling full-length transcripts from short reads without a reference genome is challenging. A recent solution to this problem is through the

full-length transcripts sequencing provided by long reads. However, in many applications, the focus can be restricted to the exon level. Identifying which exons can be alternatively spliced is already very valuable. It has been shown that local assembly of AS events is more sensitive and precise than global assembly strategies from short read data [11, 77, 107]. Therefore, we can say that long reads enable the study of full-length transcripts, while short reads are more appropriate for local assembly approaches. Indeed, long reads could also be used to study local events, like AS. However, we have two main issues with this approach: its high cost, which results in only a fraction of the transcriptome, mostly the highly expressed isoforms, being covered by long reads, and its high error rate. Although long-read sequencing is currently shallow and not as comprehensive as short-read sequencing to describe AS events, they are able to describe the complete structure of mRNAs, which is hard or impossible, in some cases, with short reads. The full-length sequencing of a given transcript provides a backbone or a guide to assemble AS events around the transcript. In this work, we therefore explore a hybrid AS assembly method, which makes use of both short and long reads, in order to list AS events in a comprehensive manner, thanks to short reads, guided by the full-length context provided by the long reads. We call attention to the fact that this work is still in preparation, and we shall improve some points of it for the publication.

Our described method is composed of four main steps. The first, hybrid DBG construction, builds a hybrid bicoloured DBG from both the short and long reads. We first build a DBG G_S from the short reads while removing potential sequencing errors. We then build a DBG G_L from the long reads. However, we do not perform any sequencing error removal procedures on G_L due to the shallowness of the long reads. Our method is thus primarily designed for perfect or low error-rate long reads, which can be obtained using PacBio SMRT Iso-Seq sequencing [104], or Nanopore INC-Seq sequencing [68], or through error-correction algorithms. Finally, we build a hybrid compressed DBG C in which we merge both graphs G_S and G_L. In the second step of exact mapping of long reads to the hybrid DBG, we map each long read l to C by retrieving a walk $w(l) \in C$ spelling out l. The third step, Unitig Linking Graph (ULG) construction, builds the ULG, an abstraction of the cDBG where the complex parts of the graph, usually associated to repeats, are removed and the remaining parts are connected using the read information. This structure is a follow-up of our paper [77]. The ULG allows to solve repeats larger than k, but shorter than the reads' length. The ULG shares similarities with several approaches that were conceived to add the read information back to the DBG in a reference-free context, *e.g.* multi-k DBG [8, 69, 90, 97, 98], and to approaches that encode the read information directly into the graph [45, 106, 127]. The main difference between the ULG and these approaches is that the ULG removes the complex, highly branching parts of the graph, and connects only the well-assembled unitigs through the read information, while the others work on the whole graph. The ULG U is built from a cDBG C. The vertices in U are the trustful unitigs, *i.e.* unitigs that are considered well assembled, have a low branching concentration, and are thus not induced by repeats. We remove non-trustful unitigs, and connect the trustful ones through the read information. We do so by mapping the short reads back to C, and creating arcs between two trustful unitigs if there is a read traversing them. The label of the arcs is precisely the sequence spelled by the read to traverse from one trustful unitig to the other. Finally, in the

last step, alternative splicing events enumeration, we enumerate the AS events that are present in the short reads and absent in the long reads, making use of the ULG and the read information it contains. We do so by iterating through the mapping of each long read l, and finding alternative paths in the ULG flanked by two unitigs stemming from l. The delay of our AS enumeration algorithm is $O(n * (m + n \log n))$, where n and m are the number of vertices and arcs of the ULG, respectively.

Moreover, we show some preliminary results of the application of the described method to some simulated datasets. Our first test case is a dataset composed of a single human gene, NEU1, which contains five transcripts. In this simple case, our method found all the AS events described by these five transcripts. Our next benchmark comprised a simulated dataset on the whole human chromosome 1. We currently obtain a recall of 99.6% and a precision of 88.7% in this larger test. We plan to improve our method with this benchmark by clarifying the 11.3% false positive events we currently have.

We finish this in-preparation paper by describing its perspectives, which we plan to develop in order to publish it, which include improving some points of the method, and benchmarking it on samples sequenced with both PacBio Iso-seq and Illumina.

1.3 Outline

This thesis is outlined as follows. Chapter 2 presents the biological, computer science, and bioinformatics concepts used in all the other chapters. Chapters 3 to 8 present each one paper, orderly from the first to the sixth. All these chapters follow the same simple structure. We first present a preamble, listing its key points, status (published, accepted, submitted or in preparation), and my contribution, along with the paper itself in its current state. Finally, Chapter 9 presents our concluding remarks and perspectives.

We realize that a text in a mixed field like bioinformatics will hardly interest all readers. Thus, in order to further help the reader to pick which chapters might interest her/him, we provide in Figure 1.1 an overview of the main content of this thesis, characterized by their keywords.

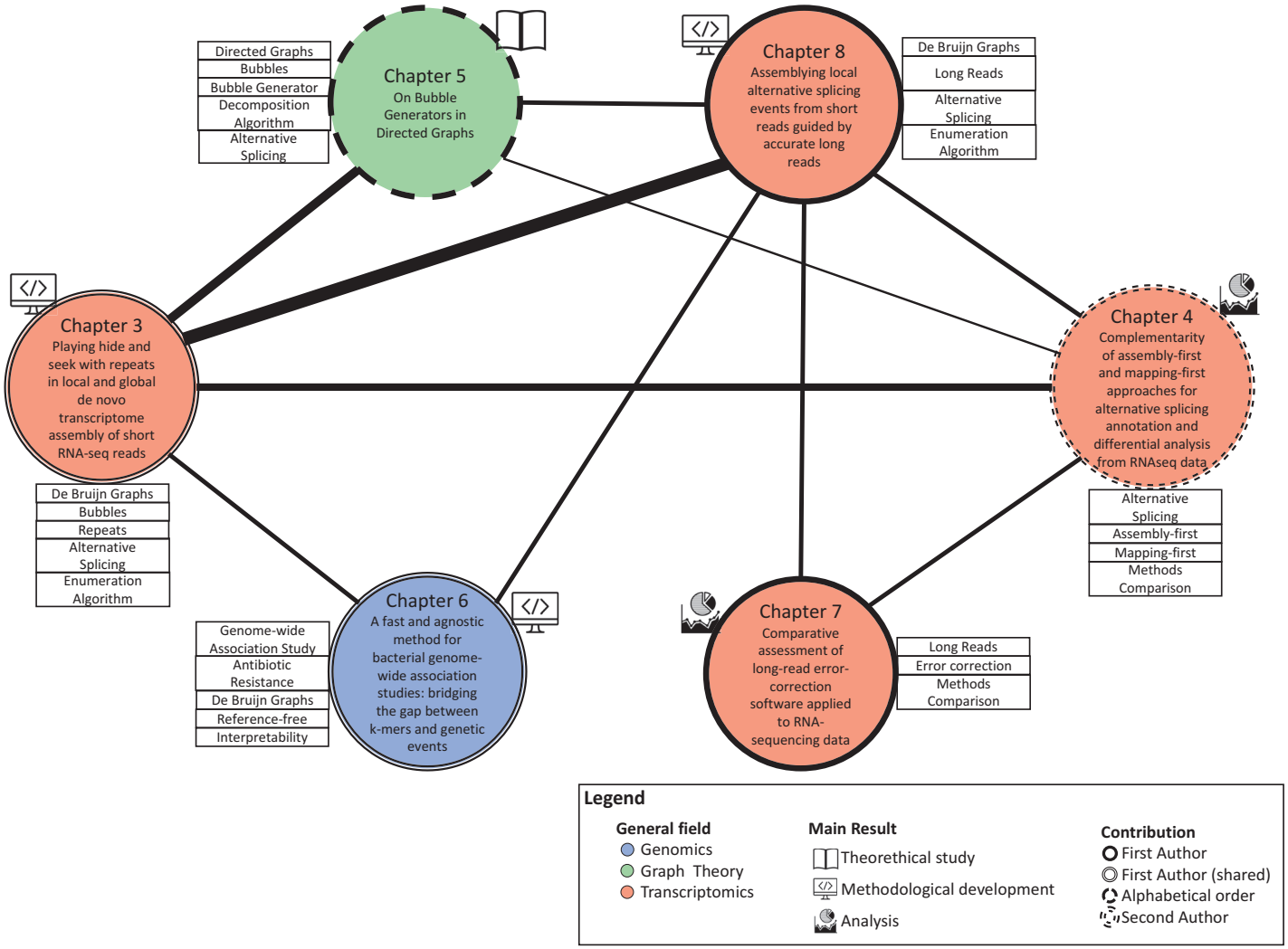


Figure 1.1: **Main content of this thesis, characterized by their keywords.** The vertices of the graph represent each paper. The vertice colour represent the general field the work is inserted on; the vertice icon represent its main result; the vertice border represent the author contribution; the information attached to the vertices list their keywords. The width of the edges denote the relationship between the papers – the larger, the more related.
Icons sources:
"software development by Chunk Icons from the Noun Project"; (`https://thenounproject.com/term/software-development/581057/`);
"Book by Curve from the Noun Project" (`https://thenounproject.com/search/?q=theory&i=623325`);
"analysis by Nibras@design from the Noun Project" (`https://thenounproject.com/search/?q=analysis&i=1961823`).

Chapter 2

Background

In this chapter, we introduce the concepts, definitions, and notations necessary to understand the main content of this thesis. Although we try to be complete and detailed in some concepts, we are unable to exhaustively cover here all the material with enough detail to make this thesis self-contained. Making this would not only be an extremely laborious task, but it would also put too much weight and attention on this chapter, which is not a result *per se*, *i.e.* this chapter contains no improvement to the literature.

The first two sections introduce biological and computer science concepts, respectively, while the third is devoted to bioinformatics. The two first fields are far more established than bioinformatics, which is a more recent area. Thus, for these two, the main concepts presented here will follow the two books I have used as reference for years, [3] and [23]. The main structure of these first two sections are therefore excerpts reproduced, combined or adapted from these two main books. We abstracted the text in both books so that details or concepts that are not relevant to this thesis are either removed or are described in a very succinct way. Although these books present a good part of the concepts, we also rely on other sources, when needed. For bioinformatics, on the other hand, we do not follow a single reference text.

2.1 Biology

We start with some biological concepts. Many of them are excerpts reproduced, combined or adapted from [3]. As such, when a concept is described and no reference is explicitly given, the reader can assume [3] as reference.

2.1.1 DNA, RNA and proteins

All living cells store their hereditary information in the form of double-stranded molecules of **DNA**, *i.e.* long unbranched paired polymer chains, formed by nucleotides. Each **nucleotide (or base pair (bp))** consists of two parts: a sugar (deoxyribose) with a phosphate group attached to it, and a base, which may be either adenine (A), guanine (G), cytosine (C) or thymine (T) (Figure 2.1A). Each sugar is linked to the next via

the phosphate group, creating a polymer chain composed of a repetitive sugarphosphate backbone with a series of bases attached to it. A DNA strand is a polymer chain of nucleotides (Figure 2.1B). DNA is synthesized on a template formed by a preexisting DNA strand. Through templated polymerization, the sequence of nucleotides in a preexisting DNA strand controls the sequence in which nucleotides are joined together in a new DNA strand. The new strand has a nucleotide sequence complementary to that of the old strand. The bases protruding from the existing strand bind to bases of the strand being synthesized, according to a strict rule defined by the complementary structures of the bases: A binds to T, and C binds to G (Figure 2.1C). This base-pairing holds fresh nucleotides in place and thereby controls the selection of which one of the four nucleotides shall be added to the growing strand next. In this way, a double-stranded structure is created, consisting of two exactly complementary sequences of As, Cs, Ts, and Gs (Figure 2.1D). The two strands twist around each other, forming a double helix (Figure 2.1E).

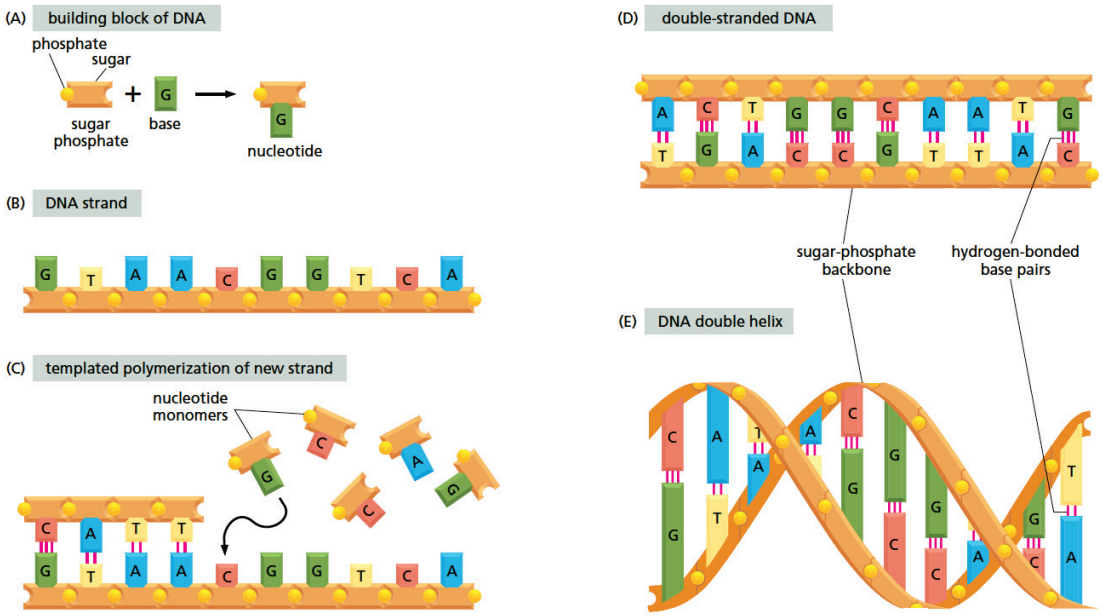


Figure 2.1: DNA and its building blocks. (A) A nucleotide and its composition; (B) A single strand of DNA consists of nucleotides joined together by sugarphosphate linkages; (C) Templated polymerization of new strand; (D) A normal DNA molecule consists of two such complementary strands; (E) The two strands twist around each other to form a double helix – a robust structure that can accommodate any sequence of nucleotides without altering its basic structure. Figure reproduced from [3].

Cells perform protein synthesis to transform the information stored in DNA molecules into protein molecules. **Proteins** are long unbranched polymer chains formed by chaining together amino acids, the monomers of proteins. Proteins are essential to organisms and have many functions. They can bind with high specificity to other molecules and

act as enzymes to catalyze reactions that make or break covalent bonds. In this way, they direct the vast majority of chemical processes in the cell. They can also maintain structures, generate movements, sense signals, and so on.

DNA molecules are generally very large, containing the specifications for thousands of proteins. However, only 1.5% of the human DNA corresponds to protein-coding genes. A **gene** is usually defined as a segment of DNA that contains the instructions for making a particular protein. A (very) simplified structure of an eucaryotic gene can be seen in Figure 2.2. This structure was purposely abstracted to present only the components relevant to this thesis. By no means it is a complete description of the structure of a gene. We will refer to this structure in the next paragraph.

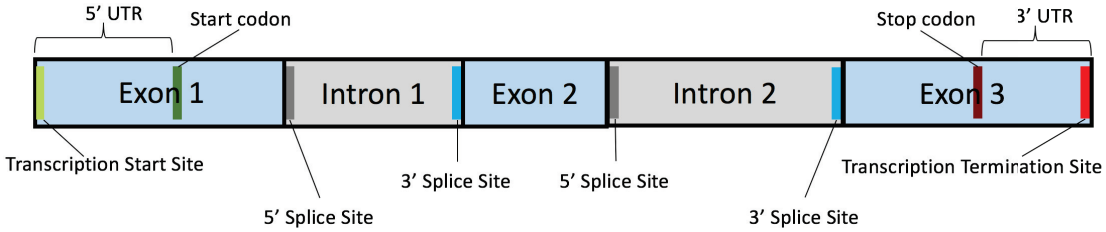


Figure 2.2: A (very) simplified structure of an eucaryotic gene with three exons and two introns.

In eucaryotes, the protein synthesis process begins with a templated polymerization called **transcription**, in which genes are used as templates for the synthesis of shorter molecules of the closely related polymer ribonucleic acid, or RNA. **RNA** is a linear polymer made of four different types of nucleotide subunits linked together by phosphodiester bonds. It differs from DNA chemically in two respects: (1) the nucleotides in RNA are ribonucleotides – that is, they contain the sugar ribose (hence the name ribonucleic acid) rather than deoxyribose; (2) like DNA, RNA contains the bases adenine (A), guanine (G), and cytosine (C), it contains the base uracil (U) instead of the thymine (T) in DNA. Since U, like T, can base-pair by hydrogen-bonding with A, the complementary base-pairing properties described for DNA apply also to RNA (in RNA, G pairs with C, and A pairs with U). The **transcription start site** is a regulatory region of the gene where the enzyme RNA polymerase starts the transcription of the gene to a RNA molecule. It finishes at the **transcription terminator site**. The RNA molecule resulting from the transcription process is termed **primary RNA transcript** or **pre-mRNA**, which is a reverse-complemented copy of the gene. The pre-mRNA is composed by alternating coding and non-coding sequences, called **exons** and **introns**, respectively. The boundaries between exons and introns are termed **splice sites**. The splice sites are in fact part of the intron, and the two first nucleotides of an intron are called **5' splice or donor site**, denoting the end of an exon and the start of an intron. The last nucleotides of an intron are called **3' splice or acceptor site**, denoting the end of an intron and the start of an exon. Before it can be translated into protein, the two ends of the pre-mRNA are modified by capping the 5' end and by polyadenylation of the 3' end, and the introns are

removed by **RNA splicing**. After all these processes take place, the resulting molecule is called **messenger RNA** or **mRNA**. We shall also widely use the terms **isoform** or **transcript** to refer to mRNAs in this thesis. Only when and if the RNA processing is completed successfully, the mRNA is transported from the nucleus to the cytoplasm through the nuclear pore complexes, where it can be translated into protein. However, just a part of the mRNA is translated. The untranslated regions (5' UTR and 3' UTR) are not translated – these are regulatory regions. The translation process builds an amino acid chain, *i.e.* a protein, based on the codons (triplets of bases) of the mRNA. The **start codon** (AUG) indicates where the cell machinery should start the translation process, and the **stop codon** (UAA, UAG or UGA) denotes where it should finish. We observe that the start codon is translated into an amino acid, but not the stop codon. We should also note that the 61 non-stop codons are translated into 20 amino acids, so some codons, known as **synonymous codons**, translate into the same amino acid. Much more can be said about the translation process, but since we do not focus on this mechanism in this thesis, we will not describe it further. Figure 2.3 summarizes the steps leading from DNA to protein in eucaryotes.

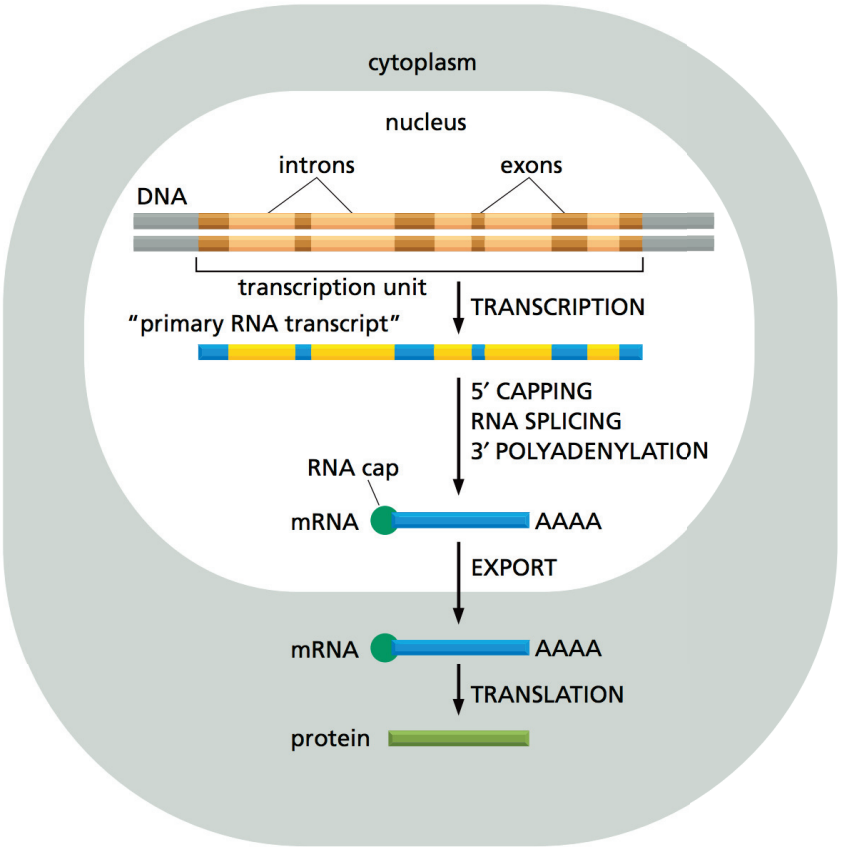


Figure 2.3: Summary of the steps leading from DNA to protein in eucaryotes. Figure adapted from [3].

2.1.2 Organization of the genetic material in a cell

In eucaryotes, the DNA in the nucleus is composed by a set of different chromosomes. A **chromosome** is a single, long linear DNA molecule containing not only genes, but also a considerable amount of interspersed DNA between the genes, called **intergenic regions**. A big part of intergenic regions is composed of repeated sequences, or repeats. **Repeats** are patterns of nucleic acids that occur in multiple copies throughout the genome, and can be classified into two broad classes: **tandem repeats**, when the copies of a segment of DNA are adjacent to one another, and **interspersed repeats**, when the copies are dispersed throughout the genome. Repeat copies evolve independently. Recent copies will be very similar to each other, while older copies will differentiate more, due to the accumulation of many mutations, mainly if the copies are not functional. This allows for the classification of repeats into families, and subfamilies, according to the similarity between their copies. There are many other details about repeats: some have been associated with regulatory and structural roles, and their replication and insertion mechanisms are complex and interesting. However, these details are out of the scope of this thesis. Finally, a **genome** is the totality of the genetic information belonging to a cell or an organism, *i.e.* the set of chromosomes. Figure 2.4 shows an overview of the organization of the genome of a cell. The classical definition of a **transcriptome** is the set of mRNAs expressed by an organism. We should note, however, that this definition does not describe the highly dynamic nature of the transcriptome of an organism. In most species, the genome is essentially the same across all cells at any given time, and it is expected to just slightly change during the life of an organism. In contrast, the transcriptome, *i.e.* the set of expressed transcripts, from cells belonging to different tissues can be very distinct. Even cells from the same tissue can have remarkable differences if they are in distinct conditions, caused by *e.g.* different developmental stages, or diseases, or external factors, etc. Figure 2.5 shows some of the human transcriptome complexity across several tissues.

2.1.3 Genomic and transcriptomic variations

We have mentioned that the RNA and, to a far less extent, the DNA molecules of an organism are not constant, immutable objects. They are dynamic entities that change in response to different conditions, creating a large number of variations. In this subsection, we will describe only a part of them, relevant to this thesis.

Some cells contain only one set of chromosomes, and are called **haploid**. However, many others contain more than one set: **diploid** cells contain two sets, **triploid** cells contain three sets, and so on. All sets of chromosomes represent the same genetic information, *i.e.* the sets are only duplications of a single set, but they also present some expected genomic variations. Humans cells, for example, are diploid. The specialized cells that carry out sexual reproduction, however, are haploid. In the final step of sexual reproduction, a haploid cell of one parent fuses with a haploid cell of the other, mixing the two genomes and restoring the diploid state. The genomes of both parents are similar, but not identical, thus a gene can have more than one version. Such alternative

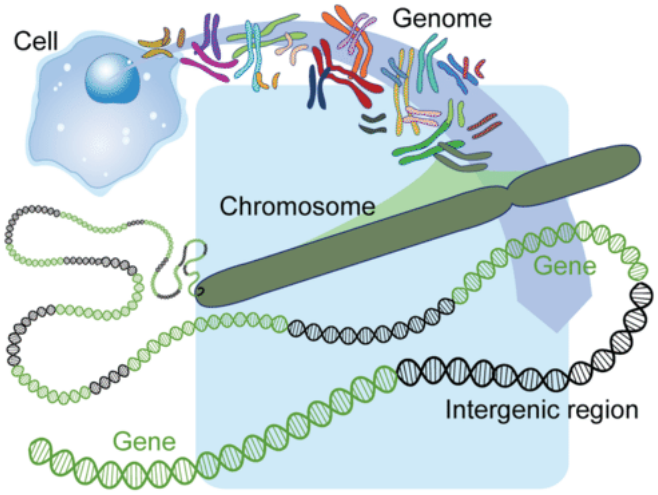


Figure 2.4: Overview of the organization of the genome, chromosome, genes and intergenic regions of a cell. Figure reproduced from [74].

versions are called **alleles**. The most common allelic variations are Single Nucleotide Polymorphisms (SNPs) and indels. **Single Nucleotide Polymorphisms (SNPs)**, as the name suggests, is when a specific nucleotide is different between the alleles. **Indels**, a short for insertion/deletions, is when the difference between the alleles are a (generally small) number of inserted or deleted nucleotides. Longer variations are also possible. **Recombinations** occur when either two chromosomes exchange a chunk of DNA, or one chromosome copies a chunk from another [81]. Genomic variations can also be acquired due to accidents during cell duplication, *i.e.* when cells are duplicating, a SNP can be created by accidentally mutating one base into another different nucleotide. If this mutation provides the organism a competitive advantage, it is probable that this variation will be transferred to its offsprings. This mechanism, called **vertical gene transfer**, is central to the evolution of eucaryotes. Bacteria, on the other hand, are able to acquire (long) genetic material, *e.g.* genes, from other species of bacteria, from its host or environment, by means other than reproduction or cell duplication, through **horizontal gene transfer (HGT)**. While the mechanisms of HGT are complex and interesting, describing them in details is out of the scope of this thesis. Further, other occasional flaws during cell duplication, can also result in the inappropriate duplication of just part of the genome, with retention of original and duplicate segments in a single cell, generating other types of variations. Once a gene has duplicated in this way, one of the two gene copies is free to mutate and may specialize to perform a different function within the same cell. Repeated rounds of this process of duplication and divergence, over many millions of years, have enabled one gene to give rise to a **family of genes** that may all be found within a single genome. Genes in two separate species that derive from the same ancestral gene in the last common ancestor of those two species are called **orthologs**. Related genes that have resulted from a gene duplication event within a

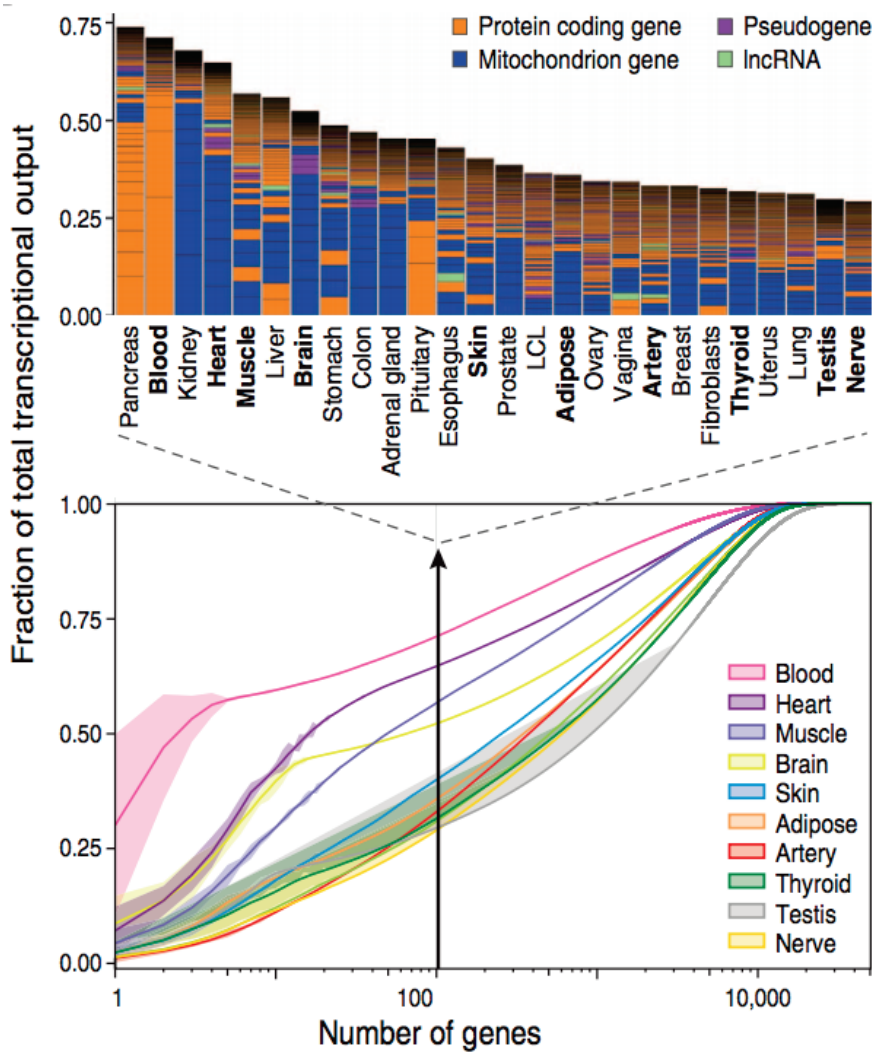


Figure 2.5: Human transcriptome complexity across several tissues. Top: Biological type and relative contribution to total transcription of the hundred most expressed genes. Height of the bars is proportional to the fraction that these genes contribute to total transcription. Bottom: Cumulative distribution of the average fraction of total transcription contributed by genes when sorted from most-to-least expressed in each tissue (x axis). Lines represent mean values across samples of the same tissue, and lighter-color surfaces around the mean represent dispersion calculated as the standard deviation divided by the cumulative sum of all means. Figure and caption adapted from [85].

single genome, and are likely to have diverged in their function, are called **paralogs**.

In Subsection 2.1.1, we introduced the process of RNA Splicing, in which the introns are removed from the pre-mRNA in one of the stages of mRNA production. However, the splicing machinery of a cell can splice the pre-mRNAs in different ways, by recognizing

different splice sites. This process is called **alternative splicing (AS)**, and allows the same gene to produce a corresponding set of different proteins. In higher eucaryotes, AS is not the exception, but the rule: it is estimated that more than 90% of genes in humans undergo AS [93, 129]. The most common pattern of AS is exon skipping. By skipping some exons, a eucaryotic gene with n exons could, in theory, produce an exponential number (on n) of different mRNAs, although generally only a fraction of these forms are experimentally observed. Figure 2.6 shows how AS can produce alternative proteins and Figure 2.7 explains the different patterns of AS. In the same way that different splice site choices can produce alternative transcripts from a same gene, so can alternative transcription sites. More specifically, this means that different transcription start or termination sites can cause a gene to produce a different protein. Such variations are termed **transcriptional variations**.

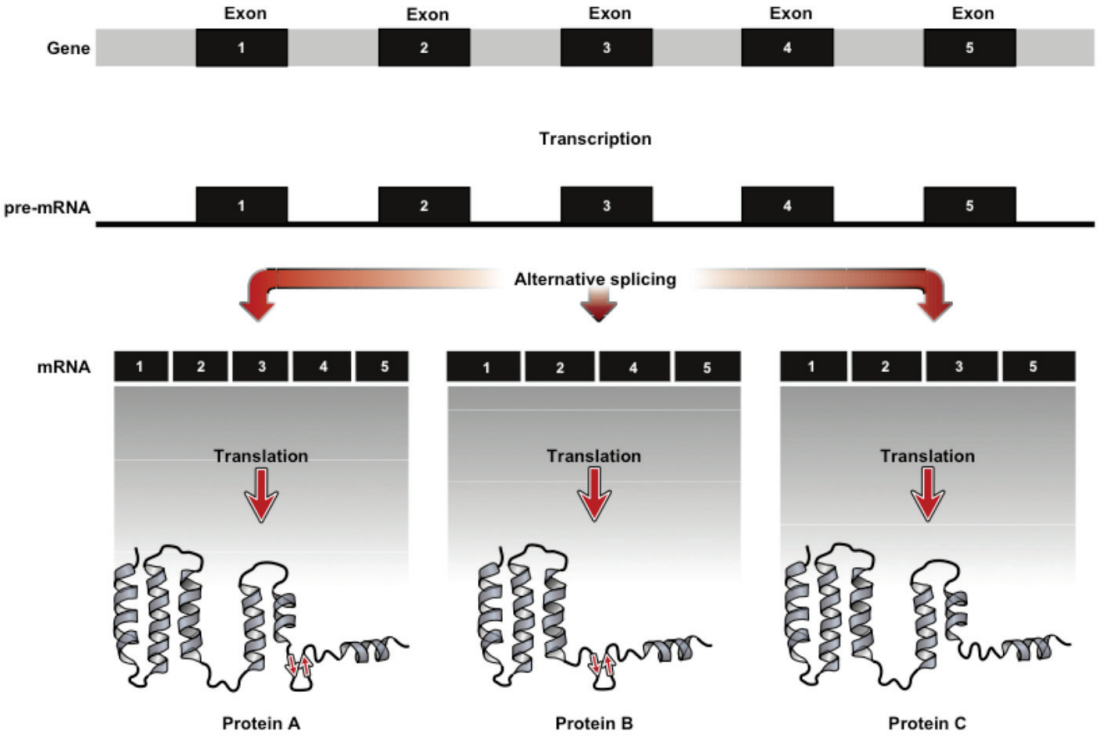


Figure 2.6: Alternative splicing creates different proteins. In the top, a gene is shown with coding regions (exons) in black and numbered, and non-coding regions in grey. In the middle, we have the pre-mRNA, which can generate different mRNAs and, consequently, proteins, through the process of alternative splicing. In this case, only exon skipping events take place: Proteins B and C skip exons 3 and 4, respectively. Figure reproduced from [113].

Variation identification and analyses are valuable because they might be linked to important phenotypes or conditions. Such analyses can be done in two contexts: genomic

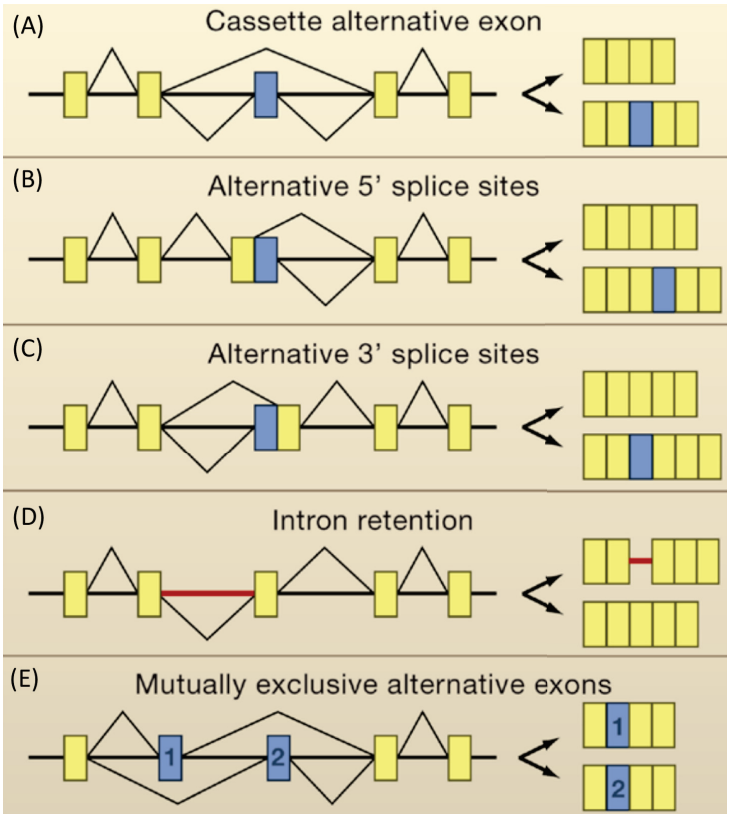


Figure 2.7: The different patterns of alternative splicing. Exons are shown as boxes and introns as straight lines. The zig-zag lines represent the removal (splicing) of a region. A gene (on the left) produces two different transcripts (on the right) in each subfigure. Constitutive exons (*i.e.* that are always included) are shown in yellow and alternative exons in blue. Red lines denote included introns. (A) Cassette alternative exon or exon skipping event: a full exon is skipped; (B) Alternative 5' splice (or donor) site: the endpoint of an exon is modified; (C) Alternative 3' splice (or acceptor) site: the start of an exon is modified; (D) Intron retention: a full intron is retained; (E) Mutually exclusive alternative exons: whenever exon (1) is retained, exon (2) is spliced and vice-versa. Figure adapted from [15].

and transcriptomic. **Genome-wide association studies (GWAS)** can identify which variations in a set of genomes are possibly linked to given phenotypes through a statistical framework, giving a measure of confidence on the inferences. As a concrete example, in one of the works in this thesis [48], we explore which genetic variants between a population of bacterial strains are highly associated with specific antibiotic resistance phenotypes, thus suggesting which mutations or horizontal gene transfers provide the resistance. In another context, transcriptomic analyses usually include the identification and quantification of gene expression, *i.e.* identifying which genes are being expressed and at which quantity. As we have seen, in all cells, the expression of individual genes is

regulated: instead of manufacturing its full repertoire of possible proteins all the time, the cell adjusts the rate of transcription and translation of different genes independently, according to its need. Given two or more conditions, finding a set of genes such that their expression are fairly contrasted between the conditions usually provides clues to understand the differences between the given conditions, *e.g.* such genes or their regulator genes can be directly related to the conditions. Such studies are termed **differential gene expression analyses**. In this thesis, however, we are more concerned with a finer-grained transcriptome variation analysis by studying differential AS expression analyses. This could provide clues on which AS events might be related to a given condition or phenotype.

2.2 Computer Science

We proceed by introducing computer science concepts. Many of them are excerpts reproduced, combined or adapted from [23]. As such, when a concept is described and no reference is explicitly given, the reader can assume [23] as reference.

2.2.1 Algorithms

An **algorithm** is a well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output. Algorithms are conceived to solve problems. We will usually call the input to a particular problem an **instance** of that problem. A classical example are sorting algorithms, which solve the sorting problem:

Sorting problem
Input: A sequence of n numbers $a_1, a_2, \ldots, a_n$.
Output: A permutation (reordering) $a'_1, a'_2, \ldots, a'_n$ of the input sequence such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

A **data structure** is a way to store and organize data in order to facilitate data access and modifications. Algorithms make use of data structures to solve complex problems efficiently. From a mathematical point-of-view, data structures can be seen as dynamic sets of elements, that can grow, shrink, or otherwise change over time. Algorithms may require several different types of operations to be performed on data structures. Such operations can be grouped into two categories: queries, which simply return information about the set (*e.g.* searching for an element, retrieving the minimum or maximum element), and modifying operations, which change the set (*e.g.* inserting or deleting an element).

Analyzing an algorithm consists in predicting the resources that it requires. In this thesis, we will generally analyze the running time of algorithms, and, in some cases, the memory usage, under a generic one-processor, random-access machine model of computation. The **running time** of an algorithm is the number of primitive operations executed, and we will describe it as a function of the size of its input. We will usually

concentrate on finding only the **worst-case running time**, that is, the longest running time for any input of size n. We will also be just interested in the order of growth of the running time, usually only on the asymptotic upper bound, denoted by the O-notation. For a given function $g(n)$, $O(g(n)) = \{f(n) : \exists$ positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$. The O-notation gives an upper bound on a function, to within a constant factor. We define an algorithm as **efficient** if its worst-case running time is polynomial, *i.e.* $O(n^k)$, where n is the size of the input and k is a constant.

Complexity

Problems can be categorized into different classes. Class P consists of problems that are solvable in polynomial time. Class NP consists of those problems that are verifiable in polynomial time, *i.e.* if we were given a certificate of a solution, then we could verify that the certificate is correct in time polynomial in the size of the input to the problem. Class NPC, or **NP-complete**, contains the problems that are in NP and are as "hard" as any problem in NP. Many problems of interest are **optimization problems**, in which each feasible solution has an associated value, and we wish to find a feasible solution with the best value. NP-completeness applies directly not to optimization problems, but to **decision problems**, in which the answer is simply yes or no. However, we usually can cast a given optimization problem as a related decision problem by imposing a bound on the value to be optimized. If we can provide evidence that a decision problem is hard, we also provide evidence that its related optimization problem is hard. If the decision version of a problem is shown to be NP-complete, we say that its optimization version is **NP-hard**.

Let us consider a decision problem A, which we would like to solve in polynomial time. Now suppose that we already know how to solve a different decision problem B in polynomial time. Finally, suppose that we have a procedure that transforms any instance α of A into some instance β of B with the following characteristics:

- The transformation takes polynomial time;
- The answers are the same. That is, the answer for α is yes if and only if the answer for β is also yes.

We call such procedure a **polynomial-time reduction algorithm** and, as Figure 2.8 shows, it provides us a way to solve problem A in polynomial time:

1. Given an instance α of problem A, use a polynomial-time reduction algorithm to transform it to an instance β of problem B;
2. Run the polynomial-time decision algorithm for B on the instance β;
3. Use the answer for β as the answer for α.

To prove the NP-completeness of a decision problem B, we follow a similar framework. We develop a polynomial-time reduction of a known NP-complete problem A into

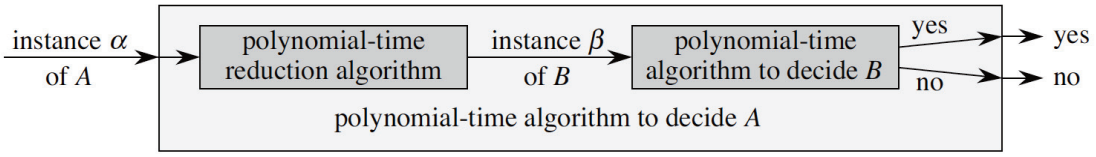


Figure 2.8: Using a polynomial-time reduction algorithm to solve a decision problem A in polynomial time, given a polynomial-time decision algorithm for another problem B. In polynomial time, we transform an instance α of A into an instance β of B, we solve B in polynomial time, and we use the answer for β as the answer for α. Figure reproduced from [23].

B. If we can solve B in polynomial time, then we can solve A in polynomial time and also all problems in class NPC. However, as these problems are unlikely to be solved in polynomial time, B is also unlikely. Since many problems have been proven to be NP-complete (a compendium can be found, for example, in [36]), this task can sometimes be simplified by choosing a "close" NP-complete problem.

Although we will introduce graphs and their related problems only in Subsection 2.2.3, we will take the liberty here of exemplifying the previous concepts with a graph problem. It is well known that the shortest path problem in directed graphs with non-negative arc weight is solvable in polynomial time [28]. More specifically, given a directed graph $G = (V, A)$ with non-negative arc weight, finding a shortest path from a source s to a target t can be done in polynomial time. A change to this problem, which seems to be "slight" at a first glance, produces an NP-hard problem. Let us say now that we want to find the longest path from s to t in G. Formally:

Longest path problem (LPP)
Input: A directed graph $G = (V, A)$ with non-negative arc weight, and two vertices s and t.
Output: The length of a longest simple path between s and t.

The first step to proving that the LPP is NP-hard involves conceiving a decision version for it, such as:

Longest path problem decision version (LPPD)
Input: A directed graph $G = (V, A)$ with non-negative arc weight, two vertices s and t, and an integer k.
Output: Is there a simple (s-t)-path $p \in G$ such that the length of p is $\geq k$?

There is a polynomial-time reduction from the Hamiltonian path problem to the LPPD [36]. The Hamiltonian path problem can be defined as follows:

Hamiltonian path problem (HPP)
Input: A directed graph $G = (V, A)$, and two vertices s and t.
Output: Is there a (s-t)-path $p \in G$ such that every vertex $v \in G$ is traversed exactly once?

A proof of the NP-completeness of the HPP can be found in [36]. To reduce the HPP to the LPPD, we transform $G = (V, A)$ into a weighted directed graph $G' = (V', A')$ such

that each arc $a \in A'$ has weight 1, and we set $k = |V'| - 1$. G', k and the original s and t vertices given as input to the HPP will compose our input to the LPPD. Finally, it is not hard to see that there exists a simple (s-t)-path $p' \in G'$ with length $|V'| - 1$ if and only if there exists a (s-t)-path $p \in G$ such that every vertex $v \in G$ is traversed exactly once. Thus we complete the proof, showing that the LPPD is NP-complete and, consequently, that its optimization version, LPP, is NP-hard.

The practical motivation of proving that a problem B is NP-Complete or NP-Hard is having a strong indication that it cannot be solved by a polynomial time algorithm. In other words, we can say that numerous very talented algorithm designers already tried for years to solve problems as equally hard as B, but were unable to do so. Thus, we can focus on searching for alternative solutions to the problem. Indeed, showing NP-completeness is usually not the end of the story, since many problems are too important to abandon merely because we do not know how to find an optimal solution in polynomial time. There are several ways to get around NP-completeness: i) if the actual inputs are small, an algorithm with exponential running time may be perfectly satisfactory; ii) we may be able to isolate important special cases that we can solve in polynomial time; iii) we might come up with approaches to find "good" solutions in polynomial time (approximation algorithms or heuristics). Succinctly, an **approximation algorithm** guarantees to find, in polynomial time, a solution to an instance of an optimization problem whose cost is within a pre-defined ratio in relation to the optimal cost. In other words, approximation algorithms guarantee to find solutions with a pre-specified quality in polynomial time. Much more can be said about approximation algorithms and their approximation ratios, but since the works in this thesis did not make use of such algorithms, we chose to not develop further. **Heuristics**, on the other hand, are criteria, methods and principles used to choose a path, among several, which is believed to be the most adequate in the search for an objective [96]. Heuristic-based algorithms are usually developed through the detailed study of problems, in order to acquire specialized knowledge which generally leads to good solutions. Although heuristics guarantees the output of feasible solutions, no restrictions are satisfied regarding execution time, or solution quality.

The theory of NP-completeness is much richer than our succinct presentation in the last paragraphs. The interested reader can find a comprehensive definition of this theory in a mathematically rigorous way in the book of Garey and Johnson [36].

Enumeration algorithms

Given an enumeration problem P and a set of constraints C, an **enumeration algorithm** A finds all feasible solutions for an instance I of P, *i.e.* all solutions satisfying C. For instance, given a directed graph $G = (V, A)$, listing all paths in G whose length is smaller than a constant k is considered an enumeration problem. An enumeration algorithm to this problem can be found in [103]. Usually, the number of feasible solutions for an instance I of an enumeration problem P can be exponential on the size of I and, as such, the complexity classes described in Subsection 2.2.1 cannot be applied to such problems, as they deal only with problems having polynomial-sized outputs. Johnson *et al.* in [52] thus defined new complexity classes to address such cases. In this thesis, we

are only interested in compact polynomial delay enumeration algorithms. A **polynomial delay enumeration algorithm** satisfies three properties: i) the time elapsed to output the first solution is polynomial on the input size; ii) the time elapsed between any two consecutive solutions is polynomial on the input size; iii) the time elapsed between the output of the last solution and the termination of the algorithm is polynomial on the input size. Fukuda *et al.* in [34] further defines that a **compact enumeration algorithm** is one whose space complexity is polynomial in the input size. We define here **compact polynomial delay enumeration algorithms**[1] as enumeration algorithms having both the properties from polynomial delay and compact enumeration algorithms.

2.2.2 Strings

An **alphabet** Σ is a finite set of characters. In this thesis, we always assume $\Sigma = \{A, C, T, G\}$, unless otherwise stated. A **string** s is a finite sequence of elements from an alphabet, *e.g.* AATTCTGTA is a string over Σ. Σ^* is the set of all strings over Σ. Given a string s, we denote its size by $|s|$. In the case that $|s| = 0$, then s is an **empty string**, also denoted by ϵ. The concatenation of two strings s and t, denoted by st, has length $|s| + |t|$ and consists of the characters from s followed by the characters from t. A string u is a **prefix** of a string s if $s = uv$ for some $v \in \Sigma^*$. Similarly, a string w is a **suffix** of s if $s = xw$ for some $x \in \Sigma^*$. Given a string $s \in \Sigma^*$, $s[i]$ denotes the i-th element of s, for any $1 \leq i \leq |s|$, and $s[i, j]$ the **substring** $s[i]s[i + 1] \dots s[j]$ for any $1 \leq i < j \leq |s|$.

There are several ways to formalize the notion of distance between two strings. Given two equal-length strings s and t, their **Hamming distance**, denoted by $d_H(s, t)$, is the number of positions i for which $s[i] \neq t[i]$ [77]. Another formalization is the edit distance, which focuses on transforming (or editing) one string into the other by a series of edit operations on individual characters [40]. The permitted edit operations are insertion of a character into the first string, the deletion of a character from the first string, or the substitution (or replacement) of a character in the first string with a character in the second string. The **edit distance** between two strings s and t is defined as the minimum number of edit operations – insertions, deletions, and substitutions – needed to transform the first string into the second. Note that matches are not counted [40]. The appropriate string distance measure to use depends on the studied problem. If only substitutions need to be modeled, then the Hamming distance can be an option. Otherwise, if insertions and deletions should also be taken into account, then the edit distance is more appropriate.

2.2.3 Graphs

A **directed graph** G is a pair (V, A), where V is a finite set and A is a binary relation on V. The set V is called the **vertex** set of G, and can also be referenced as $V(G)$. The set A is called the **arc** set of G, and can also be referenced as $A(G)$. In an **undirected graph** $G = (V, E)$, the **edge** set E consists of unordered pairs of vertices, rather than

[1] We note that this definition is not present in the literature. A similar concept can be found in [34], *(strongly) P-enumeration algorithms*, which are compact enumeration algorithms whose time complexity is linear in the output size.

ordered pairs. Many definitions for directed and undirected graphs are the same, although certain terms have slightly different meanings in the two contexts. If (u, v) is an arc in a directed graph $G = (V, A)$, we say that (u, v) is **incident from** or **leaves** vertex u and is **incident to** or **enters** vertex v. In the undirected case, we simply say that (u, v) is incident on the vertices u and v. If (u, v) is an arc in a directed graph G, we say that vertex v is **adjacent** to vertex u. When the graph is undirected, the adjacency relation is symmetric. Given a directed graph $G = (V, A)$ and a vertex $v \in V$, we denote its **out-neighbourhood** (resp. **in-neighbourhood**) by $N^+(v) = \{u \in V \mid (v, u) \in A\}$ (resp. $N^-(v) = \{u \in V \mid (u, v) \in A\}$), and its **out-degree** (resp. **in-degree**) by $d^+(v) = |N^+(v)|$ ($d^-(v) = |N^-(v)|$). The **degree** of v is defined as $d(v) = d^+(v) + d^-(v)$. In undirected graphs, the out- and in-neighbourhood coincides, and we have only the definition of **neighbourhood** of a vertex v, $N(v) = \{u \in V \mid (v, u) \in E\}$, and its **degree**, $d(v) = |N(v)|$. A vertex v is **branching** if $d^+(v) > 1$ or $d^-(v) > 1$ in directed graphs, or $d(v) > 2$ in undirected graphs. An example of an undirected and a directed graph can be seen in Figure 2.9.

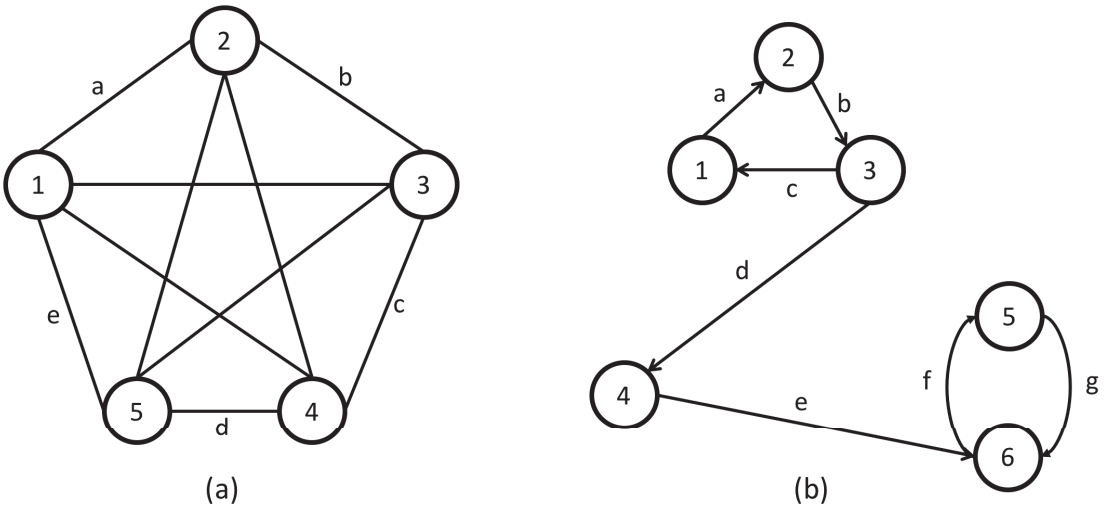


Figure 2.9: Examples of graphs. (a) A complete undirected graph with five vertices. Edge a is incident to vertices 1 and 2. All vertices have a degree of 4. (b) A directed graph. Vertex 6 has in-degree 2 and out-degree 1.

We say that a graph $G' = (V', E')$ is a **subgraph** of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given a subset of vertices $V' \subseteq V$, the subgraph of G **induced** by V', denoted by $G_{V'}$, has V' as vertex set and contains all edges of G that have both endpoints in V'. Given a subset of edges $E' \subseteq E$, the subgraph of G **induced** by E', denoted by $G_{E'}$, has E' as edge set and contains all vertices of G that are endpoints of edges in E'. Given a subset of vertices $V' \subseteq V$ and a subset of edges $E' \subseteq E$, we denote by $G \setminus V'$ the graph induced by $V \setminus V'$ and by $G \setminus E'$ the graph induced by $E \setminus E'$. Given two graphs G and H, their union $G \cup H$ is the graph F for which

$V(F) = V(G) \cup V(H)$ and $E(F) = E(G) \cup E(H)$. Their intersection $G \cap H$ is the graph F for which $V(F) = V(G) \cap V(H)$ and $E(F) = E(G) \cap E(H)$.

An undirected graph $G = (V, E)$ is **weighted** if there is a function $\omega : E \to \mathbb{R}$, associating a weight (or cost) to every edge in the graph. In **unweighted** graphs, we do not have a weight associated to edges. However, most algorithms on unweighted graphs behave equivalently on the weighted version of the graph, where the weight of each edge is 1. We will assume this and, to simplify, we then associate a function $\omega : E \to 1$ to unweighted graphs. The concepts of weighted and unweighted graphs also apply to directed graphs. A **path** from a vertex v_0 to a vertex v_k in an undirected graph $G = (V, E)$ is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that $(v_{i-1}, v_i) \in E$ for $i = 1, 2, \ldots, k$. The length of p is $|p| = \sum_{e=(v_{i-1},v_i)\in p} \omega(e)$. We assume there is always a 0-length path from u to u. If there is a path p from u to v, we say that v is **reachable** from u. A path is **simple** if all vertices in the path are distinct. All paths considered here will be simple, unless otherwise stated. A **subpath** of path $p = v_0, v_1, \ldots, v_k$ is a contiguous subsequence of its vertices. We say that the subpath $p_1 = v_0 \ldots, v_i$ ($p_2 = v_j, \ldots, v_k$) is a **prefix** (**suffix**) of p for some $0 \leq i \leq k$ ($0 \leq j \leq k$). A path in a directed graph is called a **directed path**, and all previous definitions on paths can also be applied to directed paths. An undirected graph is **connected** if every vertex is reachable from all other vertices. The **connected components** of an undirected graph are its maximal connected subgraphs. A directed graph is **strongly connected** if every two vertices are reachable from each other. The strongly connected components of a directed graph are its maximal strongly connected subgraphs. As examples, the graph in Figure 2.9a is connected and thus contains only one connected component. The graph in Figure 2.9b is not strongly connected, and its strongly connected components are: $\{1, 2, 3\}, \{4\}, \{5, 6\}$.

In a directed graph, a path $p = v_0, v_1, \ldots, v_k$ forms a **directed cycle** if $v_0 = v_k$ and the path contains at least one arc. The directed cycle is simple if, in addition, $v_1, \ldots, v_k$ are distinct. All directed cycles considered here will be simple, unless otherwise stated. Given a directed graph G and two distinct vertices $s, t \in V(G)$, an (s, t)-**bubble** consists of two (s, t)-simple-directed-paths that are internally vertex disjoint. Vertex s is the source and t is the target of the bubble. In two of our papers [1,2], we allow some cases where $s = t$. In such cases, one of the paths of the bubble has length 0, and therefore B corresponds to a directed cycle. We then say that B is a **degenerate bubble**. Unless explicitly stated, the term bubbles reference only bubbles themselves, not degenerate bubbles. A **cycle** in an undirected graph G is a subgraph of G such that all its vertices have even degree. Note that our definition of cycles and directed cycles are very different, *i.e.* a cycle can even be a disconnected graph, whereas a directed cycle is connected by definition. We chose this alternative definition for cycles in undirected graphs in order to be compatible with the definition adopted by the community of cycle basis, since in two papers of this thesis [1, 2], we work with this specific definition of undirected cycles. A cycle that is connected and for which all the vertices have degree 2 is called an **elementary cycle**. An elementary cycle is similar to directed cycles, but in undirected graphs. Figure 2.10 exemplifies cycles, elementary cycles, directed cycles and bubbles.

A **tree** is a connected, acyclic, undirected graph. If an undirected graph is acyclic

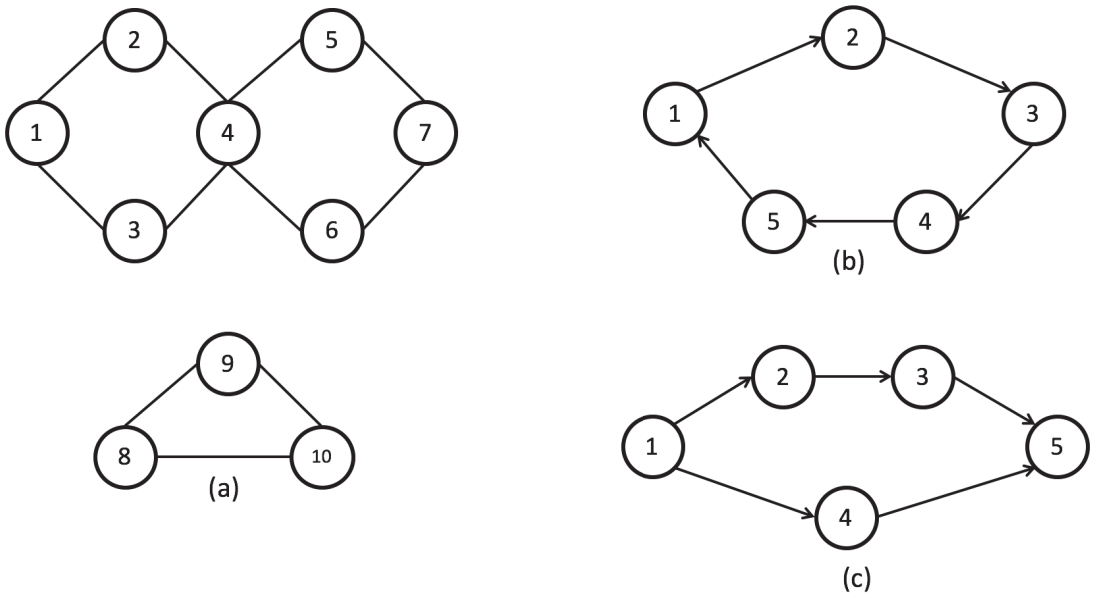


Figure 2.10: Examples of cycles, elementary cycles, directed cycles and bubbles. (a) A cycle with three elementary cycles: (1, 2, 3, 4), (4, 5, 6, 7), (8, 9, 10). (b) A directed cycle. (c) A bubble.

but possibly disconnected, it is a **forest**. A **rooted tree** is a tree in which one of the vertices is distinguished from the others. We call the distinguished vertex the **root** of the tree. In this thesis, when we refer to rooted trees, we usually direct its arcs from the parents to their children. An **orientation** of an undirected graph G is a directed graph G' such that G' is a copy of G with oriented edges. Given a directed graph G', the **underlying undirected graph** G of G' is a copy of G' without the orientation of the arcs. A **biconnected** undirected graph G is a connected graph such that, for any $v \in V(G)$, $G - v$ is connected. A **biconnected component (BCC)** is a maximal biconnected subgraph of a graph G. Figure 2.11 exemplifies all these concepts.

Given a set of elements $\mathcal{E}$, representing an **element space**, an addition (or combination) operator $\mathcal{O} : E \times E \to E$, and a subset $\mathcal{G}$ of $\mathcal{E}$, the set of all elements that can be generated using $\mathcal{O}$ starting from elements in $\mathcal{G}$ is called the **span** of $\mathcal{G}$. If $\mathcal{G}$ spans $\mathcal{E}$, then $\mathcal{G}$ is called a **generator**. Further, if $\mathcal{G}$ is minimal, *i.e.* there is no element $e \in \mathcal{G}$ which can be obtained by adding elements in $\mathcal{G} - e$, then $\mathcal{G}$ is called a **basis**. We can apply these notions to cycles. Given an undirected graph $G = (V, E)$, we can combine two cycles C_1 and C_2 of G through the **symmetric difference operator**, denoted as $+$. The symmetric difference operator applied to C_1 and C_2 produces a subgraph C_3 induced by the edges of $(E(C_1) \cup E(C_2)) \setminus (E(C_1) \cap E(C_2))$, which is again a cycle [39]. The **cycle space** $\mathcal{C}$ of G is the set of all cycles of G. The cycle space is thus closed under the symmetric difference operator, as $C_1 + C_2$ always results in a cycle, for any $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$, and can be represented as a vector space over $\mathbb{Z}_2^E$, where $\mathbb{Z}_2^E$ is the field of

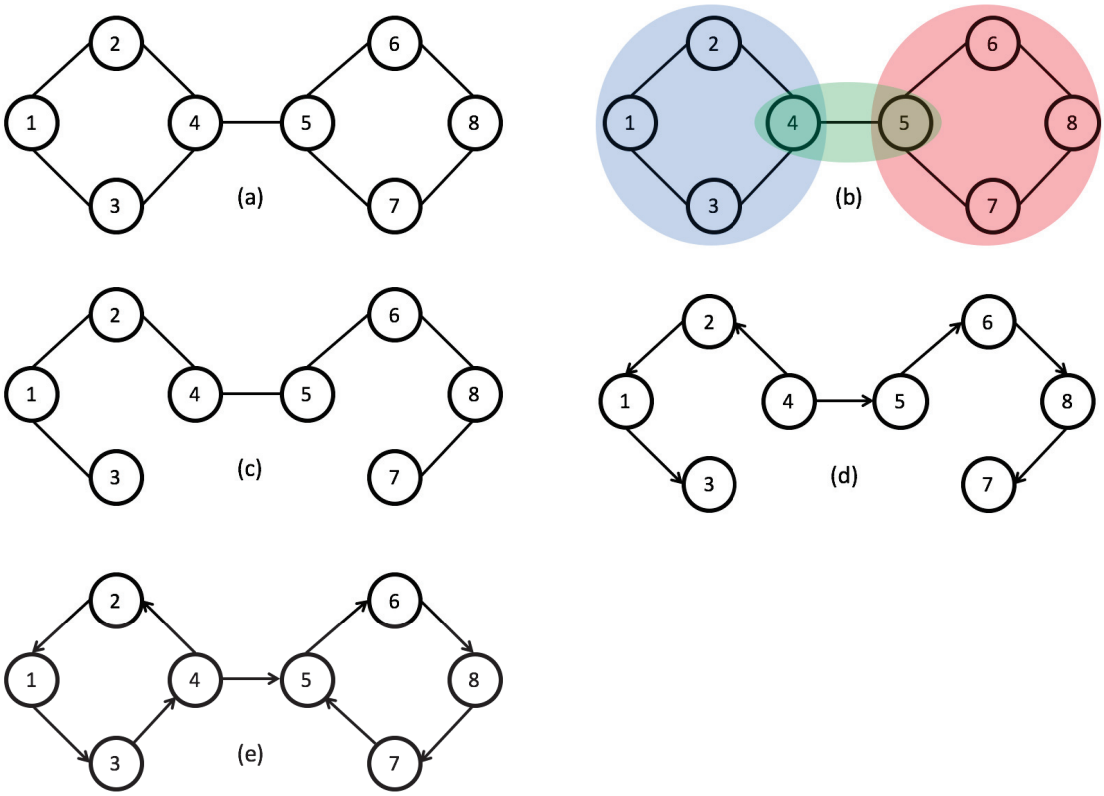


Figure 2.11: Examples of BCCs, tree, rooted tree, oriented graph and underlying undirected graph. (a) An undirected graph G. (b) The three BCCs of G. (c) A tree of G. (d) A rooted tree T of G with root $r = 4$. (e) A directed graph G', which is an orientation of G. G is also the underlying undirected graph of G'. Note that T is also an oriented tree of G'.

two elements indexed by E [54]. Given a cycle C, the vectorial representation V of C is such that $V_e = 1$ if and only if $e \in C$. The addition and the multiplication by a scalar of cycles represented as vectors can be defined as the addition and the multiplication by a scalar of integer vectors taken modulo 2. An intuitive way of finding a cycle basis is to represent the cycle space as a vector space. As such, the vector basis of the corresponding cycle space is a **cycle basis**. In informal terms, cycle bases are a compact description of the set of all cycles of a graph. Figure 2.12 shows the combination of two cycles through the symmetric difference operator, and the cycle basis of a graph G.

As with cycle bases in undirected graphs, we can also define a symmetric difference operator, but whose operands are bubbles. Given two bubbles B_1 and B_2, the constrained symmetric difference operator Δ is such that $B_1 \Delta B_2$ is defined if and only if the subgraph induced by the arcs of $(A(B_1) \cup A(B_2)) \setminus (A(B_1) \cap A(B_2))$ is a bubble. Otherwise, we say that $B_1 \Delta B_2$ is undefined. Since not all pairs of bubbles of G are combinable (*e.g.* a trivial example is two disjoint bubbles), the bubble space is not closed under Δ, and

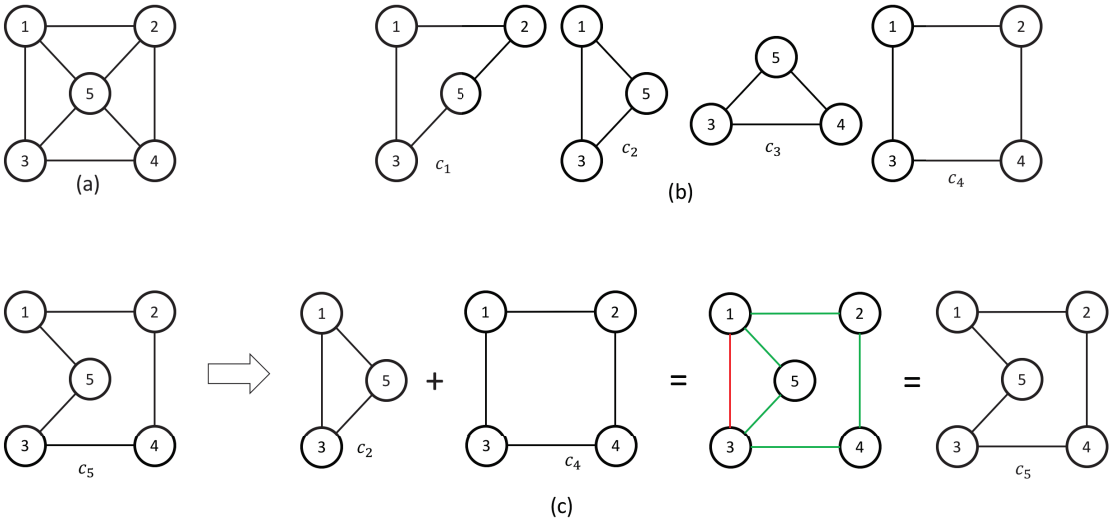


Figure 2.12: Cycle basis and cycle combinations. (a) An undirected graph G. (b) The cycles c_1-c_4 compose the basis of the cycle space of G. (c) A cycle c_5 of G is the combination of cycles c_2 and c_4. During the combination, the edges to be removed (present in both c_2 and c_4) are highlighted in red, and the edges to be kept (present in either c_2 or c_4) are highlighted in green.

therefore it does not form a vector space over $\mathbb{Z}_2^E$. Thus, as the bubbles space cannot be represented by a vector space, the techniques used to find a basis of a vector space cannot be applied to find a basis of the bubble space. New techniques must be developed, and we explore some of these in two papers in this thesis [1,2].

2.3 Bioinformatics

The last section of this chapter introduces bioinformatics concepts. The definitions here described will make use of those contained in both Sections 2.1 and 2.2.

2.3.1 DNA and RNA sequencing

DNA sequencing is the process of obtaining the DNA sequence from a set of cells. Likewise, RNA sequencing is the process of obtaining the sequences of the mRNAs that are being expressed by a set of cells. More precisely, mRNAs are extracted and complementary DNA (cDNA) chains are synthesized by the reverse transcriptase enzyme from some template mRNAs, which are then sequenced. Although it is also possible to sequence mRNAs directly [35,92], in this thesis whenever we refer to RNA sequencing, we mean cDNA sequencing, unless otherwise stated. DNA and RNA sequencing are fundamental inputs to many bioinformatics problems: many projects and studies take sequenced data as the starting point to answer complex biological questions. The output of sequencing is a set of strings called **reads**, which are substrings of the nucleotide sequence of the

chromosomes, in the case of DNA, or of the mRNAs (cDNAs), in the case of RNA. We will not describe how these sequencing technologies work, since this can be very technical and out of the scope of this thesis. We will mainly focus on the characteristics of their output, and on their cost. The interested reader can find more details in [116] and [37]. We can define three major breakthroughs in sequencing technology: Sanger Sequencing, Second generation sequencing (2GS) and Third generation sequencing (3GS).

Sanger Sequencing

The **Sanger Sequencing** was the first practical sequencing technology. It is mainly used in two contexts: shotgun *de novo* sequencing and targeted resequencing. In shotgun *de novo* sequencing, DNA is randomly fragmented and then cloned into a high-copy-number plasmid. In targeted resequencing, PCR amplification is carried out with primers that flank the target. The output of both approaches is an amplified template, which then goes through two other processes (abstracted in this text, but details can be found in [116]) until the final sequences. Sanger sequencing can achieve read-lengths of up to 1,000 bp, and per-base raw accuracies as high as 99.999% [116]. The main problem with Sanger Sequencing is its cost. As highlighted in [116], Sanger sequencing might require expensive reagents, processing of multiple samples in 96- or 384-well formats, maintenance of capillary-based sequencers, extensive bioinformatics infrastructure to handle the flow of data and dedicated support staff to maintain complicated equipment. In an informal survey done by Shendure *et al.* in [116], the overall cost to conventionally sequence the DNA sequences of 100 genes from 100 samples, assuming each gene has an average of 10 exons, ranged from $300,000 to over $1,000,000. Shendure and Ji [116] estimate that the cost of Sanger sequencing is on the order of $500,000 per Gbp. This cost is beyond the range of most individual laboratories.

Second generation sequencing (2GS)

The **Second generation sequencing (2GS)** was conceived to i) reduce the per-base cost of sequencing by several orders of magnitude, and ii) reduce the infrastructure requirements for sequencing. However, this improvement also comes with some disadvantages: 2GS reads are a lot shorter than Sanger reads, and their error-rate is higher. Therefore, the focus of genomic studies have changed. Before, the hardest task was to generate data. Due to the low cost of 2GS, many research institutions and laboratories were able to sequence DNAs at an affordable cost. 2GS also popularised transcriptomic studies with RNA sequencing. As such, 2GS was heavily employed in several applications other than *de novo* genome assembly, such as: large-scale and targeted polymorphism discovery, discovery of inherited and acquired structural variation, quantification of gene expression and alternative splicing, microRNA profiling, genome-wide mapping of protein-DNA interactions, etc (see [116]). The low cost of 2GS data and their wide application resulted in innumerous 2GS datasets being produced and made available to the community. The huge amount of data sequenced in each experiment and the fact that 2GS reads are shorter and less accurate than Sanger reads motivated the develop-

ment of a plethora of algorithms and methods to efficiently process this type of data. 2GS approaches fall under two broad categories: sequencing by ligation and sequencing by synthesis [37]. To keep this section succinct, we will skip the 2GS technical details and focus in one specific sequencing by synthesis technology: Illumina. This choice is supported by a practical motivation: the large majority of available 2GS datasets in the literature is Illumina data, as are all the 2GS datasets in the works performed in this thesis. It is worth observing that in this sequencing technology, DNA/RNA molecules are also fragmented into millions of pieces, and the fragments' ends are sequenced. Although the Illumina instruments are unable to sequence reads as long as Sanger sequencing, they can provide **paired-end reads**, *i.e.* a pair of reads coming from both ends of one specific fragment. Further, it is also possible to know the approximate distance between paired-end reads, which can help algorithms and methods to process such data. We note that fragments are obtained through random breakage, and although we can select a range of fragment length (by migrating on a gel), fragment length cannot be precisely chosen. However, fragment lengths follow a normal distribution, with a mean usually in the selected range. Current common Illumina sequencers, like Illumina HiSeq 4000, are able to sequence 150-bp paired-end reads, with a throughput of 650-750 Gbs, a 0.1% substitution error-rate, with an approximate cost of $22 per Gb [37]. There are, however, several models of Illumina sequencers, like the less powerful benchtop series (iSeq, MiniSeq, MiSeq, NextSeq), and the new NovaSeq series, which outperforms the HiSeq series [47]. However, we will assume Illumina HiSeq 4000 as the default 2GS sequencing instrument (in this thesis, custom 2GS sequencing involved a Illumina HiSeq 2500 in [11] and Illumina HiSeq 4000 in [76]). 2GS sequencers present two characteristic disadvantages when compared to Sanger sequencing: i) 2GS read length is shorter[2] (the Illumina instruments reach a maximum of 300 bps compared to 1000 bps from Sanger); ii) 2GS error rate can be 100-fold higher than the Sanger's error rate (0.1% compared to 0.001%) [37]. However, these disadvantages are overcome by its two main advantages: i) more than 20,000 times cheaper (Illumina HiSeq 4000 costs around $22 per Gb, while Sanger sequencing around $500,000 per Gb); ii) far simpler infrastructure requirements for sequencing.

Third generation sequencing (3GS)

Many complex genomes contain several long repetitive elements, copy number alterations and structural variations. In many cases, these elements are so long that 2GS reads are insufficient to resolve them (this issue will be further explored in Subsection 2.3.2). **Third generation sequencing (3GS)** are recent sequencing technologies producing reads with several kilobases, allowing for the resolution of these large structural features. Such long reads can span complex or repetitive regions with a single continuous read, thus eliminating ambiguity in the positions or size of genomic elements. Long reads can also be useful for transcriptomic research, as they are capable of spanning entire

[2] 454 pyrosequencing reached up to 1000 bps, 700 on average, but their very high cost was not competitive, and, as such, these sequencers are not available anymore [37].

mRNAs, allowing researchers to identify the precise connectivity of exons and discern gene isoforms [37].

There are two main 3GS technologies: single-molecule real-time sequencing (SMRT) and synthetic approaches. The latter rely on 2GS technologies to produce long reads *in silico*. In this thesis, we will describe only the former, SMRT sequencing, since it is only the type of 3GS technology explored in this thesis (in [75, 76]).

The two dominant SMRT sequencing approaches are Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT). The main problem of both technologies are: i) high error rate, from 10 to 15%; ii) cost higher than Illumina (1 to 2 orders of magnitude); iii) lower throughput than Illumina. It is not easy to keep track of the technological improvements of both these technologies. They are evolving so fast that a review from 2016 [37], for example, could be considered outdated nowadays. Therefore, we will not use papers to refer to the characteristics of the PacBio and ONT sequencing instruments, but communications from the companies themselves. Although this can provide an updated source of information, it can also be biased.

In PacBio sequencing, the high error rate can be dropped down, even to the level of Illumina error rate. A sensitive parameter to produce highly accurate PacBio reads is the size-selection, which will select which target sequences will be sequenced based on their size. The shorter a target sequence is, the lower its error rate will be. Skipping details, before sequencing takes place, the target sequence is transformed in a circular sequence. This circular sequence can be sequenced over and over again if the read length exceeds the target length. For example, if the current read being produced will have around 50kb, and the target sequence is 5kb long, then the raw PacBio read, called **continuous long read (CLR)**, will contain around 10 copies of the target sequence. Each such copy is called **subread**, and the consensus of all subreads is a **Read of Insert (ROI)**. However, if a target sequence is too long to be sequenced multiple times, only a (partial) single subread is generated. A nice feature of PacBio sequencing is the absence of systematic sequencing errors: errors are randomly distributed. Therefore, if a CLR contains enough copies of a target sequence, its ROI will represent a fragment of DNA of several kbs and low error rate (>99% accuracy can be reached with 15 copies [31, 102]). As expected, longer target sequences yield fewer copies in a CLR, and therefore produce less accurate ROIs. In a PacBio communication [13], it is shown that the most recent PacBio instrument (PacBio Sequel) outputs reads (CLRs) with an average length of 30kb. In one experiment, by size-selecting target DNAs with more than 20kb, half of the output base pairs are in reads with more than 45kb (also known as N50 > 45kb), and the length of the 5% longest reads exceeds 150kb. If the size-selection procedure targets sequences with less than 20kb, N50 > 190kb, and the length of the 5% longest reads exceeds 280kb.

ONT sequencing has, theoretically, no instrument-imposed limitation on the size of reads that can be generated [62]. However, recent results show similar read lengths as PacBio's. Ultralong reads protocols, such as the one described in [51], achieved an average read length of 24kb and N50 around 100kb. However, ONT also achieved the first ever >1 Mbp read in December 2017 [121]. The main advantages of ONT in relation to PacBio are: i) portability – one of the ONT instruments, MinION, weighs under

100g, and plugs into a PC or laptop, no additional computing infrastructure is required, thus its usage is not constrained to a laboratory environment; ii) lower instrument cost; iii) ability to sequence RNA directly. The main disadvantages are: i) unlike PacBio, ONT presents systematic indel sequencing errors in homopolymer runs (sequences of consecutive bases, in practice, runs with more than 6 bases can already be problematic); ii) ONT's $1D^2$ protocol reduces errors by sequencing a target DNA twice (similar to PacBio's ROIs). However, it is unable to lower the error rate to the PacBio level due to systematic sequencing errors and the number of copies being too small (maximum of two).

Main differences between DNA and RNA sequencing

While DNA and RNA sequencing share a lot of similarities, they also present striking differences. In DNA sequencing, all sequencing technologies explained in the previous subsections generally achieve an almost uniform coverage of the genome, *i.e.* at any given position of the genome there is an average number of reads covering it (*e.g.* on a 50x sequencing, we have on average 50 reads covering any position of the genome). In RNA sequencing, we do not have a uniform read coverage of the expressed transcripts, due to the different levels of isoform expression. Some transcripts are highly expressed, and therefore highly covered, while others are lowly expressed and thus lowly covered. Sequencing errors derived from reads from a highly expressed transcript may be more abundant than correct bases derived from reads from a transcript that is not highly expressed [38]. Therefore, dealing with errors in RNA sequencing is more complicated than in the DNA context. Moreover, read coverage may be uneven across the transcript's length, owing to sequencing biases [38]. The mRNA can also be degraded when collected to be sequenced, which results in partial transcript sequencing, mostly observed with 3GS (staircase effect). Further, the most commonly used protocol to extract RNA yields pre-mRNA fractions between 5 and 15% [122]. This small mix of pre-mRNA can cause issues on processing the data, if not addressed correctly [77]. While 3GS has the power to provide the full transcript structure, the large majority of mRNAs do not exceed a few kilobases. Therefore, having the ability to sequence very long reads might not be as useful as in the DNA scenario. However, PacBio takes advantage of sequencing reads longer than transcripts by having several copies of a target mRNA in a single CLR, creating very accurate ROIs, even for poorly expressed genes. Size selection is however an issue, as it favours some transcripts over others, and produces biased quantifications.

2.3.2 Processing of 2GS and 3GS data

In this section, we describe some means to process 2GS and 3GS data. We will restrict ourselves only to the methods and analysis pipelines in the scope of this thesis.

References

A **reference genome** is a set of strings that try to represent the nucleotide sequences of an organism's chromosomes. The reference genome is usually the best assembled genome for a given species, normally built using high-quality data from different sequencing machines and several assembly methods, validated by post-assembly analyses. However, it is still an haploid version of the genome of a single random individual, and does not represent the polymorphisms present in this individual, and in population from which this individual was extracted. In many applications, the value of the genome is only as good as its annotation [118]. **Genome annotation** is the process of taking the raw DNA sequence produced by the genome-sequencing projects and adding layers or tracks with biological information and interpretation about specific fragments of DNA [118]. A concrete example is taking a newly sequenced genome and identifying which segments of the DNA sequence correspond to genes, and repeats, for instance. In addition, the genes can be further annotated by identifying novel transcription and splice sites, and the function of each alternative transcript. Repeats can be classified into different families, based on their similarity. Finally, the aim of high-quality annotation is to identify the key features of the genome: genes, splicing sites, alternative transcripts, non-coding RNAs, regulatory regions, repetitive elements, variations, etc [118]. A **reference transcriptome** is the set of known mRNA sequences of an organism. It can be obtained from the genome annotation.

Read mapping or read alignment

Read mapping or read alignment is the process of determining the most likely source within a reference genome sequence for the observed sequencing read, given the knowledge of which species the read has come from. In the absence of a reference genome for the studied species, sequencing reads may also be aligned to other genomes, assuming the evolutionary distance between the genomes is appropriate [32]. Genomic read mapping algorithms aim at determining the fragments of a genome that are very similar to a given read, *i.e.* the edit distance between the read and such fragments must be small, bounded by a function on the species polymorphism rate and the sequencing technology error rate [32]. In one situation, aligning DNA and RNA-sequencing reads to a reference genome can be very different. This happens when a RNA-sequencing read spans two or more exons. In this case, the mapping algorithm must be aware that a read can be mapped in a genome with long gaps, which represent the introns between the spanned exons. Methods implementing such algorithms are known as **splice-aware mappers**. In some works of this thesis, we make use of such mappers to align 2GS and 3GS RNA sequencing reads. Among the most appropriate splice-aware mappers, we can cite: BBMap [18], gmap [130], Hisat2 [56], minimap2 [71], STAR [29], and Tophat2 [57].

Transcriptome and genome assembly from 2GS data

Transcriptome (genome) assembly is the task of assembling the original transcriptome (genome) from a set of sequenced reads. There are two approaches for transcriptome

(genome) assembly: reference-based and *de novo*. Here we will focus mainly on *de novo* transcriptome assembly from 2GS data. Assembling transcriptomes from short reads is not a trivial task. Grabherr *et al.* [38] highlight some of these difficulties: (i) some transcripts have low coverage, whereas others are highly expressed; (ii) read coverage may be uneven across the transcript's length, owing to sequencing biases; (iii) reads with sequencing errors derived from a highly expressed transcript may be more abundant than correct reads from a transcript that is not highly expressed; (iv) transcripts encoded by adjacent loci can overlap and thus can be erroneously fused to form a chimeric transcript; (v) data structures need to accommodate multiple transcripts per locus, owing to alternative splicing; and (vi) sequences that are repeated in different genes introduce ambiguity.

A **reference-based transcriptome assembly** algorithm uses alignments of reads to the genome to identify clusters of reads that represent potential transcripts. It then builds transcript assemblies from these alignments. If paired-end reads are available, they improve the ability of the assembler to link together exons belonging to the same transcript [99]. Some advantages of reference-based strategies are: i) very high sensitivity, since a few reads mapping to a known isoform can be enough evidence for its identification; ii) performance, since the underlying algorithms of such methods are aided by high-quality references, usually translating into faster methods. Their disadvantages include: i) the resulting assemblies might be biased towards the used reference, and true variations might be discarded in favour of known isoforms; ii) unsuitable for samples with a partial or missing reference genome [38]; iii) such methods depend on correct read-to-reference alignment, a task that is complicated by splicing, sequencing errors, polyploidism, multiple read mapping, mismatches caused by genome variation, and the lack or incompleteness of many reference genomes [38, 105]; iv) sometimes, the model being studied is sufficiently different from the reference because it comes from a different strain or line such that the mappings are not altogether reliable [114]. Some of the state-of-the-art methods for reference-based transcriptome assembly are: Cufflinks [126], MISO [53], Scallop [115], Scripture [41], StringTie [99], Traph [124], and Traphlor [60]. ***De novo* transcriptome assembly**, on the other hand, uses only the information from the reads, being agnostic to any additional external information. Some advantages of *de novo* strategies are: i) they do not require any read-reference alignments, important when the genomic sequence is not available, is gapped, highly fragmented or substantially altered, as in cancer cells [38]; ii) the fact that they are applicable to the discovery of transcripts that are missing or incomplete in the reference [42]. The disadvantages include: i) the assembly of short reads is itself difficult, and only the most abundant transcripts are likely to be fully assembled [42]; ii) reconstruction heuristics are usually employed, which may lead to missing infrequent alternative transcripts while highly similar transcripts are likely to be assembled into a single transcript [84]; iii) *de novo* methods usually require far more computational power than reference-based strategies. Some of the state-of-the-art methods for *de novo* transcriptome assembly are: Oases [114], SOAPdenovo-Trans [132], Trans-ABySS [105] and Trinity [38].

Reconstructing full-length transcripts from short reads in a *de novo* context is chal-

lenging. The outcome of this process could lead to misassemblies, like chimeric or truncated transcripts. An assembled **chimeric transcript** is an artificial isoform composed of parts from two or more real isoforms. Chimerism usually happens when assemblers try to infer the correct assembly, despite not having enough information to do so. Even with perfect reads and uniform coverage across the transcripts, alternative splicing might induce ambiguities in the assembly process that short reads data are incapable to solve. This is particularly true when two transcripts have similar expression. A small example of full-length transcriptome assembly difficulty, which can lead to chimerism, can be seen in Figure 2.13. Sometimes, some assemblers choose the alternative to be conservative, *i.e.* not extending the assembled sequence due to the risk of creating chimeric transcripts, but then producing truncated transcripts. An assembled **truncated transcript** is a partial isoform of a real isoform. Although such conservative strategy can lead to an accurate assembly, it will produce a very fragmented one, which is not desirable. Transcript truncation can be observed mainly when transcripts share inter-gene repeats, *i.e.* repeats present in several unrelated genes. In such cases, the choice an assembler has to make is far more complicated than the theoretical situation depicted in Figure 2.13, occasionally leading to the assembler making no choice and thus truncating the transcript. As a real example, Figure 2.14 shows a transcript assembled by Trinity [38] in a real dataset whose end was truncated due to a high-copy-number repeat. Many factors contribute to the hardness of full-length transcriptome assembly. The main issue is certainly that reads are short, and can therefore be ambiguously assigned to multiple transcripts. In particular, in the case of alternative splicing, reads stemming from constitutive exons can be assigned to any alternative transcript containing this exon. Finding the correct transcript is often not possible given the short read data, and any choice will be arguable [107].

The issue presented in Figures 2.13 and 2.14 is not specific to transcriptomics. We have a similar scenario in genomics, when sequencing reads (or fragments) are shorter than repeats. This short read/long repeat issue is an old problem that has been around since the first algorithms for genome assembly. Even though the problems repeats cause in both DNA and RNA contexts are similar, they have also some characteristics that are specific to each. In genome assembly, repeats tend to be longer and present in more copies. In transcriptome assembly, constitutive exons between different isoforms from a same gene can be regarded as repeated sequences connecting unique regions. Paralog genes compose a special case, which can affect both contexts. Such genes can indeed be seen as genomic repeats, and if more than one copy of a paralogous family is expressed, it can also cause repeat-related issues to transcriptome assembly. Further, genomic repeats can also be located within genes, although they tend to be shorter and in fewer copies. This is specially true when genes host transposable elements within their introns, and less frequently but still present, within their UTRs and also as exons (*e.g.* exonised repeats). Even if a repeat-containing intron is spliced out in the splicing phase, this intron, and consequently the repeat, can still be present in RNA-seq data. The most commonly used RNA extraction protocol yields pre-mRNA fractions around 5% [122]. Thus, more introns than expected are sequenced, generating problems for transcriptome assemblers, particularly when such introns contain several members of a specific repeat

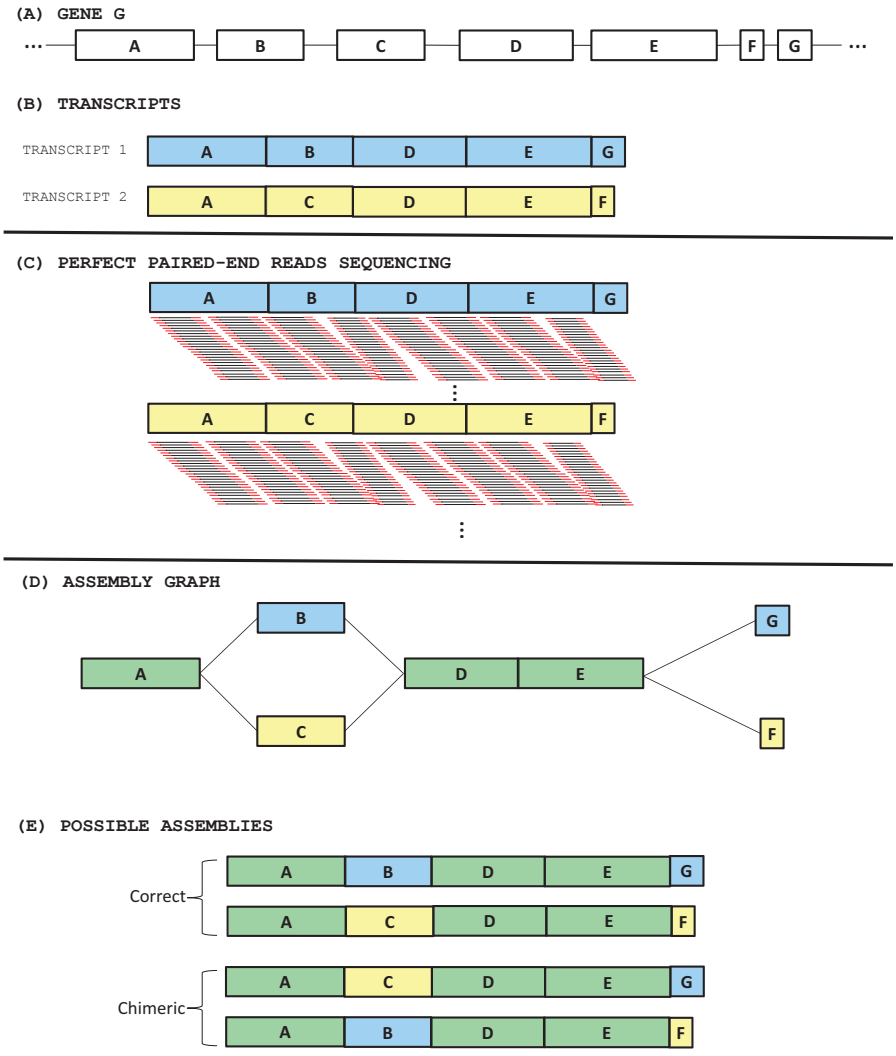


Figure 2.13: A theoretical scenario showing a small example of full-length transcriptome assembly difficulty, which can potentially lead to chimerism. (A) A gene `G` with its exons represented as boxes and introns as lines (this example is not realistic since normally introns are way longer than exons, but we will take the liberty of drawing very short introns due to illustration purposes); (B) The two expressed transcripts from `G`; (C) A perfect paired-end sequencing of both transcripts: error-free, uniform, and deep read coverage through the transcripts; (D) Any *de novo* assembler, be it based on graphs or other models, will eventually have to make decisions which are equivalent on how to assemble paths in the depicted assembly graph. Observe that the vertices' colouring, which allows us to differentiate which vertices are common and unique to each transcript, are not available to assemblers; (E) The possible assemblies can be correct or chimeric. Since the length of the fragments is shorter than the common region (exons `D` and `E`) of both transcripts that connects their unique regions (exons `B`, `C`, `F`, and `G`), there is no read information connecting exons `B` and `G` or exons `C` and `F`. In this situation, assemblers might use the coverage information to distinguish unique regions from both transcripts and try to correctly assemble them. However, even in this simple example, if transcripts have similar expression, it is unclear how to differentiate the correct assemblies from the chimeric ones. In real datasets, heterogeneity of coverage across the transcripts makes this task even harder.

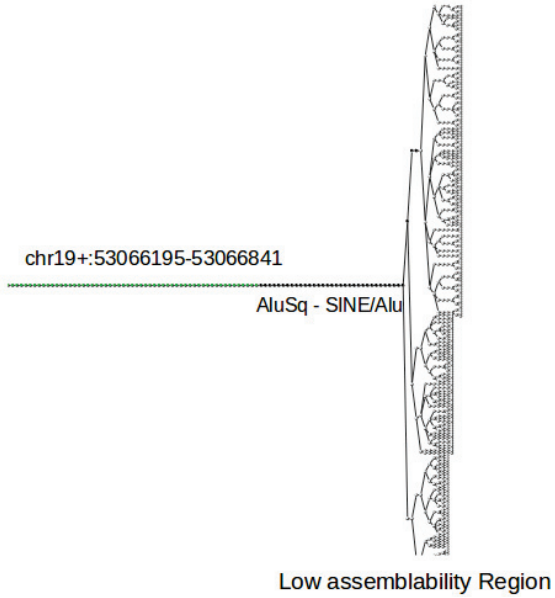


Figure 2.14: A real scenario in a human dataset where a transcript was truncated due to a short interspersed repeat, an Alu, which is present over one million times in the human genome. This small neighbourhood view around the transcript already shows how complex it is to make a decision. Transcriptome assemblers will usually employ techniques to remove paths that are unlikely to be correct, but the decision is still far from being trivial and error-prone. Thus, some assemblers decide to truncate the transcript.

family. Therefore, we can say that alternative splicing, paralogy and genic repeats causes issues and complicates the transcriptome assembly task.

Some strategies have been employed by the vast number of assemblers developed in the last years to try to cope with the short read/long repeat issue. One of the most well known is Myers' A-statistics [87,89]. It uses the coverage of a sequence to discriminate contigs that correspond to repeats, as the coverage is related to the copy-number of the repeat in the genome. However, in the RNA context, the coverage of a gene reflects mostly its expression level. RNA-seq specificities complicate the application of a genomic repeat-solving strategy to the transcriptomic context. Although some strategies managed to be successful in some scenarios, this issue can only be reliably solved with the read information. As such, many complex and repeat-dense regions a genome are being correctly assembled only in the recent years, with the advent of 3GS, which is able to span far longer repeats than 2GS. In the case of transcriptome assembly, 3GS reads are usually able to sequence full-length transcripts directly, thus eliminating the assembly step altogether. Such long reads are able to completely describe the isoforms' structure, revealing previously unknown distant exon couplings.

De novo assembly using de Bruijn Graphs

The two main approaches for *de novo* assembly are based on the Overlap-Layout-Consensus strategy and on de Bruijn graphs. In short, the **Overlap-Layout-Consensus (OLC)** strategy, as the name suggests, is composed by three phases. The first phase, overlap, computes all the overlaps between all pair of reads in order to find significant prefix-suffix overlaps (which can be inexact). In this phase, some optimizations can take place in order to avoid computing fruitless overlaps, and an overlap is considered significant if it exceeds a score or length threshold. The overlap graph $OG = (V, E)$ is then built, where V is the set of reads and $(r_1, r_2) \in E$ if there is a significant overlap between a suffix of r_1 and a prefix of r_2. As the overlap graph can be complex and tangled, the layout phase simplifies it, with the goal of linearizing it. The most common simplification in this step is the transitive reduction [86]. In the last phase, consensus, paths are enumerated which theoretically correspond to the original sequenced molecules. The OLC strategy is fit for sequencing technologies that produce longer but fewer reads, and is able to handle a high-rate of sequencing errors through inexact overlaps. Thus it was successfully applied to Sanger data, and now it is making a comeback due to the characteristics of 3GS reads [19]. The computational burden of the overlap phase restricted a bit its use on 2GS reads, due to the fact that such reads are short and massively produced, but it was still appropriate for processing 454 data. As such, the OLC strategy has been applied to several Sanger and 2GS assemblers, *e.g.* CAP3 [46], Celera [87], Edena [44], Newbler [83], etc. More recently, it has also been applied to 3GS reads, *e.g.* Canu [12], Miniasm [70], LQS[3] [78], etc. However, as this thesis does not focus on OLC strategies, we will not develop further on this approach.

The majority of the works in this thesis is based on de Bruijn graphs. Here, we

[3]This assembly pipeline does not have a proper name, LQS are the initials of its authors' names.

describe this data structure. Given a string s, a k-mer is a substring of s of length k. Given an integer k and a set S of strings each of length $n \geq k$, we define $span(S,k)$ as the set of all distinct k-mers in S. The (directed) **de Bruijn graph (DBG)** of a set S of strings with order k is the graph $G_k(S) = (V, A)$ where $V = span(S,k)$ and $(u,v) \in A$ if and only if $u[2,k] = v[1,k-1]$. Informally, DBGs are directed graphs that efficiently represent much of the information contained in a set of sequences. Vertices represent all the unique k-mers, and arcs represent (k-1)-exact-overlaps between k-mers. See Figure 2.15A-B for an example of a DBG.

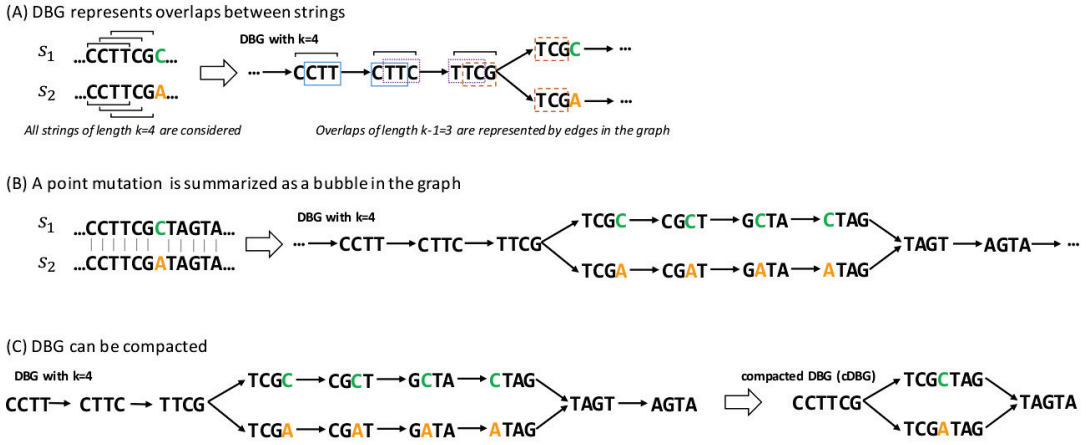


Figure 2.15: Compacted DBG construction over a set of sequences differing by a single point mutation. In this example, two sequences s_1 and s_2 of length 12 differ by a single letter. (A) All k-mers ($k = 4$) present in these sequences are listed. A link is drawn between two k-mers when the $k-1 = 3$ last nucleotides of the first k-mer equal the 3 first nucleotides of the second k-mer; (B) The bubble pattern represents the SNP C to A; each branch of the bubble represents an allele; (C) Linear paths of the graph are compacted; the compacted DBG of the example only contains four vertices (unitigs) and represents the same variation as the original DBG, which contained 13 vertices (k-mers). Figure reproduced from [48].

The **abundance** of a vertex $v \in G_k(S)$, denoted by $a(v)$, is the number of times its associated k-mer appears in S. The **relative out-abundance (in-abundance)** of an arc $e = (s,t) \in G_k(S)$ is $ra^+(e) = a(t)/\sum_{v \in N^+(s)} a(v)$ ($ra^-(e) = a(s)/\sum_{v \in N^-(t)} a(v)$). An arc $(u,v) \in A$ is called **compressible** if $d^+(u) = 1$ and $d^-(v) = 1$. The intuition behind this definition comes from the fact that every path passing through u should also pass through v. It should therefore be possible to "compress"' or contract this arc without losing any information. A **compressed de Bruijn graph (cDBG)** (some authors also use the term **compacted de Bruijn graph**) can be obtained from a de Bruijn graph by replacing, for each compressible arc (u,v), the vertices u, v by a new vertex x, where $N^-(x) = N^-(u)$, $N^+(x) = N^+(v)$ and the label is the concatenation of the k-mer of u and the k-mer of v without the overlapping part (see Figure 2.16). This

process is repeated until no compressible arc remains. An alternative similar approach to create cDBGs is by merging linear paths (sequences of vertices not linked to more than two other vertices) of the DBG into a single vertex, with the label of this vertex being the sequence spelled by the linear path. The vertices of cDBGs are called **unitigs**. Figure 2.15C shows an example of constructing a cDBG.

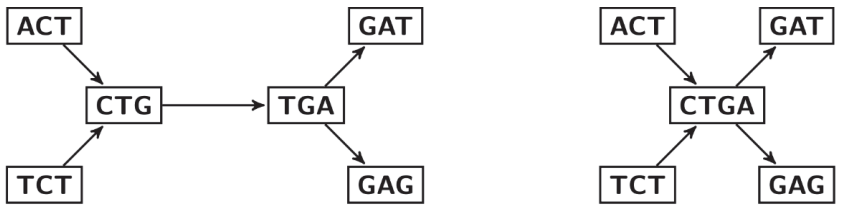


Figure 2.16: Example of compressible arc in a DBG. (a) The arc (CTG, TGA) is the only compressible arc in the given DBG ($k = 3$). (b) The result of compressing the arc. Figure reproduced from [77].

DBGs were extensively used in processing Illumina reads, due to some reasons. The first one is efficiency. Usually, the first step in building a DBG is breaking reads into k-mers – it is far more efficient computationally to work on the set of unique k-mers than on the set of Illumina reads. The information lost due to breaking reads into k-mers is not a severe issue when working with Illumina data, since such reads are shorter than Sanger and 3GS reads, and not so longer than the k-mer size. Further, DBGs can be efficiently implemented through data structures that allow fast (amortized) insert and query operations, as it can be viewed as a set of elements (k-mers). As such, DBGs can be efficiently implemented through specialized structures, such as hash tables [23], bloom filters [21, 30], FM-indexes [10, 81], etc. Therefore, DBG construction avoids altogether the expensive all-pairs read overlap of the OLC approach, being more suitable to the massive Illumina datasets. A second reason lies on the fact that Illumina reads have low error-rate and very high throughput. Thus, the probability of having error-free parts of short reads covering a region is very high, and therefore the issue that DBGs do not model inexact overlaps is minimized. Further, in some applications, like searching for small genomic variations, such as SNPs and small indels, explicitly representing each base from the input reads can be an advantage. We should expand, however, on one of the biggest flaws of modeling sequences through DBGs: the fact that DBGs lose the information that a set of k-mers came from the same read, which can be valuable. As an example, it might not be possible to correctly assemble two different transcripts that share a common substring s with $|s| = k$, if the read information is lost. However, as the read length is normally larger than k, the resolution of such short repeats is possible using the read information. When the length of the reads becomes larger, and the read information becomes more valuable, then the use of the classical DBG is arguable, with approaches utilizing the full read length, like OLC, maybe performing better.

DBG-based assemblers usually present a common set of steps. The first is, as ex-

pected, building the DBG from the set of raw reads. The second is graph simplification, where bubbles, tips (short dead-ends), arcs and vertices likely corresponding to sequencing errors are removed and the graph becomes smaller and more linear. In general, global assemblers are usually not interested in small genomic variations, and thus remove bubbles induced by true SNPs/indels. Assemblers can differentiate sequencing error artifacts from true variations using coverage. In a diploid genome, for example, both paths of a bubble due to a SNP are expected to have similar coverage, while in a sequencing-error-induced bubble, the erroneous base would spell a very low coverage path. Sequencing errors removal is trickier in the RNA context due to: i) errors in highly expressed genes generating k-mers sometimes more frequent than correct k-mers in non-highly expressed genes; ii) SNPs in poorly expressed genes may be mistaken for sequencing errors due to the low number of reads supporting each allele; iii) transcripts exhibit a 5' to 3' heterogeneity of coverage due to technical sequencing reasons. Genome assemblers are usually more aggressive than transcriptome assemblers in this step, since their goal is to assemble the reads into few sequences, and they can rely on an almost uniform coverage across the genome. Transcriptome assemblers cannot afford to be so aggressive, since they risk removing true variations due to alternative splicing and transcription. The linear paths obtained after such simplifications are called **contigs**. Transcriptome assemblers normally include an additional step responsible for partitioning the graph into gene components. Finally, the last step involves finding the most likely set of paths that describes the original chromosomes or isoforms.

Local assembly of alternative splicing events

This subsection is heavily based on [107].

We have explored in the previous subsections how hard and error-prone full-length transcriptome assembly from short reads can be. Although current full-length transcriptome assemblers do a great job, many transcripts remain hard to be fully assembled, mainly due to the short length of 2GS reads. This could impact downstream analyses, which are biased towards well-assembled isoforms. Further, assemblers apply heuristics to produce longer sequences, such as tip and bubble removal, which could result in a loss of information that could be relevant to study variations in transcriptomic data. Therefore, it is not always necessary or desirable to aim at the difficult goal of assembling full-length molecules. For the study of alternative splicing, for example, assembling only the variable parts between the isoforms can be already very valuable.

If we represent the raw reads data through DBGs, the variable parts between two isoforms due to alternative splicing will be flanked by invariable parts, thus composing a bubble in the DBG. In general, any pattern asb and $as'b$ in the input sequences, with a, b, s, $s' \in \Sigma^*$, $|a| \geq k, |b| \geq k$, and s and s' not sharing any k-mer, creates a bubble in the DBG. We note that i) transcriptional events do not compose bubbles but forks; ii) other types of events, besides alternative splicing, can compose bubbles, *e.g.* genomic SNPs or indels, recombinations, repeats, etc. Figure 2.17 shows two transcripts with alternative start and termination sites (transcriptional variations) and with two alternative splicing events, and the structures these variations induce in a DBG.

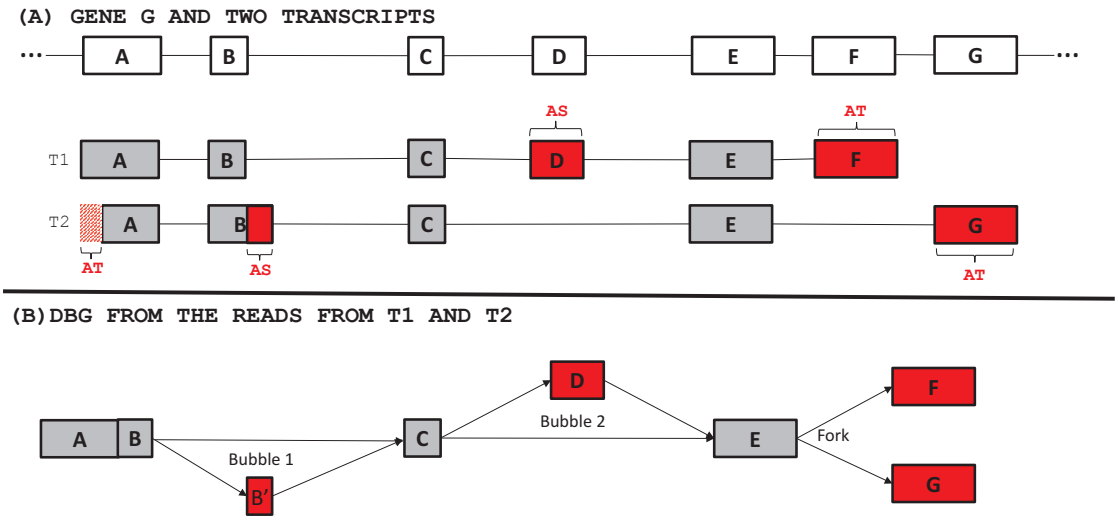


Figure 2.17: Two transcripts $T1$ and $T2$ from a same gene G with transcriptional and alternative splicing variations, and the structures these variations induce in a DBG. (A) The gene G and the two transcripts $T1$ and $T2$. Variable parts are highlighted in red, constitutive (invariable) parts are shown in grey. AT stands for Alternative Transcription event, and AS for Alternative Splicing event. (B) The DBG from $T1$ and $T2$. For simplification purposes, we are not representing junction k-mers (*i.e.* k-mers having bases stemming from two consecutive exons, *e.g.* in the arc from exon B to C in Bubble 1, in reality, we have k-mers that make the junction of exons B and C). This specific alternative transcription start site does not induce any structure in the DBG, but the alternative splicing events induce bubbles that are flanked by constitutive parts (grey regions) and composed by variable parts (red regions). Alternative transcriptions can, however, induce fork structures, like the alternative transcription end depicted here. Bubble 1 represents an alternative 3' splice site of exon B, and Bubble 2 represents the skipping of exon D.

KisSplice [107] is a method for assembling alternative splicing events through the enumeration of bubbles in DBGs built from short RNA-sequencing reads. It is composed of seven main steps:

1. **DBG construction.** Construction of the cDBG from short RNA-sequencing reads. To deal with sequencing errors, k-mers present less than c (default 2) times in the read set are removed;

2. **Relative error removal.** The absolute threshold applied in the previous step normally does not work on highly expressed genes, where an error can be present more than c times due to deep coverage. In this step, a relative error removal procedure is applied, by removing the arcs $e \in G_S | ra^+(e) < ra_{min}$ or $ra^-(e) < ra_{min}$, where ra_{min} is the minimum relative abundance threshold (defaults to 0.05);

3. **Biconnected component (BCC) decomposition.** BCCs of an undirected graph form a partition of the edges with two important properties: every cycle is contained in exactly one BCC, and every edge not contained in a cycle forms a singleton BCC. All singleton BCCs are then discarded, since they cannot contain any bubble. This step reduced a lot the memory footprint and the computation time of the pipeline;

4. **Four-vertices compression.** Single substitution events (SNPs, sequencing errors) generate a large number of cycles themselves included into bigger ones, creating a combinatorial explosion of the number of possible bubbles. This step compresses bubbles such that both of its paths are composed by only one non-branching vertex, and the sequences of these vertices differ by only one mismatch. It does not compress all the SNPs, but a part of them;

5. **Bubble enumeration.** Bubbles are listed in this step. This step is critical for performance and has been improved in several works. The first algorithm used a backtracking procedure augmented with two pruning criteria [107]. Birmelé *et al.* in [14] describe the first linear delay algorithm to enumerate all bubbles with a given source. However, these two algorithms have two big issues: (i) the bubble space is huge in real data, so exploring a big part of it is costly; (ii) not all bubbles are interesting – some constraints can be applied to filter out the majority of non-interesting bubbles. Clearly, such filters can always be applied to post-process the output of a bubble enumeration algorithm, but this does not translate into faster running times. Further, observing that AS events are usually not so long, in [108] the authors proposed the first polynomial-delay algorithm to list bubbles with maximum length constraints in a weighted directed graph. Formally, the algorithm described in [108] enumerates $(s, t, \alpha_1, \alpha_2)$-bubbles, *i.e.* bubbles with source s and target t such that the paths p_1 and p_2 of the bubbles satisfy $|p_1| \leq \alpha_1$ and $|p_2| \leq \alpha_2$. Some AS events are systematically missed by this algorithm, such as intron retention events (can be several kbs long), and multi exon splicing events (can also be on the order of kbs), but in many species these are not the common AS events. Even so, there were still certain complex regions in the graph, likely containing repeat-associated subgraphs, but also real AS events, where the most improved algorithm would still take a huge amount of time. In practice, the enumeration is halted after a given timeout and the bubbles trapped inside these regions were thus missed. A study on the subgraphs induced by repeats showed that enumerating all bubbles with at most b branching vertices ($b = 5$ by default) in each path allowed the algorithm to implicitly avoid repeat-containing regions during the assembly, and thus be able to enumerate even bubbles trapped inside repeat-induced subgraphs (if their paths do not traverse repeats). Of course, all AS events containing repeats (*e.g.* exonised exons and many intron retentions) are missed due to this filtering, but the advantage is that the non-repeat-containing bubbles are efficiently and exhaustively enumerated, enabling a better understanding of the AS events in datasets of many species. Chapter 3 explores in detail this algorithm;

6. **Results filtration and classification.** By comparing and aligning both paths of a bubble, the latter can be classified into putative: (i) SNPs; (ii) AS events; (iii) small indels; (iv) repeats; (v) undefined.

7. **Read coherence and coverage computation.** Reads from each input dataset are mapped to each path of the bubble. If at least one nucleotide of a path is covered by no read, the bubble is said to be not read-coherent and is discarded. The coverage of each position of the bubble corresponds to the number of reads overlapping this position.

KisSplice enables to tackle the problem of finding AS events without a reference genome and without assembling full-length transcripts, which may be time consuming and uses heuristics that may lead to a loss of information. It was the first software to tackle the problem of exploring variations in RNA-sequencing data through a local assembly perspective. It not only constitutes an important software to study AS events, but also a useful complement to general purpose transcriptome assemblers.

Other relevant processing tasks of 2GS and 3GS data

Differential expression analyses. Differential expression analyses always come associated to a biological question. In most cases, this question is inferring which genes or transcripts are differentially expressed between two or more pathological or physiological conditions. This would allow one to know the genes or transcripts that are inhibited in one condition and activated in another condition, or vice-versa, giving clues that such features are related to the studied condition. To find differentially expressed genes or transcripts, we first need to know their expression levels. We usually do not have access to such information, but it can be estimated by two main approaches. The first maps the reads to a reference genome or transcriptome, and post-processes the mapping output to estimate the quantification, using tools such as Cufflinks/Cuffquant [126], eXpress [104], RSEM [66], featureCounts [73], etc. The mapping process is a rather computationally expensive task. As such, other approaches avoid this step, using alignment-free or pseudo-alignment algorithms, such as Sailfish [95], Kallisto [17], and Salmon [94]. Finally, assessing if a gene or transcript is differentially expressed boils down to verifying if their estimated expression levels significantly changes across conditions. This can be tested using a variety of methods such as DESEQ2 [80], Cufflinks/Cuffdiff [125], NOISeq [120], etc.

It is also possible to be more fine-grained in differential expression analysis by verifying if there are differential AS events between two or more experimental conditions. This could be more appropriate than gene or transcript differential analyses to answer some biological questions, like inferring if the spliceosome acts differently in the studied conditions, or more generally if there is a differential exon usage across conditions. In some cases, where only the variable regions between isoforms are of interest, a differential AS events analysis can be more appropriate, as it could be easier to assemble and more reliable to quantify such events than full-length transcripts, especially in a *de novo* context with Illumina reads. The methodological principle is the same: first the AS events are

quantified, and then a statistical method retrieves those that are differentially expressed. Differential AS events can be found by tools such as KisSplice [107] (quantification) + kissDE [11, 79] (statistical method), Leafcutter [72], DEXSeq [4], etc.

Error correction. Error correction is the task of correcting sequencing errors from reads. It is not a crucial task for 2GS reads, as the sequencing error rate of common Illumina datasets is around 0.1%, but it can be seen as essential for some applications with 3GS data, where the error rate ranges from 10 to 15%. There exist two types of 3GS error-correction algorithms, those using information from long reads only (**self or non-hybrid correction**), and those using short reads to correct long reads (**hybrid correction**). The output of such correction approaches can be classified into three types: full-length, trimmed and split. Usually, due to methodological reasons, the ends of long reads are harder to correct. As an example, hybrid corrections based on mapping short to long reads and calling the consensus from the mapping have difficulties aligning short reads to the ends of long reads. As such, some methods output **trimmed error-corrected reads**, *i.e.* error-corrected reads such that their uncorrected ends were removed. Examples of methods producing this type of output are HALC [9], LoRDEC [110], LSC [6], proovread [43], daccord [123], and pbdagcon [22]. Sometimes, internal parts of long reads can also be hard to correct, due to a lack of coverage of short reads, or due to a very high variation rate, for example. Some algorithms thus output **split error-corrected reads**, splitting one long read into several well-corrected fragments, such as HALC [9], LoRDEC [110], PBcR [58], and LoRMA [111]. Finally, some tools decide to not trim or split the original reads, outputting full-length error-corrected reads. Examples include HALC [9], LoRDEC [110], LSC [6], proovread [43], canu [59], daccord [123], MECAT [131], and pbdagcon [22]. As can be noted, some tools produce more than one type of output, sometimes all the three types.

Chapter 3

Playing hide and seek with repeats in local and global de novo transcriptome assembly of short RNA-seq reads

Preamble

Key points

- In opposition to the general consensus, repeats are an underestimated problem in *de novo* transcriptome assembly, creating ambiguities and confusing assemblers. Their presence are mainly due to the 5-15% of pre-mRNA fraction in common RNA extraction protocols;

- We introduce a simple, but realistic enough, formal model for representing high copy-number and low-divergence repeats in RNA-seq data;

- In the specific case of local assembly of alternative splicing events, we can avoid processing repeats. This led us to lose all events containing repeats, but also to significantly increase the sensitivity and the precision, outperforming previous versions of KiSplice [107, 108], Trinity [38], and Oases [114];

- We show a proof of concept that exploring the topology of the subgraph around a transcript can give some hints about its confidence level, quality, assembly hardness, etc. This information can be as valuable as read and coverage information for transcriptome assemblers and evaluators;

- Transcriptome assemblers can be improved by explicitly and formally modeling repeats.

Status

Published in journal *Algorithms for Molecular Biology* [77].

Author contributions

The first authorship is shared between *L.* and *B. Sinaimeri*. This paper is a journal extension of the conference paper [109]. The conference paper focused only on local transcriptome assembly. Here, we turn our focus also to global transcriptome assembly.

The main improvement regarding [109] is that we introduce a measure that can be viewed as a proof of concept that exploring the topology of the subgraph around a transcript can give some hints about its confidence level, quality, assembly hardness, etc. More specifically, we explore a very simple characteristic of the topology of DBGs, based on the observation that repeats create complicated regions, and we show that it is able to flag chimeric and very fragmented transcripts built by Trinity [38], a state-of-the-art transcriptome assembler, on real data. Furthermore, we compared the performance of our simple measure with state-of-the-art transcriptome evaluation methods, Rsem-Eval [67] and TransRate [117], on identifying assembled chimeric transcripts on two simulated and two real datasets. Our simple measure outperforms both methods by a large margin in all tests. Rsem-Eval and TransRate failed to flag chimeric transcripts stemming from genes with similar expression levels, hinting that they rely heavily on coverage information. We therefore show that the topology of assembly graphs should be taken into consideration by both assemblers and evaluators.

Lima *et al. Algorithms Mol Biol (2017) 12:2*
DOI 10.1186/s13015-017-0091-2

Algorithms for Molecular Biology

RESEARCH Open Access

Playing hide and seek with repeats in local and global de novo transcriptome assembly of short RNA-seq reads

Leandro Lima[1,2*†], Blerina Sinaimeri[1,2†], Gustavo Sacomoto[1,2], Helene Lopez-Maestre[1,2], Camille Marchet[3], Vincent Miele[2], Marie-France Sagot[1,2] and Vincent Lacroix[1,2]

Abstract

Background: The main challenge in de novo genome assembly of DNA-seq data is certainly to deal with repeats that are longer than the reads. In de novo transcriptome assembly of RNA-seq reads, on the other hand, this problem has been underestimated so far. Even though we have fewer and shorter repeated sequences in transcriptomics, they do create ambiguities and confuse assemblers if not addressed properly. Most transcriptome assemblers of short reads are based on de Bruijn graphs (DBG) and have no clear and explicit model for repeats in RNA-seq data, relying instead on heuristics to deal with them.

Results: The results of this work are threefold. First, we introduce a formal model for representing high copy-number and low-divergence repeats in RNA-seq data and exploit its properties to infer a combinatorial characteristic of repeat-associated subgraphs. We show that the problem of identifying such subgraphs in a DBG is NP-complete. Second, we show that in the specific case of local assembly of alternative splicing (AS) events, we can *implicitly* avoid such subgraphs, and we present an efficient algorithm to enumerate AS events that are not included in repeats. Using simulated data, we show that this strategy is significantly more sensitive and precise than the previous version of KisSplice (Sacomoto et al. in WABI, pp 99–111, 1), Trinity (Grabherr et al. in Nat Biotechnol 29(7):644–652, 2), and Oases (Schulz et al. in Bioinformatics 28(8):1086–1092, 3), for the specific task of calling AS events. Third, we turn our focus to full-length transcriptome assembly, and we show that exploring the topology of DBGs can improve de novo transcriptome evaluation methods. Based on the observation that repeats create complicated regions in a DBG, and when assemblers try to traverse these regions, they can infer erroneous transcripts, we propose a measure to flag transcripts traversing such troublesome regions, thereby giving a confidence level for each transcript. The originality of our work when compared to other transcriptome evaluation methods is that we use only the topology of the DBG, and not read nor coverage information. We show that our simple method gives better results than Rsem-Eval (Li et al. in Genome Biol 15(12):553, 4) and TransRate (Smith-Unna et al. in Genome Res 26(8):1134–1144, 5) on both real and simulated datasets for detecting chimeras, and therefore is able to capture assembly errors missed by these methods.

Keywords: Transcriptome assembly, RNA-seq, Repeats, Alternative splicing, Formal model for representing repeats, Enumeration algorithm, De Bruijn graph topology, Assembly evaluation

*Correspondence: leandro.ishi.lima@gmail.com
†Leandro Lima and Blerina Sinaimeri contributed equally to this work
[2] CNRS, UMR5558, Université Claude Bernard Lyon 1, 43, Boulevard du 11 Novembre 1918, 69622 Villeurbanne, France
Full list of author information is available at the end of the article

Background

Transcriptomes can now be studied through sequencing. However, in the absence of a reference genome, de novo assembly remains a challenging task. The main difficulty certainly comes from the fact that sequencing reads are short, and repeated sequences within transcriptomes could be longer than the reads. This short read/long repeat issue is of course not specific to transcriptome sequencing. It is an old problem that has been around since the first algorithms for genome assembly. Even though the problems repeats cause in both contexts are similar, they have also some characteristics that are specific to each. In genome assembly, repeats tend to be longer and present in more copies. In transcriptome assembly, repeats are located within genes and tend to be shorter and in fewer copies. However, in this last case, coverage cannot be applied to discriminate contigs that correspond to repeats, as it can be in genomics by using e.g. Myers' A-statistics [6, 7], since the coverage of a gene does not only reflect its copy-number in the genome, but also and mostly its expression level. Some genes are highly expressed and therefore highly covered, while most genes are poorly expressed and therefore poorly covered. Such specificities complicate the application of a genomic repeat-solving strategy to the transcriptomic context.

Initially, it was thought that repeats would not be a major issue in RNA-seq, since they are mostly in introns and intergenic regions. However, the truth is that many regions which are thought to be intergenic are transcribed [8] and introns are not always already spliced out when mRNA is collected to be sequenced [9]. Repeats, especially transposable elements, are therefore very present in real samples and cause major problems in transcriptome assembly, if not addressed properly.

Most, if not all current short-read transcriptome assemblers are based on de Bruijn graphs. Among the best known are Oases [3], Trinity [2], and to a lesser degree Trans-Abyss [10] and IDBA-tran [11]. Common to all of them is the lack of a clear and explicit model for repeats in RNA-seq data. Heuristics are thus used to try and cope efficiently with repeats. For instance, in Oases short vertices are thought to correspond to repeats and are therefore not used for assembling genes. They are added in a second step, which hopefully causes genes sharing repeats not to be assembled together. In Trinity, there is no attempt to deal with repeats by explicitly modelling them. The first module of Trinity, Inchworm, will try and assemble the most covered contig which hopefully corresponds to the most abundant alternative transcript. Then alternative exons are glued to this major transcript to form a splicing graph. The last step is to enumerate all alternative transcripts. If repeats are present, their high coverage may be interpreted as a highly expressed link between two unrelated transcripts. Overall, assembled transcripts may be chimeric or spliced into many sub-transcripts.

In the method we had previously developed, KisSplice, which is a local transcriptome assembler [12], repeats are less problematic since the goal is not to assemble full-length transcripts. KisSplice instead aims at finding variants in transcriptomes (SNPs, indels and alternative splicings). However, as we reported in [12], KisSplice was not able to deal with large portions of a de Bruijn graph containing subgraphs associated to highly repeated sequences, e.g. transposable elements, the so-called complex Biconnected Components.

Here, we try and achieve three goals: (1) give a clear formalisation of the notion of repeats with high copy-number in RNA-seq data, (2) apply it on local transcriptome assembly by giving a practical way to enumerate bubbles that are lost because of such repeats, and (3) apply it on global transcriptome assembly by showing that the topology of the subgraph around a transcript can give some hints about its confidence level. Recall that we are in a de novo context, so we assume that neither a reference genome/transcriptome nor a database of known repeats, e.g. RepBase [13], are available.

First, we formally introduce a model for representing high copy-number repeats and exploit its properties to infer that repeat-associated subgraphs in a de Bruijn graph contain few compressible arcs. However, we show that the problem of identifying, in a de Bruijn graph, a subgraph corresponding to repeats according to such characterisation is NP-complete. A polynomial time algorithm is therefore unlikely to exist.

Second, we show that in the specific case of a local assembly of alternative splicing (AS) events, by using a strategy based on the compressible-arc characterization, we can *implicitly* avoid such subgraphs. More precisely, it is possible to find the structures (i.e. bubbles) corresponding to AS events in a de Bruijn graph that are not contained in a repeat-associated subgraph (see Fig. 3 for an example). While there has been great efforts in the literature to solve repeats, there has been almost no exploration on how to avoid them. This is explained by the fact that most efforts in assembly concentrate on full-length genome and transcriptome assembly, in which avoiding repeats is not an option, and the performance of an assembler can be narrowed down to how well it solves repeats. However, in our case, repeat-avoidance can be an effective technique. Indeed, this fact was confirmed by our experiments, where using human simulated RNA-seq data, we show that the new algorithm improves significantly the sensitivity of KisSplice, while also improving its precision. We further compared our algorithm to

two of the best transcriptome assemblers, namely Trinity [2] and Oases [3], in the specific task of calling AS events, and we show that our algorithm is more sensitive than both tools, while also being more precise. In addition, our results show that the advantage of using the new algorithm proposed in this work is more evident when the input data contains high pre-mRNA content or the AS events of interest stem from highly-expressed genes. Moreover, we give an indication of the usefulness of our method on real data.

Third, we show that the method described can also be applied in the context of full-length transcriptome assembly. We introduce a measure based on the proposed model to identify low-confidence transcripts, which are the ones that traverse complex regions in the de Bruijn Graph. Within these complex parts of the graph generated by repeats, any assembler will have to choose the "right" path(s) among the many present. This choice is not simple and may lead to incorrect solutions (e.g. chimeric or truncated transcripts). It is therefore important to be able to identify the transcripts coming from such complex regions in order to know that the solution presented is not the only one, and furthermore may not be the right one. We compared our measure against two state-of-the-art methods for de novo transcriptome evaluation, namely Rsem-Eval [4] and TransRate [5], for the specific task of identifying chimeric transcripts in both real and simulated datasets. We show that our measure provides good results despite the fact that it uses only the graph topology, and not coverage, nor read information. The results obtained thus suggest that exploring the topology of the subgraph around a transcript, an information that is currently disregarded by transcriptome evaluation methods, can be useful to infer some of the transcript's properties, such as confidence level, quality, assembly hardness, etc. Therefore, our measure can improve the state-of-the-art methods for de novo transcriptome evaluation, since it is able to capture assembly errors missed by these tools.

Preliminaries

Let Σ be an alphabet of fixed size σ. Here we always assume $\Sigma = \{A, C, T, G\}$. Given a sequence (string) $s \in \Sigma^*$, let $|s|$ denote its length, $s[i]$ the ith element of s, and $s[i, j]$ the substring $s[i]s[i+1]\dots s[j]$ for any $1 \leq i < j \leq |s|$.

A *k-mer* is a sequence $s \in \Sigma^k$. Given an integer k and a set S of sequences each of length $n \geq k$, we define $span(S, k)$ as the set of all distinct k-mers that appear as a substring in S.

Definition 1 Given a set of sequences (reads) $R \subseteq \Sigma^*$ and an integer k, we define the directed de Bruijn graph $G_k(R) = (V, A)$ where $V = span(R, k)$ and $(u, v) \in A$ if and only if $u[2, k] = v[1, k-1]$.

Given a directed graph $G = (V, A)$ and a vertex $v \in V$, we denote its *out-neighbourhood* (resp. *in-neighbourhood*) by $N^+(v) = \{u \in V \mid (v, u) \in A\}$ (resp. $N^-(v) = \{u \in V \mid (u, v) \in A\}$), and its *out-degree* (resp. *in-degree* by $d^+(v) = |N^+(v)|$ ($d^-(v) = |N^-(v)|$). A (simple) *path* $\pi = s \rightsquigarrow t$ in G is a sequence of distinct vertices $s = v_0, \dots, v_l = t$ such that, for each $0 \leq i < l$, (v_i, v_{i+1}) is an arc of G. If the graph is weighted, i.e. there is a function $w : A \to Q_{\geq 0}$ associating a weight to every arc in the graph, then the *length* of a path π is the sum of the weights of the traversed arcs, and is denoted by $|\pi|$.

An arc $(u, v) \in A$ is called *compressible* if $d^+(u) = 1$ and $d^-(v) = 1$. The intuition behind this definition comes from the fact that every path passing through u should also pass through v. It should therefore be possible to "compress" or contract this arc without losing any information. Note that the compressed de Bruijn graph [2, 3] commonly used by transcriptomic assemblers is obtained from a de Bruijn graph by replacing, for each compressible arc (u, v), the vertices u, v by a new vertex x, where $N^-(x) = N^-(u)$, $N^+(x) = N^+(v)$ and the label is the concatenation of the k-mer of u and the k-mer of v without the overlapping part (see Fig. 1).

Repeats in de Bruijn graphs

Given a de Bruijn graph $G_k(R)$ generated by a set of reads R for which we do not have any prior information, our goal is to identify whether there are subgraphs of $G_k(R)$ that correspond each to a set of high copy-number repeats in R. To this end, we identify and then exploit some of the topological properties of the subgraphs that are induced by repeats. Starting with a formal model for representing repeats with high-copy number, we show that the number of compressible arcs, which we denote by γ, is a relevant parameter for such a characterisation. This parameter will play an important role in the algorithm of "Bubbles "drowned" in repeats" section.

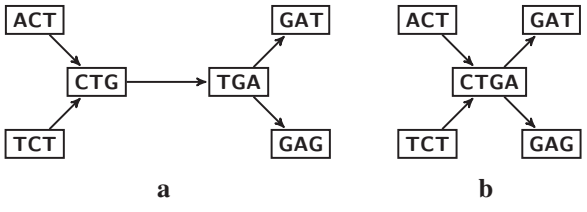


Fig. 1 Example of compressible arc in a de Bruijn graph. **a** The arc (*CTG*, *TGA*) is the only compressible arc in the given de Bruijn graph ($k = 3$). **b** The corresponding compressed de Bruijn graph

However, we also prove that, for an arbitrary de Bruijn graph, identifying a subgraph G' with bounded $\gamma(G')$ is NP-complete.

Simple uniform model for repeats

We now present the model we adopted for representing high copy-number repeats, e.g. transposable elements, in a genome or transcriptome. First, we would like to clarify that our model is a simple one and, as such, should be seen as only a first approximation, yet realistic enough, of what may happen in reality. We consider here that sequencing errors can be successfully removed. Indeed, there are several techniques to remove the big majority of the sequencing errors in RNA-seq data. In KisSplice, for example, we prune the de Bruijn graph using an absolute and a relative cut-off based on the k-mer coverage. The absolute cut-off enables us to remove sequencing errors in general, and the relative one is tailored to deal with highly-expressed genes (more details can be found in [14]). Furthermore, while we realise that there is room for improvement, in practice, the sequencing-error-removal procedure in KisSplice seems to be effective, as most sequencing errors are removed at the expense of losing some rare genomic variants [14].

Basically, our model consists of several "similar" sequences, each generated by uniformly mutating a fixed initial sequence. In particular, it enables to model well recent invasions of transposable elements which often involve high copy-number and low divergence rate (i.e. divergence from their consensus sequence). Consider indeed as an example the recent subfamilies AluYa5 and AluYb8 with 2640 and 1852 copies respectively, which both present a divergence rate below 1% [15] (see [16] for other subfamilies with high copy-number and low divergence).

The model is as follows. First, due to mutations, the sequences $s_1, \ldots, s_m$ that represent the repeats are not identical. However, provided that the number of such mutations is not high (otherwise the concept of repeats would not apply), the repeats are considered "similar" in the sense of having a small pairwise Hamming distance between them. We recall that, given two equal length sequences s and s' in Σ^n, their *Hamming distance*, denoted by $d_H(s, s')$, is the number of positions i for which $s[i] \neq s'[i]$. Indels are thus not considered in this model.

The model has then the following parameters: Σ, the length n of the repeat, the number m of copies of the repeat, an integer k (for the length of the k-mers considered), and the mutation rate, α, i.e. the probability that a mutation happens in a particular position. The sequences $s_1, \ldots, s_m$ are then generated by the following process. We first choose uniformly at random a sequence $s_0 \in \Sigma^n$. At step $i \leq m$, we create a sequence s_i as follows: for each position j, $s_i[j] = s_0[j]$ with probability $1 - \alpha$, whereas with probability α a value different from $s[j]$ is chosen uniformly at random for $s_i[j]$. We repeat the whole process m times and thus create a set $S(m, n, \alpha)$ of m such sequences from s_0 (see Fig. 2 for a small example). The generated sequences thus have an expected Hamming distance of αn from s_0.

Topological characterisation of the subgraphs generated by repeats

Given a de Bruijn graph $G_k(R)$, if a is a compressible arc labelled by the sequence $s = s_1 \ldots s_{k+1}$ then, by definition, a is the only outgoing arc of the vertex labelled by the sequence $s[1, k]$ and the only incoming arc of the vertex labelled by the sequence $s[2, k+1]$. Hence the $(k-1)$-mer $s[2, k]$ appears as a substring in R, always preceded by the symbol $s[1]$ and followed by the symbol $s[k+1]$. We refer to such $(k-1)$-mers as being *boundary rigid*. It is not difficult to see that the set of compressible arcs in a de Bruijn graph $G_k(R)$ stands in a one-to-one correspondence with the set of boundary rigid $(k-1)$-mers in R.

We now calculate and compare among them the expected number of compressible arcs in $G = G_k(R)$ when R corresponds to a set of sequences that are generated: (1) uniformly at random, and (2) according to our model. We show that γ is "small" in the cases where the induced graph corresponds to similar sequences, which provides evidence for the relevance of this parameter.

Claim 1 *Let R be a set of m sequences randomly chosen from Σ^n. Then the expected number of compressible arcs in $G_k(R)$ is $\Theta(mn)$.*

Proof The probability that a sequence of length $k-1$ occurs in a fixed position in a randomly chosen sequence of length n is $(1/4)^{k-1}$. Thus the expected number of

$$
\begin{array}{c}
\begin{array}{cccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} \end{array} \\
\begin{pmatrix}
A & A & C & T & G & T & A & T & C & C \\
A & C & C & T & G & T & A & G & C & C \\
G & A & C & T & C & A & A & T & C & C \\
A & A & C & T & C & T & A & T & C & C \\
A & A & C & A & G & T & A & T & C & A \\
A & A & T & T & G & T & A & G & C & C \\
A & G & C & T & G & T & A & T & C & A \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
A & A & G & T & G & A & A & T & C & C
\end{pmatrix}
\begin{array}{c} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ \\ s_{20} \end{array}
\end{array}
$$

Fig. 2 An example of a set of repeats $S(20, 10, 0.1)$

appearances of a sequence of length $k-1$ in a set of m randomly chosen sequences of length n is given by $m(n-k+2)(1/4)^{k-1}$. If $m(n-k+2) \leq 4^{k-1}$, then this value is upper bounded by 1, and all the sequences of length $k-1$ are expected to be boundary rigid (as a sequence is expected to appear once). The claim follows by observing that there are $m(n-k+2)$ different $(k-1)$ -mers. □

We consider now $\gamma(G_k(R))$ for $R = S(m,n,\alpha)$. We upper bound the expected number of compressible arcs by upper bounding the number of boundary rigid $(k-1)$ -mers.

Theorem 1 *Given integers k, n, m with $k < n$ and a real number $0 \leq \alpha \leq 3/4$, the de Bruijn graph $G_k(S(m,n,\alpha))$ has $o(nm)$ expected compressible arcs.*

Proof Let s_0 be a sequence chosen randomly from Σ^n. Let $S(m,n,\alpha)$ be the set $\{s_1,\ldots,s_m\}$ of m repeats generated according to our model starting from s_0. Consider now the de Bruijn graph $G = G_k(S(m,n,\alpha))$. Recall that the number of compressible arcs in this graph is equal to the number of boundary rigid $(k-1)$-mers in $S(m,n,\alpha)$. Let X be a random variable representing the number of boundary rigid $(k-1)$-mers in G. Consider the repeats in $S(m,n,\alpha)$ in a matrix-like ordering as in Fig. 2 and observe that the mutations from one column to another are independent. Due to the symmetry and the linearity of the expectation, $E[X]$ is given by $m(n-k+2)$ (the total number of $(k-1)$-mers) multiplied by the probability that a given $(k-1)$-mer is boundary rigid.

The probability that the $(k-1)$-mer $\hat{s} = s[i,i+k-2]$ is boundary rigid clearly depends on the distance from the starting sequence $\hat{s}_0 = s_0[i,i+k-2]$. Let d be the distance $d_H(\hat{s},\hat{s}_0)$.

Observe that if the $(k-1)$-mer $s[i]\ldots s[i+k-2]$ is not boundary rigid then there exists a sequence y in $S(m,n,\alpha)$ such that $y[j] = s[j]$ for all $i \leq j \leq i+k-2$ and either $y[i+k-1] \neq s[i+k-1]$ or $y[i-1] \neq s[i-1]$. It is not difficult to see that the probability that this happens is lower bounded by $(2\alpha - 4/3\alpha^2)(1-\alpha)^{k-1-d}(\alpha/3)^d$. Hence we have:

$$Pr[\hat{s} \text{ is boundary rigid} | d_H(\hat{s},\hat{s}_0) = d] \leq \left(1 - (2\alpha - 4/3\alpha^2)(1-\alpha)^{k-1-d}(\alpha/3)^d\right)^{m-1}.$$

By approximating the above expression we therefore have that:

$$\begin{aligned} E[X] &\leq (n-k-1)m \sum_{d=0}^{k-1} Pr[\hat{s} \text{ is boundary rigid} | d_H(\hat{s},\hat{s}_0) = d] \\ &\leq (n-k-1)me^{-(m-1)(2\alpha-4/3\alpha^2)/(\frac{\alpha}{3})^{k-1}}. \end{aligned} \quad (1)$$

For a sufficiently large number of copies $\left(\text{e.g.} m = \binom{k}{\alpha k}\right)$ and using the fact that $\binom{k}{\alpha k} \geq (1/\alpha)^{\alpha k}$, we have that $E[X]$ is $o(mn)$. This concludes the proof. □

The previous result shows that the number of compressible arcs is a good parameter for characterising a repeat-associated subgraph.

Identifying a repeat-associated subgraph

As we showed, a subgraph due to repeated elements has a distinctive feature: it contains few compressible arcs. Based on this, a natural formulation to the repeat identification problem in RNA-seq data is to search for large enough subgraphs that do not contain many compressible arcs. This is formally stated in Problem 1. In order to disregard trivial solutions, it is necessary to require a large enough *connected* subgraph, otherwise any set of disconnected vertices or any small subgraph would be a solution. Unfortunately, we show that this problem is NP-complete, so an efficient algorithm for the repeat identification problem based on this formulation is unlikely.

Problem 1 [Repeat Subgraph] *INSTANCE:* A directed graph G and two positive integers m, t.

DECIDE: If there exists a connected subgraph $G' = (V',E')$, with $|V'| \geq m$ and having at most t compressible arcs.

In Theorem 2, we prove that this problem is NP-complete for all directed graphs with (total) degree, i.e. sum of in and out-degree bounded by 3. The reduction is from the Steiner tree problem which requires finding a minimum weight subgraph spanning a given subset of vertices. It remains NP-hard even when all arc weights are 1 or 2 (see [17]). This version of the problem is denoted by STEINER(1, 2). More formally, given a complete undirected graph $G = (V,E)$ with arc weights in $\{1,2\}$, a set of *terminal* vertices $N \subseteq V$ and an integer B, it is NP-complete to decide if there exists a subgraph of G spanning N with weight at most B, i.e. a connected subgraph of G containing all vertices of N.

We specify next a family of directed graphs that we use in the reduction. Given an integer x, we define the directed graph $R(x)$ as a cycle on $2x$ vertices numbered in a clockwise order and where the arcs have alternating

directions, i.e. for any $i \leq x$, (v_{2i}, v_{2i+1}) is an arc. Observe that in $R(x)$, all vertices in even positions, i.e. all vertices v_{2i}, have out-degree 2 and in-degree 0, while all vertices v_{2i+1} have out-degree 0 and in-degree 2. Clearly, none of the arcs of $R(x)$ is compressible.

Theorem 2 *The Repeat Subgraph Problem is NP-complete even for directed graphs with degree bounded by* d, for any $d \geq 3$.

Proof Given a complete graph $G = (V, E)$, a set of terminal vertices N and an upper bound B, i.e. an instance of STEINER(1, 2), we transform it into an instance of the *Repeat Subgraph Problem* for a graph G' with degree bounded by 3. Let us first build the graph $G' = (V', E')$. For each vertex v in $V \setminus N$, add a corresponding subgraph $r(v) = R(|V|)$ in G' and for each vertex v in N, add a corresponding subgraph $r(v) = R(|E| + |V|^2 + 1)$ in G'. For each arc (u, v) in E with weight $w \in \{1, 2\}$, add a simple directed path composed by w compressible arcs connecting $r(u)$ to $r(v)$ in G'; these are the subgraphs corresponding to u and v. The first vertex of the path should be in a sink of $r(u)$ and the last vertex in a source of $r(v)$. By construction, there are at least $|V|$ vertices with in-degree 2 and out-degree 0 (sink) and $|V|$ vertices with out-degree 2 and in-degree 0 (source) in both $r(v)$ and $r(u)$. It is clear that G' has degree bounded by 3. Moreover, the size of G' is polynomial in the size of G and it can be constructed in polynomial time.

In this way, the graph G' has one subgraph for each vertex of G and a path with one or two (depending on the weight of the corresponding arc) compressible arcs for each arc of G. Thus, there exists a subgraph spanning N in G with weight at most B if and only if there exists a subgraph in G' with at least $m = 2|N| + 2|E||N| + 2|V|^2|N|$ vertices and at most $t = |B|$ compressible arcs. This follows from the fact that any subgraph of G' with at least m vertices necessarily contains all the subgraphs $r(v)$, where $v \in N$, since the number of vertices in all $r(v)$, with $v \in V \setminus N$, is at most $|E| + 2|V|^2$ and the only compressible arcs of G' are in the paths corresponding to the arcs of G.□

We can obtain the same result for the specific case of subgraphs of de Bruijn graphs. The reduction is more technical but follows similarly.

Theorem 3 *The Repeat Subgraph Problem is NP-complete even for subgraphs of de Bruijn graphs on* $|\Sigma| = 4$ *symbols.*

Bubbles "drowned" in repeats

In the previous section, we showed that an efficient algorithm to *directly* identify the subgraphs of a de Bruijn graph corresponding to repeated elements according to our model (i.e. containing few compressible arcs), is unlikely to exist since the problem is NP-complete. However, in this section we show that in the specific case of a local assembly of alternative splicing (AS) events based on the compressible-arc characterisation of "Topological characterisation of the subgraphs generated by repeats" section, we can *implicitly* avoid such subgraphs. More precisely, it is possible to find the structures (i.e. bubbles) corresponding to AS events in a de Bruijn graph that are not contained in a repeat-associated subgraph, thus answering to the main open question of [12].

KisSplice [12] is a method for de novo calling of AS events through the enumeration of so-called *bubbles*, that correspond to pairs of vertex-disjoint paths in a de Bruijn graph. The bubble enumeration algorithm proposed in [12] was later improved in [1]. However, even the improved algorithm is not able to enumerate all bubbles corresponding to AS events in a de Bruijn graph. There are certain complex regions in the graph, likely containing repeat-associated subgraphs but also real AS events [12], where both algorithms take a huge amount of time. Figure 3 shows an example of a complex region with a bubble corresponding to an AS event. In practice, the enumeration is halted after a given timeout. The bubbles *drowned* (or trapped) inside these regions are thus missed by KisSplice.

In "Repeats in de Bruijn graphs" section, the repeat-associated subgraphs are characterised by the presence of few compressible arcs. This suggests that in order to avoid repeat-associated subgraphs, we should restrict the search to bubbles containing many compressible arcs. Equivalently, in a compressed de Bruijn graph (see "Preliminaries" section), we should restrict the search to bubbles with few branching vertices. We recall that a *branching vertex* is a vertex of in-degree or out-degree strictly at least 2. Indeed, in a compressed de Bruijn graph, given a fixed sequence length, the number of branching vertices in a path is inversely proportional to the number of compressible arcs of the corresponding path in the non-compressed de Bruijn graph. We thus modify the definition of $(s, t, \alpha_1, \alpha_2)$-bubbles in compressed de Bruijn graphs (Def. 1 in [1]) by adding the extra constraint that each path should have at most b branching vertices.

Definition 2 Given a weighted directed graph $G = (V, E)$ and two vertices $s, t \in V$, an $(s, t, \alpha_1, \alpha_2, b)$ -bubble is a pair of vertex-disjoint st-paths π_1, π_2 with lengths bounded by α_1, α_2, each containing at most b branching vertices.

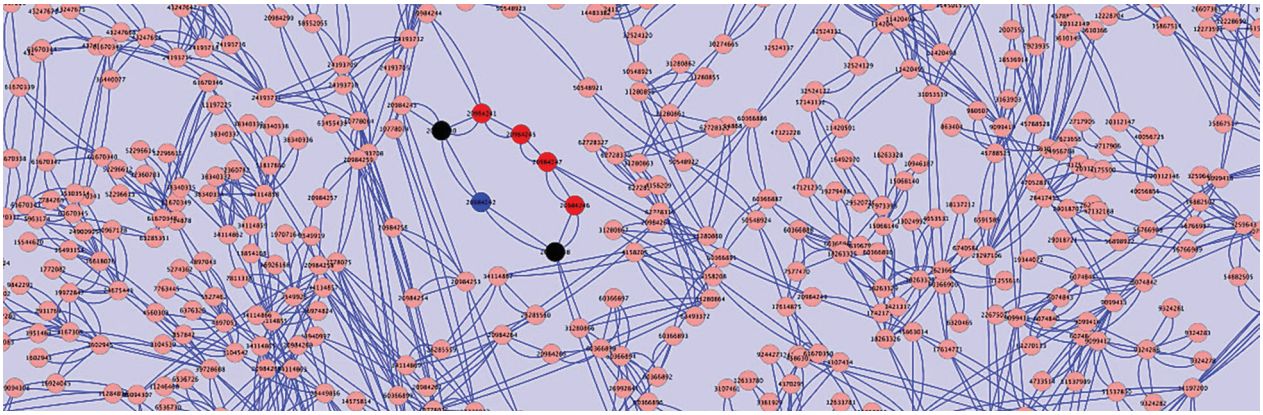


Fig. 3 An alternative splicing event in the SCN5A gene (human) [22] trapped inside a complex region, likely containing repeat-associated subgraphs, in a de Bruijn graph. The alternative isoforms correspond to a pair of paths shown in *red* and *blue*

By restricting the search to bubbles with few branching vertices, we are able to enumerate them in complex regions implicitly avoiding repeat-associated subgraphs. Indeed, in "Experimental results" section we show that by considering bubbles with at most b branching vertices in KisSplice, we increase both its sensitivity and precision. This supports our claim that by focusing on $(s, t, \alpha_1, \alpha_2, b)$-bubbles, we avoid repeat-associated subgraphs and recover at least part of the bubbles trapped in complex regions.

Enumerating bubbles avoiding repeats

In this section, we modify the algorithm of [1] to enumerate all bubbles with at most b branching vertices in each path. Given a weighted directed graph $G = (V, E)$ and a vertex $s \in V$, let $\mathcal{B}_s(G)$ denote the set of $(s, *, \alpha_1, \alpha_2, b)$ -bubbles of G. The algorithm recursively partitions the solution space $\mathcal{B}_s(G)$ at every call until the considered subspace is a singleton (contains only one solution), and in that case it outputs the corresponding solution. In order to avoid unnecessary recursive calls, it maintains the invariant that the current partition contains at least one solution. The algorithm proceeds as follows.

Invariant At a generic recursive step on vertices u_1, u_2 (initially, $u_1 = u_2 = s$), let $\pi_1 = s \rightsquigarrow u_1, \pi_2 = s \rightsquigarrow u_2$ be the paths discovered so far (initially, π_1, π_2 are empty). Let G' be the current graph (initially, $G' := G$). More precisely, G' is defined as follows: remove from G all the vertices in π_1 and π_2 but u_1 and u_2. Moreover, we also maintain the following invariant (INV): there exists at least one pair of paths $\bar{\pi}_1$ and $\bar{\pi}_2$ in G' that extend π_1 and π_2 so that $\pi_1 \cdot \bar{\pi}_1$ and $\pi_2 \cdot \bar{\pi}_2$ belong to $\mathcal{B}_s(G)$.

Base case When $u_1 = u_2 = u$, output the $(s, u, \alpha_1, \alpha_2, b)$ -bubble given by π_1 and π_2.

Recursive rule Let $\mathcal{B}_s(\pi_1, \pi_2, G')$ denote the set of $(s, *, \alpha_1, \alpha_2, b)$-bubbles to be listed by the current recursive call, i.e. the subset of $\mathcal{B}_s(G)$ with prefixes π_1, π_2. It is the union of the following disjoint sets:

- The bubbles of $\mathcal{B}_s(\pi_1, \pi_2, G')$ that use e, for each arc $e = (u_1, v)$ outgoing from u_1, that is $\mathcal{B}_s(\pi_1 \cdot e, \pi_2, G' - u_1)$, where $G' - u_1$ is the subgraph of G' after the removal of u_1 and all its incident arcs.
- The bubbles that do not use any arc from u_1, that is $\mathcal{B}_s(\pi_1, \pi_2, G'')$, where G'' is the subgraph of G' after the removal of all arcs outgoing from u_1.

The same holds for u_2 instead of u_1.

In order to maintain the invariant (INV), we only perform the recursive calls when $\mathcal{B}_s(\pi_1 \cdot e, \pi_2, G' - u)$ or $\mathcal{B}_s(\pi_1, \pi_2, G'')$ are non-empty. In both cases, we have to decide if there exists a pair of (internally) vertex-disjoint paths $\bar{\pi}_1 = u_1 \rightsquigarrow t_1$ and $\bar{\pi}_2 = u_2 \rightsquigarrow t_2$, such that $|\bar{\pi}_1| \leq \alpha'_1$, $|\bar{\pi}_2| \leq \alpha'_2$, and $\bar{\pi}_1, \bar{\pi}_2$ have at most b_1, b_2 branching vertices, respectively. Since both the length and the number of branching vertices are monotonic properties, i.e. both are smaller for a prefix instead of for the full path, we can drop the vertex-disjoint condition. Indeed, let $\bar{\pi}_1$ and $\bar{\pi}_2$ be a pair of paths satisfying all conditions but the vertex-disjoint one. The prefixes $\bar{\pi}_1^* = u_1 \rightsquigarrow t^*$ and $\bar{\pi}_2^* = u_2 \rightsquigarrow t^*$, where t^* is the first intersection of the paths, satisfy all conditions and are internally vertex-disjoint.

Moreover, using a dynamic programming algorithm, we can obtain the following result.

Lemma 1 *Given a non-negatively weighted directed graph $G = (V, E)$ and a source $s \in V$, we can compute the shortest paths from s using at most b branching vertices in $O(b|E|)$ time.*

Proof Let $d[\beta, t]$ denote the distance from s to t using at most β branching vertices (s is never counted as a branching vertex, even if it is branching). The recurrence to calculate $d[\beta, t]$, for $0 \leq \beta \leq b$ and $t \in V$ is:

Initialisation step:

$d[0, s] = 0$;
$d[0, t] = |(s, t)|$ if $(s, t) \in E$ and t is not branching;
$d[\beta, t] = +\infty$ if $d[\beta, t]$ was not initialised.

Main recurrence:

$$d[\beta, t] = \begin{cases} \min(\min_{v \in N^-(t)}\{d[\beta - 1, v] + |(v, t)|\}, d[\beta - 1, t]), & \text{if t is branching} \\ \min(\min_{v \in N^-(t)}\{d[\beta, v] + |(v, t)|\}, d[\beta - 1, t]), & \text{if t is not branching.} \end{cases}$$

This recurrence works only on compressed graphs, i.e. it requires that the neighbours of simple vertices are branching. However, since the graph compression procedure described in "Preliminaries" section can be applied to general graphs, this recurrence is also applicable to general graphs. The calculation order for $d[\beta, t]$ in the main recurrence must be by increasing value of β and, for a fixed β, the branching vertices must be processed before the non-branching ones. Moreover, the shortest paths themselves can be constructed by a traceback procedure.

Finally, since the calculation of each value $d[\beta, t]$ takes $O(|N^-(t)|)$ time, the algorithm runs in $O(b \sum_{t \in V} |N^-(t)|) = O(b|E|)$ time. We can guarantee that this algorithm runs in time polynomial in the length of the input by upper-bounding b by $|V|$ (if $b > |V|$, we simply set $b = |V|$). □

As a corollary of Lemma 1, we can decide if $\mathcal{B}_s(\pi_1, \pi_2, G)$ is non-empty in $O(b|E|)$ time. Now, using an argument similar to [1], i.e. the leaves of the recursion tree and the solutions are in one-to-one correspondence and the height of the recursion tree is bounded by $4b$, we obtain the following theorem.

Theorem 4 *The $(s, *, \alpha_1, \alpha_2, b)$-bubbles can be enumerated in $O(b^2|E||\mathcal{B}_s(G)|)$ time. Moreover, the time elapsed between the output of any two consecutive solutions (i.e. the delay) is $O(b^2|E|)$.*

Measuring the confidence of a transcript in full-length transcriptome assemblers

Reconstructing full-length transcripts from reads is a challenging task because two transcripts, even from different genes, may very well share subsequences that are longer than the sequenced reads, or even longer than the fragments in case of paired-end sequencing. This is specially true when genes host transposable elements within their introns, and less frequently but still present, within their UTRs and also exons (e.g. exonised repeats). Even if a repeat-containing intron is always spliced out in the splicing phase, this intron, and consequently the repeat, can still be present in RNA-seq data. The fraction of introns present in the sequenced data depends on the cell compartment that is sampled (nucleus, cytoplasm or both) and the protocol to remove rRNA (ribo-0 or polydT primers). As estimated in [9], the level of pre-mRNA can be assumed to vary between 2 and 22%. The true level of pre-mRNA may however be in practice higher, because the methods used for estimating it are mapping-based and therefore deal poorly with reads stemming from repeated regions. Besides, the upper bound given in [9] corresponds to extraction protocols which are harder to obtain. In this work, we considered the most commonly used extraction protocol to extract RNA, and assumed that they yielded pre-mRNA fractions between 5 and 15%. Thus, more introns than expected are sequenced, generating problems to transcriptome assemblers, particularly when they span several members of a specific repeat family.

Most transcriptome assemblers are based on de Bruijn graphs and have no clear and explicit model for repeats in RNA-seq data, relying instead on heuristics to deal with them. Within the complex parts of the graph generated by repeats, any assembler will have to choose the "right" path(s) among the many present. Even with hints given by (paired-end) reads, assemblers can still have several arguable options to extend a contig (see Fig. 4). This problem gets harder if the (paired-end) reads do not span the repeat entirely, thereby not giving the assembler any reliable information on how to connect the unique regions. If the assembler decides to guess a path, it may erroneously extend a contig and create a chimeric transcript. It can also choose to be conservative by not choosing any path in complicated regions of the de Bruijn graph, and instead truncating the transcript. Although this strategy can lead to an accurate assembly, it will produce a very fragmented one, which is not desired. Whatever the strategy (conservative or permissive), the resulting assembled transcript may be erroneous (chimeric or truncated).

It is hence important to be able to identify low-confidence transcripts, which are the ones traversing complex regions of a de Bruijn graph, in order to know that the solution presented is the result of a "difficult" choice and therefore *may* not be the right one. To identify such transcripts, we introduce the concept of *Branching Measure* of a transcript. Consider the set of transcripts $\mathcal{T}$ output

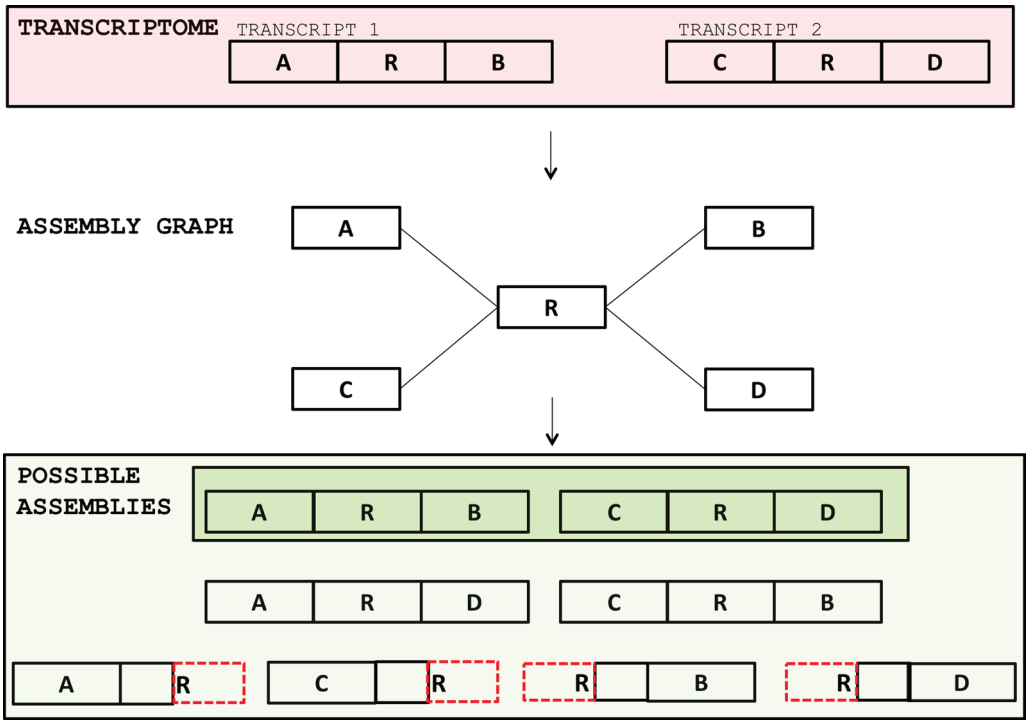
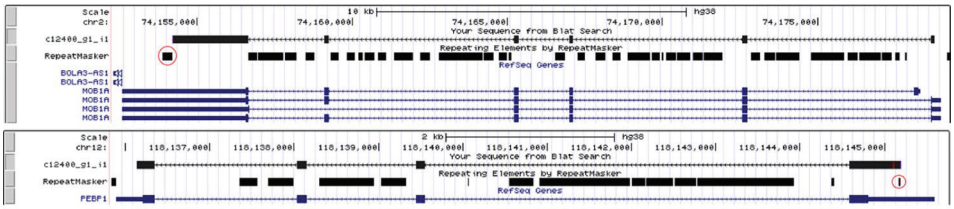


Fig. 4 A theoretical scenario showing some problems repeats cause to assemblers. On the top of the figure, we can see two real transcripts containing each one a member of a repeat R. When building the assembly graph, the two copies of R may collapse into a single region of the graph, and connect the unique regions of both transcripts. The only correct assemblies are ARB and CRD, but the assembly graph also allows for the generation of the chimeric transcripts ARD and CRB, or truncated transcripts, in case the assembler chooses to be conservative

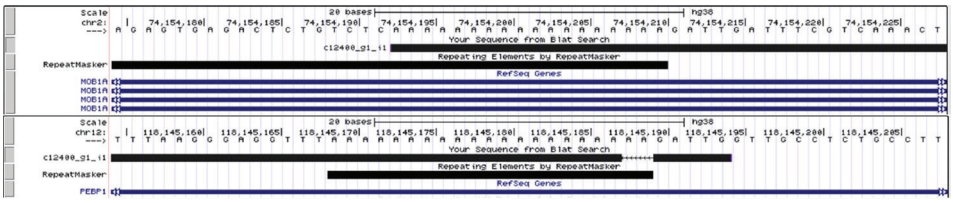
by a full-length transcriptome assembler starting from a set of reads $\mathcal{R}$. We construct the de Bruijn graph $G_k(R)$, and map back each transcript $t \in \mathcal{T}$ to the graph by identifying each of its k-mers. Given a positive integer w, let W be a w-sized window (or substring) with the largest number of branching k-mers in t. We define the Branching Measure of a transcript t, $B(t)$, as the proportion of branching k-mers in W. By looking at $B(t)$, it is possible to infer if t traversed a hard-to-assemble region in the de Bruijn graph, and this can be used as a measure of its confidence, i.e. the higher $B(t)$ is, the lower is the confidence of t.

As a proof of concept, in the following we show two examples of the application of the Branching Measure to transcripts assembled by Trinity on RNA-seq data from the GEUVADIS project [18].

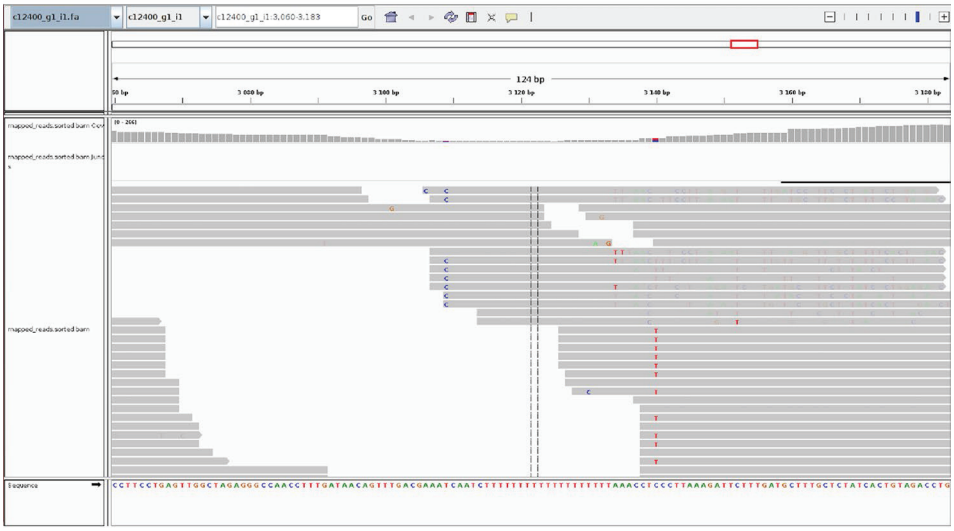
The first example (Fig. 5) is the chimeric transcript c12400_g1_i1 that aligns to the gene MOB1A in chromosome 2 and also to the gene PEBP1 in chromosome 12, in which the fusion of these genes is due to a small identical region shared between two different repeats present in their UTR regions. Figure 5a shows the alignment of the transcript c12400_g1_i1 to reference hg38, visualised using the UCSC Genome Browser. The alignment on the top shows that the built transcript aligns almost perfectly to an isoform of gene MOB1A in chromosome 2. Due to the repeats inside the red circles, the alignment is truncated in the 3′-UTR of MOB1A, and continued on the 5′-UTR of gene PEBP1 in chromosome 12 (alignment on the bottom). Thus, here we have a chimeric transcript. Figure 5b zooms in the regions where both alignments intersect the repeats that cause the chimerism. The main reason of the junction between the two genes is due to a stretch of 18 As shared between the A-tail of a SINE AluY in the 3′-UTR of MOB1A and a Simple Repeat A(n) in the 5′-UTR of PEBP1. Even though this repeated region is short, it was enough to cause problems to Trinity, which had access to 76-bp paired-end reads, with an average insert size of 158 bp. In Fig. 5c we mapped all reads back to transcript c12400_g1_i1 and visualised them using IGV [19]. As we can see, there are no single or paired-end reads traversing the small repeat. This shows that this chimera is not an in vitro or a biological one, but indeed an assembly mistake by Trinity. Figure 5d conveys a local visualisation of the subgraph induced by the k-mers of transcript c12400_g1_i1 at the junction point which causes the chimerism (the full graph can be accessed at http://kissplice.prabi.fr/bm/graph_chimera.html). We can see that this is a complex region since the transcript (red path) traverses a region



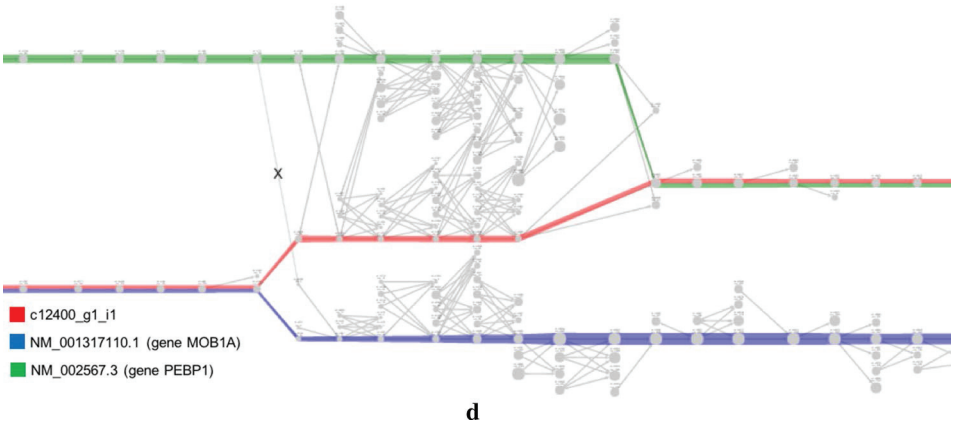
a



b



c



d

Fig. 5 The chimeric transcript c12400_g1_i1 that aligns to the gene MOB1A in chromosome 2 and also to the gene PEBP1 in chromosome 12, in which the fusion of these genes is due to a small identical region shared between two different repeats present in their UTR regions (see "Measuring the confidence of a transcript in full-length transcriptome assemblers" section for details of each panel). **a** The alignment of the transcript c12400_g1_i1 to reference hg38, visualised using the UCSC Genome Browser. **b** The regions where both alignments intersect the repeats that cause the chimerism. **c** The mapping of all reads to transcript c12400_g1_i1 visualised using IGV. **d** A local visualisation of the subgraph induced by the *k*-mers of transcript c12400_g1_i1 at the junction point which causes the chimerism

having 11 branching *k*-mers in a window of 12, and could thus be flagged by the Branching Measure. There is no other such complex region in this transcript, i.e. this is the only hard-to-assemble region that this transcript goes through. We can also see in the picture the correct extension which should have been followed as the reference transcripts (the green and blue paths). Observe that even the reference transcripts could also have been flagged by our method since they traverse regions containing a concentration of branching vertices due to the repeated elements presented in Fig. 5a, b.

The second case, depicted in Fig. 6, shows a mis-assembly of the last exon of gene SLC35F2, in which Trinity assembled several truncated transcripts instead of the full exon. Figure 6a shows, on the 3' → 5' orientation (reverse strand), the three truncated short transcripts: c65590_g1_i1, c64_g1_i1, and c14482_g2_i1. The truncation points were cause caused by repeats, where the first split is due to a simple repeat (A(n)) and the second is due to 2 consecutive Alus (AluJo and AluSz). Figure 6b displays a schematic global view on how the last exon of gene SLC35F2 was assembled by Trinity and how the three

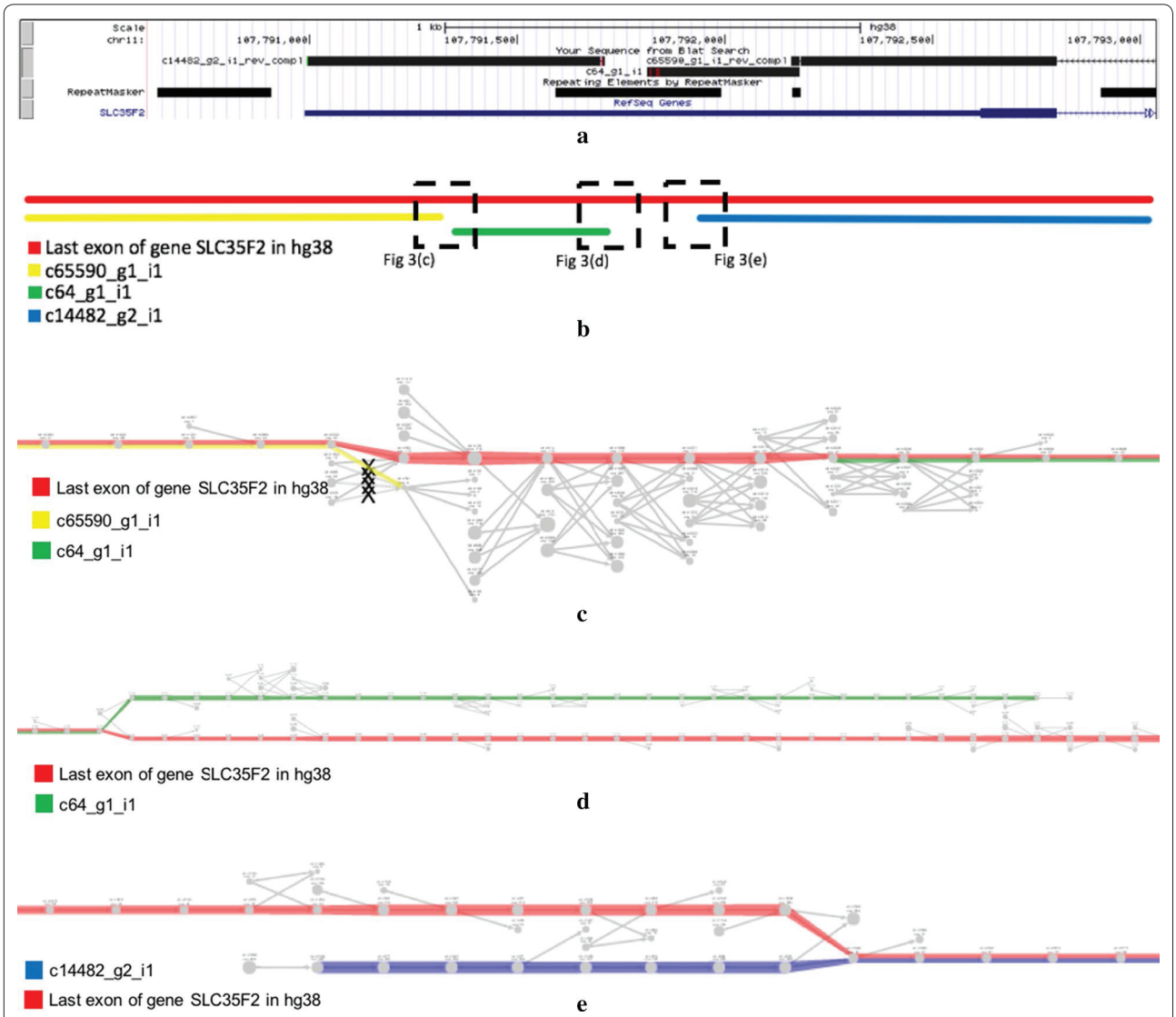


Fig. 6 A mis-assembly of the last exon of gene SLC35F2, in which Trinity assembled several truncated transcripts instead of the full exon (see "Measuring the confidence of a transcript in full-length transcriptome assemblers" section for details of each panel). **a** The three truncated short transcripts: c65590_g1_i1, c64_g1_i1, and c14482_g2_i1. **b** A schematic global view on how the last exon of gene SLC35F2 was assembled by Trinity and how the three next figures are connected in the full graph drawing. **c** A local visualisation of the truncation point between c65590_g1_i1 and c64_g1_i1 due to a simple repeat. **d** A local view of the region that traverses the repeat AluJo, and where the assembler has chosen to truncate the transcript c64_g1_i1. **e** A local view of the region that traverses the repeat AluSz, and where the assembler has chosen to truncate the transcript c14482_g2_i1

next figures are connected in the full graph drawing. This figure and the next assume the $5' \rightarrow 3'$ orientation. Figure 6c conveys a local visualisation of the truncation point between c65590_g1_i1 and c64_g1_i1 due to a simple repeat. We can see that Trinity mis-assembled the very end of c65590_g1_i1 (only the last base) and truncated the transcript. The yellow path is accurate although truncated and does not go through a complicated region (one having a concentration of branching vertices). Even though the reference exon path in this region has 11 consecutive branching vertices and would be flagged by the Branching Measure, this method is unable to flag c65590_g1_i1 since it is truncated too early, before entering the complex region. Figure 6d shows a local view of the region that traverses the repeat AluJo, and where the assembler has chosen to truncate the transcript c64_g1_i1. We can see that Trinity mis-assembled the last 29 bases of c64_g1_i1 and truncated it. At the end of c64_g1_i1, we have 23 branching vertices in a window of 34 vertices, so this truncated transcript can be flagged by our method, as it is deeply enough plunged into a complex region. Finally, Fig. 6e displays a local view of the region that traverses the repeat AluSz, and where the assembler has chosen to truncate the transcript c14482_g2_i1. Again, the Branching Measure is not able to flag this transcript since it is not deeply enough plunged into a complex region. The full graph of Fig. 6b–e can be accessed at http://kissplice.prabi.fr/bm/graph_truncated.html.

Experimental results

Local assembly: experimental setup

To evaluate the performance of our method, we simulated RNA-seq data using the FluxSimulator version 1.2.1 [20]. We generated 100 million reads of 75 bp using its default error model. We used the RefSeq annotated Human transcriptome (hg19 coordinates) as a reference and we performed a two-step pipeline to obtain a mixture of mRNA and pre-mRNA (i.e. with introns not yet spliced). To achieve this, we first ran the FluxSimulator with the Refseq annotations. We then modified the annotations to include the introns and re-ran it on this modified version. In this second run, we additionally constrained the expression values of the pre-mRNAs to be correlated to the expression values of their corresponding mRNAs, as simulated in the first run. Finally, we mixed the two sets of reads to obtain a total of 100M reads. We tested two values, namely 5 and 15% for the proportion of reads from pre-mRNAs. Those values were chosen so as to correspond to realistic ones (see "Measuring the confidence of a transcript in full-lengthtranscriptome assemblers" section).

On these simulated datasets, we ran KisSplice [12] versions 2.1.0 (Ks_2.1.0) and 2.2.0 (Ks_2.2.0, with a maximum number of branching vertices set to 5) and obtained lists of detected bubbles that are putative alternative splicing (AS) events. We also ran the full-length transcriptome assemblers Trinity version r2013_08_14 and Oases version 0.2.09 on both datasets, obtaining a list of predicted transcripts, from which we then extracted a list of putative AS events. For Oases, there was one locus in each dataset for which we were not able to extract the putative AS events. A manual inspection revealed that they were mostly composed of subparts of introns or UTRs flanked by repeats, usually copies of ALUs. The presence of such high copy-number repeats in these transcripts induced the clustering of all these unrelated sequences into one complex locus. More precisely, in the dataset containing 5% of the reads from pre-mRNAs, the largest locus that Oases predicted comprised 20,769 transcripts, while the second largest contained only 139 transcripts. In the other simulated dataset, the largest locus comprised 39,389 transcripts, and the second largest contained only 205 transcripts. This indicates that Oases faces similar issues to Ks_2.1.0. For fairness of comparison, we did not post-process these complex loci, and we were then unable to retrieve the potential AS events that they could describe. It is worth mentioning though, that the majority of the transcripts inside these loci corresponded to subparts of introns or UTRs, and not to full introns or exons, and therefore could not describe AS events.

In order to assess the precision and the sensitivity of our method, we compared our set of *found* AS events to the set of *true* AS events. Following the definition of Astalavista, an AS event is composed of two sets of transcripts, the inclusion/exclusion isoforms respectively. We consider that an AS event is *true* if at least one transcript among the inclusion isoforms and one among the exclusion isoforms is present in the simulated dataset with at least 5 reads per kilobase (RPK). The rationale for adding this threshold is that very rare events are considerably hard, or even impossible, to detect by all methods.

To compare the results of KisSplice with the *true* AS events, we propose that a true AS event is a *true positive* (TP) if there is a bubble such that one path matches the inclusion isoform and the other the exclusion isoform. If there is no such bubble among the results of KisSplice, the event is counted as a *false negative* (FN). If a bubble does not correspond to any *true* AS event, it is counted as a *false positive* (FP). To align the paths of the bubbles to transcript sequences, we used the Blat aligner [21] with 95% identity and a constraint of 95% of each bubble path length to be aligned (to account for the sequencing errors simulated by FluxSimulator). We computed the

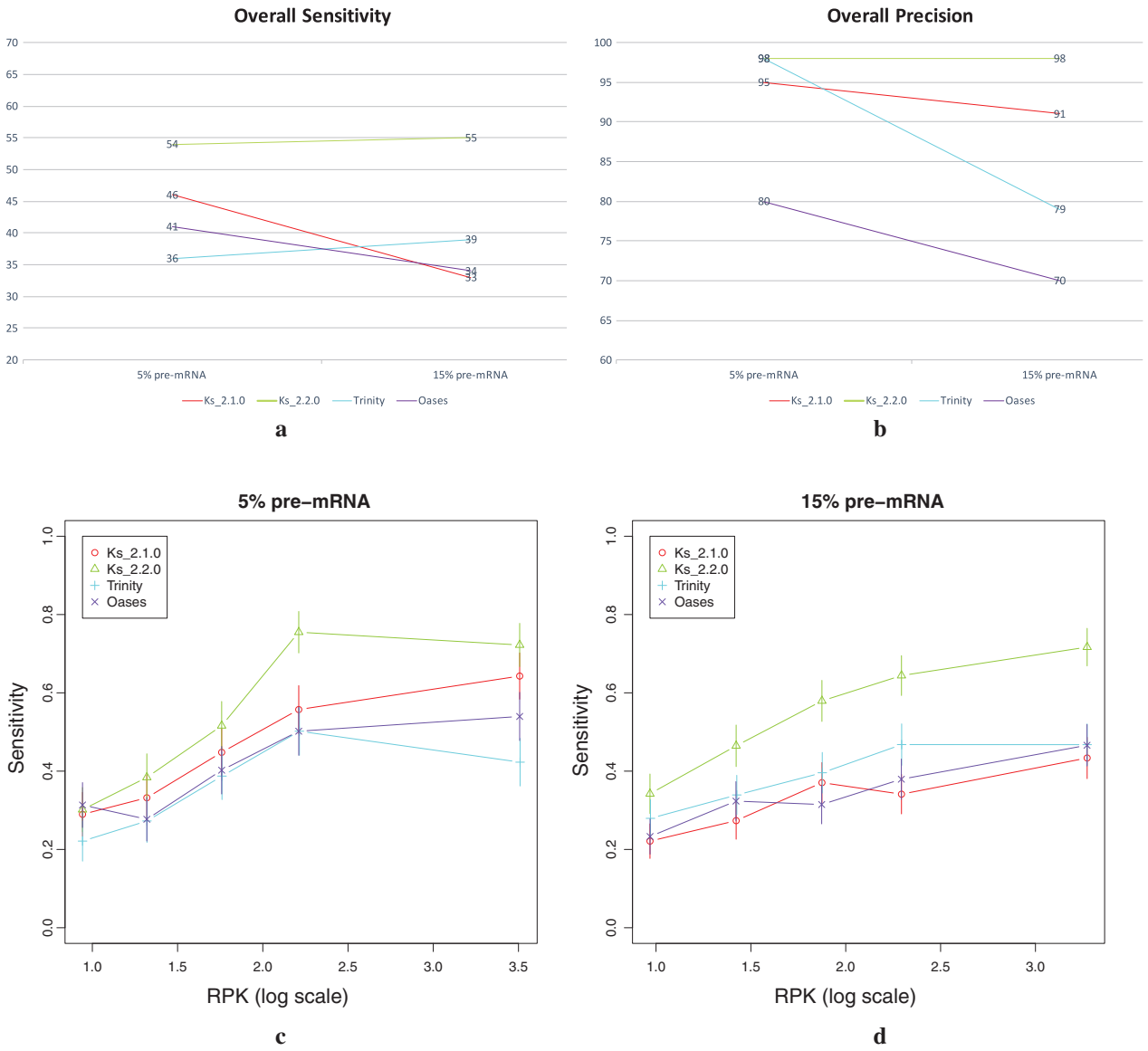


Fig. 7 The overall values for sensitivity and precision, and the detailed sensitivity by expression levels of Ks_2.1.0, Ks_2.2.0, Trinity and Oases on the two simulated datasets. **a** Overall sensitivity of the four methods on the two simulated datasets. **b** Overall precision of the four methods on the two simulated datasets. **c** Detailed sensitivity by expression levels of the four methods on the 5% pre-mRNA dataset. **d** Detailed sensitivity by expression levels of the four methods on the 15% pre-mRNA dataset. The expression levels in **c** and **d** represent several classes of expression of the minor isoform. Each class (i.e. point in the graph) contains the same number of AS events. It is therefore an average sensitivity on a potentially broad class of expression

sensitivity TP/(TP+FN) and precision TP/(TP+FP) for each simulation case and we report their values for various classes of expression of the minor isoform. Expression values are measured in RPK.

Local assembly: results

The overall sensitivity and precision of Ks_2.2.0, Ks_2.1.0, Trinity and Oases can be found in Fig. 7a, b, respectively. A first look reveals that Ks_2.2.0 outperforms the other three methods in both measures and datasets.

A closer look at Fig. 7a shows that both versions of KisSplice had better sensitivity than both transcriptome assemblers in the 5% pre-mRNA dataset. However, due to its inability to deal with repeat-associated regions, the performance of Ks_2.1.0 drops substantially, from 46 to

33%, when a higher rate of 15% of pre-mRNA is present in the data. The same happened with Oases. Ks_2.2.0 and Trinity, on the other hand, were able to slightly improve their sensitivity from the 5 to the 15% pre-mRNA dataset. It is however worth mentioning that the sensitivity of Ks_2.2.0 is substantially higher than the one of Trinity in the 15% pre-mRNA dataset. In summary, we can say that Ks_2.2.0 is substantially more sensitive than all the other three methods. This reflects the fact that most problematic repeats are in introns. A small unspliced mRNA rate leads to few repeat-associated subgraphs, so there are not many AS events drowned in them. In this case, the advantage of using Ks_2.2.0 is less obvious, whereas a large proportion of pre-mRNA leads to more AS events drowned in repeat-associated subgraphs which are identified by Ks_2.2.0 and usually missed by the other methods.

In Fig. 7b we can see that Ks_2.2.0 and Trinity presented the highest precision (98%) of all methods in the 5% pre-mRNA dataset. Although Ks_2.1.0 is ranked third, it still presents a very high precision (95%), while Oases presented a moderate value (80%). Nevertheless, the most important aspect to be observed in Fig. 7b is that Ks_2.2.0 kept the same high precision even when a higher rate of 15% of pre-mRNA is present in the data. Trinity, on the other hand, dropped significantly from 98 to 79%. This drop in precision is actually mostly due to the prediction of a large number of intron retentions, since Trinity assembles both the mRNA and the pre-mRNA. Ks_2.1.0 dropped slightly from 95 to 91%, and Oases dropped moderately, from 80 to 70%. In summary, we can say that both versions of KisSplice are more precise than both transcriptome assemblers, except that Trinity shows comparable precision if a small rate of pre-mRNA is present in the data, and, more specifically, that Ks_2.2.0 outperformed all the other three methods. The high precision we obtain indicates that we have very few false positives. Those mostly correspond to repeat-induced bubbles mistakenly identified as AS events.

Finally, Fig. 7c, d present the detailed sensitivity by expression levels of the four methods on both datasets, allowing for a better understanding of their performance. As we can see, Ks_2.2.0 presented the best sensitivity in practically all expression levels in both datasets, while the other three methods were worse, but comparable between themselves. We can also observe that the gap between the sensitivity of Ks_2.2.0 and the sensitivity of the other methods tends to increase as the expression levels of the genes increase, especially in the 15% pre-mRNA dataset. Since highly-expressed genes tend to present higher levels of pre-mRNA content, they bring several repeat copies in their introns, and thus some parts of their associated graphs are complex and repeat-induced. Therefore, AS events inside such genes tend to be trapped in troublesome regions of the graph, making them harder to find. As Ks_2.2.0 is able to avoid such complex regions and retrieve the AS events deeply plunged into them, it presents better sensitivity than the other methods, especially in highly-expressed genes and datasets with high rate of pre-mRNAs.

As was already reported in [12], KisSplice (i.e. both Ks_2.2.0 and Ks_2.1.0) is faster and uses considerably less memory than Trinity and Oases. For instance, on these datasets, KisSplice uses around 5 GB of RAM, while Trinity uses more than 20 GB, and Oases, around 18 GB. However, it should be noted that both these latter methods try to solve a more general problem than KisSplice, that is reconstructing the full-length transcripts.

To conclude, our results show that Ks_2.2.0 is significantly more sensitive and precise than Ks_2.1.0, Trinity and Oases for the task of identifying AS events. The advantage of using Ks_2.2.0 over the other three methods is more evident when the input data contains high pre-mRNA content or the AS events of interest stem from highly-expressed genes.

On the usefulness of Ks_2.2.0 on real data

In order to give an indication of the usefulness of our repeat-avoiding bubble enumeration algorithm with real data, we also ran Ks_2.2.0 and Ks_2.1.0 on the SK-N-SH Human neuroblastoma cell line RNA-seq dataset (wgEncodeEH000169, total RNA). In Fig. 8, we have an example of a *non-annotated* exon skipping event not found by Ks_2.1.0. Observe that the intronic region contains several transposable elements (many of which are Alu sequences), while the exons contain none. This is a good example of a bubble (exon skipping event) drowned in a complex region of the de Bruijn graph. The bubble (composed by the two alternative paths) itself contains no repeated elements, but is surrounded by them. In other words, this is a bubble with few branching vertices that is surrounded by repeat-associated subgraphs. Since Ks_2.1.0 is unable to differentiate between repeat-associated subgraphs and the bubble, it spends a prohibitive amount of time in the repeat-associated subgraph and fails to find the bubble.

Global assembly

To test our hypothesis that the Branching Measure is able to identify problematic transcripts, we evaluated it on the transcripts assembled by Trinity on the two simulated RNA-seq datasets described in "Local assembly: results" section, and on two other real RNA-seq datasets: one from the GEUVADIS project [18][1] and one from a neuro-

[1] This dataset can be found at the ArrayExpress database (http://www.ebi.ac.uk/arrayexpress/) under the accession number E-GEUV-6, and we used the individual named NA06994, extract name "NA06994.2.M_111215_7 extract".

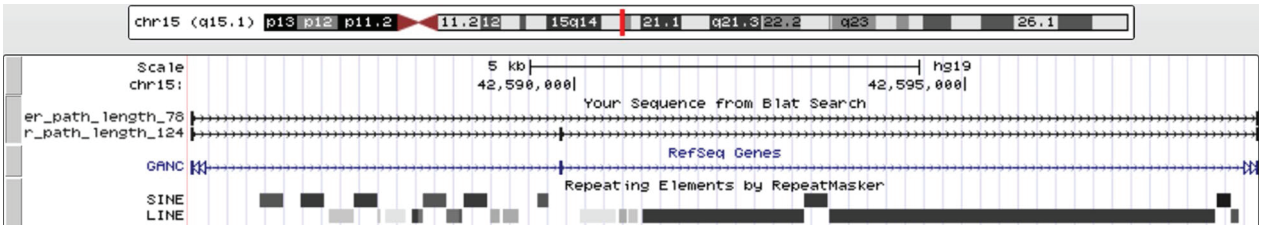


Fig. 8 One of the bubbles found only by Ks_2.2.0 with the corresponding sequences mapped to the reference human genome and visualised using the UCSC Genome Browser. The *first two lines* correspond to the sequences of, respectively, the shortest (exon exclusion variant) and longest paths (exon inclusion variant) of the bubble mapped to the genome. The *blue line* is the Refseq annotation. The *last line* shows the annotated SINE and LINE sequences (transposable elements)

blastoma SK-N-SH cell line (ENCODE) differentiated or not using retinoic acid.[2] Even though our method is reference-free, we have chosen to evaluate it under a model species so that we could make use of annotated reference genomes to assess if our predictions are correct. We compared our measure against two state-of-the-art methods for de novo transcriptome evaluation, RSEM-EVAL [4] and TRANSRATE [5], on the specific task of identifying chimeric transcripts in TRINITY's assemblies on all four described datasets. In all our tests, we used the *contig impact score* of RSEM-EVAL as a measure of contig correctness. Formally, the *contig impact score* is a statistical measure that compares the hypothesis that a particular contig (i.e. transcript) is a true contig with the null hypothesis that the reads composing the contig actually represent the background noise [4]. In other words, the *contig impact score* determines the relative contribution of each transcript to explaining the assembly. Well-assembled transcripts should therefore have a high *contig impact score*, and badly assembled transcripts, including chimeras, should have a low *contig impact score*. TRANSRATE [5], on the other hand, allowed us to work with a specific metric representing the probability that a contig is derived from a single transcript. This metric denotes the probability that the read coverage of a transcript is best modelled by a single Dirichlet distribution, rather than two or more distributions, and it corresponds to the field sCseq of TRANSRATE's output file CONTIGS.CSV.

As was shown before, one of the main errors that transcriptome assemblers do is to build chimeric transcripts. We compared the performances of the Branching Measure, RSEM-EVAL, and TRANSRATE on identifying chimeric transcripts. In order to have our ground truth, we first identified which assembled transcripts are chimeric with respect to a reference genome by using Algorithm 1. In total, 253 out of 18,706 transcripts (1.3%) in the 5% pre-mRNA dataset, 376 out of 26,407 transcripts (1.4%) in the 15% pre-mRNA dataset, 375 out of 99,591 transcripts (0.3%) in the GEUVADIS dataset, and 2830 out of 457,383 transcripts (0.6%) in the SKNSH dataset were classified as chimeric. Figure 9 depicts four ROC curves showing the performance of the three methods on all datasets. We can observe that the Branching Measure outperforms both RSEM-EVAL and TRANSRATE by a large margin in all tests and, with a high-value threshold, is also able to flag a majority of the chimeric transcripts while keeping a low false positive rate. These experiments show that, in the provided datasets, chimeric transcripts could be well captured by the Branching Measure. Our false positives include well-assembled transcripts traversing high copy-number low divergence repeats, and our false negatives include chimeric transcripts that did not go through a complex region. The main issue with RSEM-EVAL and TRANSRATE, on the other hand, is that both methods failed to find chimeric transcripts assembled from genes with similar expression levels. These chimeras had low scores and corresponded to the false negatives at the end of the ROC curves for RSEM-EVAL and TRANSRATE. As a side effect of this misclassification, many well-assembled transcripts had higher scores than several real chimeras, and were mistakenly flagged as chimeras.

[2] This dataset can be found at http://genome.crg.es/encode_RNA_dashboard/hg19/, and is also accessible with the following accession numbers: ENCSR000CPN—SRA: SRR315315, SRR315316 and ENCSR000CTT—SRA: SRR534309, SRR534310. For cell lines treated by retinoic acid, the reads were 76nt long, while they were 100nt long for the non treated cells. Hence we trimmed all reads to 76nt.

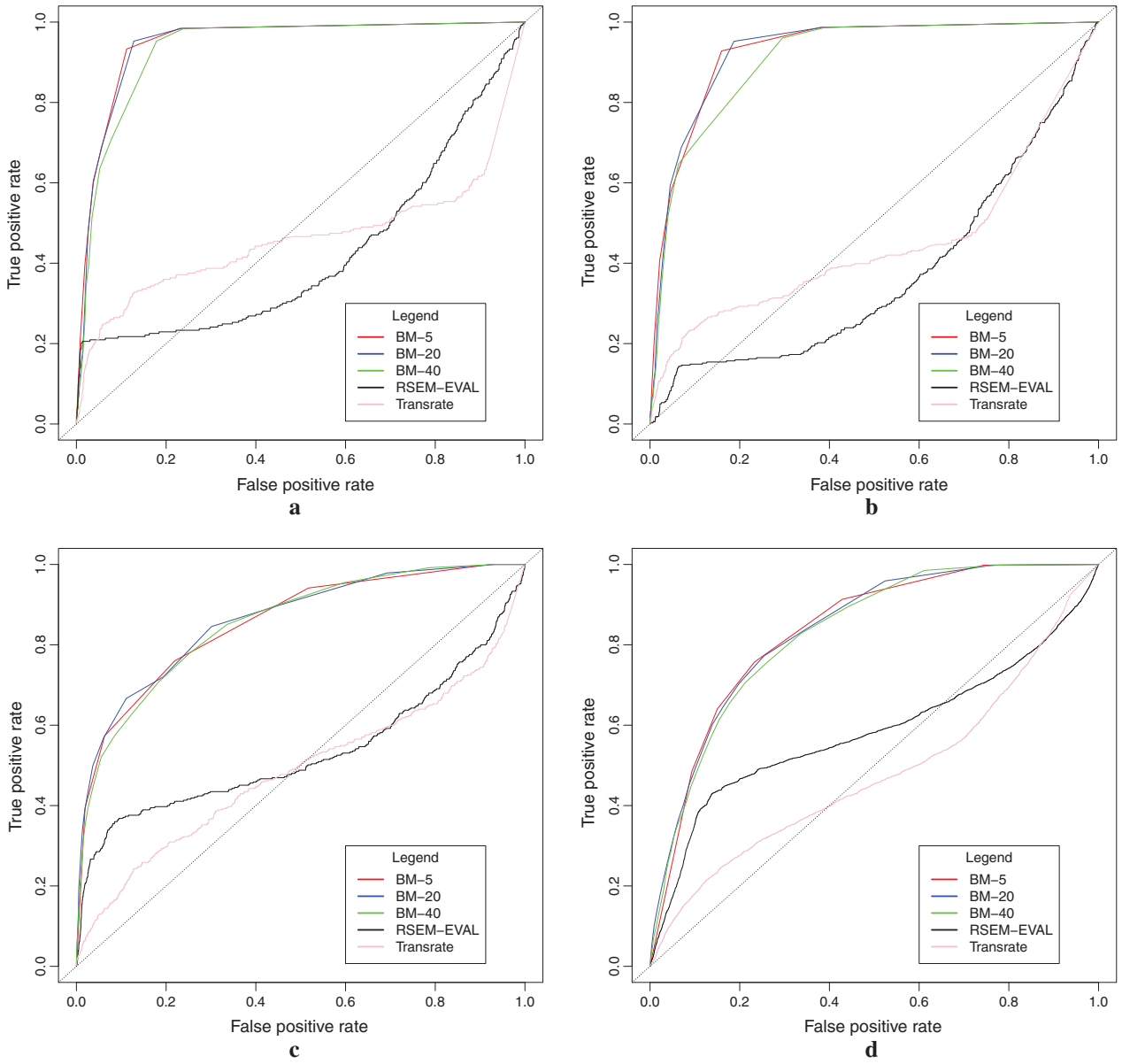


Fig. 9 The performance of the Branching Measure, RSEM-EVAL, and TRANSRATE on identifying chimeric transcripts on the four datasets described in "Global assembly" section. BM-*x* stands for Branching Measure using a window of size *x*. In this test, the 10% leftmost and rightmost parts of the transcripts were disregarded in the Branching Measure calculation. **a** Simulated dataset with 5% pre-mRNA. **b** Simulated dataset with 15% pre-mRNA. **c** GEUVADIS dataset. **d** SKNSH dataset

Algorithm 1: GetChimericTranscripts($\mathcal{T}$, $\mathcal{G}$)

Definition 1: An alignment $a(t, \mathcal{G})$ of t to $\mathcal{G}$ is a **good alignment** if it aligns more than 80% of t with matches;
Definition 2: An alignment $a(t, \mathcal{G})$ of t to $\mathcal{G}$ is a **potential chimeric alignment** if it aligns at least 100 bases, but less than 80% of t with matches;
Definition 3: If we have two alignments a_1 and a_2 such that the largest covers at least 80% of the smallest, we can **merge** a_1 and a_2 into an alignment a_m, where the start position of a_m is the leftmost start position between a_1 and a_2 and the end position of a_m is the rightmost end position between a_1 and a_2.
Data: Set of transcripts $\mathcal{T}$ and a reference genome $\mathcal{G}$
Result: Set of chimeric transcripts $\mathcal{C}$
Map each $t \in \mathcal{T}$ to $\mathcal{G}$ (e.g. using BLAT);
$\mathcal{C} \leftarrow \emptyset$;
foreach $t \in \mathcal{T}$ **do**
  if *t has no good alignments and t has 2 or more potential chimeric alignments* **then**
    Let $MPCA$ be all maximal potential chimeric alignments of t;
    Let $MMPCA$ be a set obtained by merging all alignments in $MPCA$ until convergence;
    if $|MMPCA| \geq 2$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup t$
return $\mathcal{C}$

Concluding remarks and perspectives

Although transcriptome assemblers are now commonly used, their way to handle repeats is not satisfactory, arguably because the presence of repeats in transcriptomes has been underestimated so far. Given that most RNA-seq datasets correspond to total mRNA extractions, many introns are still present in the data and their repeat content cannot be simply ignored. Although repeats in transcriptomic and genomic data cause similar problems to assemblers, the specificities of each mean that a successful strategy in one context may fail in the other. It is thus essential for transcriptome assemblers to formally address the repeats problem.

In this paper, we first proposed a simple formal model for representing high copy-number repeats in RNA-seq data. Exploiting the properties of this model, we established that the number of compressible arcs is a relevant quantitative characteristic of repeat-associated subgraphs. We proved that the problem of identifying in a de Bruijn graph a subgraph with this characteristic is NP-complete. However, this characteristic drove the design of an algorithm for efficiently identifying AS events that are not included in repeated regions. The new algorithm was implemented in KisSplice version 2.2.0, and by using simulated RNA-seq data, we showed that it improves significantly the sensitivity of the previous version of KisSplice, while also improving its precision. In addition, we compared our algorithm to Trinity and Oases, and showed that for the specific task of calling AS events, our algorithm is significantly more sensitive while also being more precise. Our results also showed that the advantage of using KisSplice version 2.2.0 is more evident when the input data contains high pre-mRNA content or the AS events of interest stem from highly-expressed genes. Moreover, we gave an indication of the usefulness of our method on real data. Finally, we explored the proposed model in the context of full-length transcriptome assembly by introducing the Branching Measure, which is able to flag the transcripts that go through a complex region in the de Bruijn graph. Even though one should not directly consider low-confidence transcripts as erroneous ones since they could have been correctly assembled despite the hardness, the described

measure is useful from a user's point-of-view since it enables to flag the transcripts that result from a "difficult" choice during the assembly, no matter which assembler is used. We showed that this measure can indeed capture assembly errors in real cases and, when compared to Rsem-Eval [4] and to TransRate [5] on the specific task of identifying chimeric transcripts, the measure we propose outperformed the ones used by Rsem-Eval and TransRate by a large margin. The originality of our work, when compared to other methods for transcriptome assembly evaluation, is that we use only the topology of the graph. Despite the successful application of the Branching Measure in global transcriptome assembly, it remains a simple method, and in particular, we would like to emphasise that it must be seen as a proof of concept that exploring the topology of the subgraph around a transcript can give some hints about its confidence level, quality, assembly hardness, etc. The method proposed is not a full-fledged one for assessing transcripts in a de novo context. It could however be a promising direction to improve transcriptome assembly evaluation, especially when combined with statistical and read-mapping approaches (e.g. Rsem-Eval [4] or TransRate [5]).

As concerns the local transcriptome assembly of variations, the most interesting open problem which currently remains is how to efficiently enumerate AS events whose variable region (e.g. skipped exon, retained intron) traverses repeats. Although the application of the proposed model enabled to retrieve several AS events that were previously missed, the current algorithm is still only able to *avoid* repeats, not to solve them. The presence of repeats in RNA-seq data shows that transcriptome assemblers should formally address the repeats issue, as is generally the case of genome assemblers, preferably by solving them. Even if repeats are less frequent in transcriptomic data and are thus easier to solve than in the genomic context, the complexity and ambiguity they add are enough to cause problems if not addressed properly. If this is not done, it will impact the assembly of full-length transcripts or variants, leading to either erroneous or fragmented ones, especially in regions that are more prone to contain repeats, such as introns, UTRs, and exonised repeats.

As concerns future works, our repeats model could be improved. One direction would be to employ a tree-like structure to take into account the evolutionary nature of repeat (sub)families. Variability in the sizes of the copies of a repeat family would also enable to model more realistically the true nature of families of transposable elements (the type of repeats which cause most trouble in assembly). Another example would be to explicitly model sequencing errors in Theorem 1. Although, in practice, assemblers like KisSplice [1] employ a sequencing error removal module, it remains unclear how to distinguish the structures created by sequencing errors from the ones induced by a lowly-expressed member of a highly-expressed family of repeats, or by infrequent allelic differences in pool-seq. The difficulty increases in regions that are highly expressed or that present sequencing error bias. In practice, error removal strategies may be too stringent and erroneously remove SNPs and repeats. Taking into account the sequencing errors in the model would make it applicable even without any pre-processing of the data, and would thus be more sensitive for finding repeats if such errors are correctly modeled. Finally, the Branching Measure could also be extended to identify truncated transcripts and isoforms stemming from paralogous genes.

Authors' contributions

BS, GS, MFS and VL developed the model for repeats. LL, BS, GS, MFS developed and implemented the algorithms. LL, HLM, CM, VM performed the experiments. Part of this material appeared in WABI 2014. All authors read and approved the manuscript.

Author details

[1] Inria Grenoble, 655, Avenue de l'Europe, 38334 Montbonnot, France. [2] CNRS, UMR5558, Université Claude Bernard Lyon 1, 43, Boulevard du 11 Novembre 1918, 69622 Villeurbanne, France. [3] IRISA Inria Rennes Bretagne Atlantique; GenScale Team, Université Rennes 1, 263, Avenue Général Leclerc, 35042 Rennes, France.

Acknowledgements

LL acknowledges CNPq/Brazil for the financial support. This work was performed using the computing facilities of the CC LBBE/PRABI.

Competing interests

The authors declare that they have no competing interests.

Funding

LL was funded by the Brazilian Ministry of Science, Technology and Innovation (in portuguese, Ministério da Ciência, Tecnologia e Inovação - MCTI) through the National Counsel of Technological and Scientific Development (in portuguese, Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq), under the Science Without Borders (in portuguese, Ciências Sem Fronteiras) scholarship grant process number 203362/2014-4. VL was funded by the Agence Nationale de la Recherche ABS4NGS ANR project (ANR-11-BINF-0001-06) and Action n3.6 Plan Cancer 2009–2013. This work was also funded by the Agence Nationale de la Recherche ANR-12-BS02-0008 (Colib'read) with grants to LL, BS, GS, HL-M, CM, M-FS, and VL. This work was also funded by the European Research Council under the European Community's Seventh Framework Programme (FP7 /2007–2013)/ERC Grant Agreement No. [247073]10. with grants to BS, GS, M-FS, and VL.

Received: 27 July 2016 Accepted: 27 January 2017
Published online: 22 February 2017

References

1. Sacomoto G, Lacroix V, Sagot M-F. A polynomial delay algorithm for the enumeration of bubbles with length constraints in directed graphs and its application to the detection of alternative splicing in RNA-seq data. In: WABI, pp. 99–111 (2013).
2. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q, Chen Z, Mauceli E, Hacohen N, Gnirke A, Rhind N, di Palma F, Birren BW, Nusbaum C, Lindblad-Toh K, Friedman

N, Regev A. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol. 2011;29(7):644–52.
3. Schulz M, Zerbino D, Vingron M, Birney E. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics. 2012;28(8):1086–92.
4. Li B, Fillmore N, Bai Y, Collins M, Thomson J, Stewart R, Dewey C. Evaluation of de novo transcriptome assemblies from RNA-Seq data. Genome Biol. 2014;15(12):553.
5. Smith-Unna R, Boursnell C, Patro R, Hibberd J, Kelly S. TransRate: reference free quality assessment of de novo transcriptome assemblies. Genome Res. 2016;26(8):1134–44.
6. Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KHJ, Remington KA, Anson EL, Bolanos RA, Chou H-H, Jordan CM, Halpern AL, Lonardi S, Beasley EM, Brandon RC, Chen L, Dunn PJ, Lai Z, Liang Y, Nusskern DR, Zhan M, Zhang Q, Zheng X, Rubin GM, Adams MD, Venter JC. A whole-genome assembly of *Drosophila*. Science. 2000;287(5461):2196–204.
7. Novák P, Neumann P, Macas J. Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. BMC Bioinform. 2010;11(1):378.
8. Djebali S, Davis CA, Merkel A, Dobin A, Lassmann T, Mortazavi A, Tanzer A, Lagarde J, Lin W, Schlesinger F, Xue C, Marinov GK, Khatun J, Williams BA, Zaleski C, Rozowsky J, Röder M, Kokocinski F, Abdelhamid RF, Alioto T, Antoshechkin I, Baer MT, Bar NS, Batut P, Bell K, Bell I, Chakrabortty S, Chen X, Chrast J, Curado J, Derrien T, Drenkow J, Dumais E, Dumais J, Duttagupta R, Falconnet E, Fastuca M, Fejes-Toth K, Ferreira P, Foissac S, Fullwood MJ, Gao H, Gonzalez D, Gordon A, Gunawardena H, Howald C, Jha S, Johnson R, Kapranov P, King B, Kingswood C, Luo OJ, Park E, Persaud K, Preall JB, Ribeca P, Risk B, Robyr D, Sammeth M, Schaffer L, See L-HH, Shahab A, Skancke J, Suzuki AMM, Takahashi H, Tilgner H, Trout D, Walters N, Wang H, Wrobel J, Yu Y, Ruan X, Hayashizaki Y, Harrow J, Gerstein M, Hubbard T, Reymond A, Antonarakis SE, Hannon G, Giddings MC, Ruan Y, Wold B, Carninci P, Guigó R, Gingeras TR. Landscape of transcription in human cells. Nature. 2012;489(7414):101–8.
9. Tilgner H, Knowles D, Johnson R, Davis C, Chakrabortty S, Djebali S, Curado JA, Snyder M, Gingeras T, Guigó R. Deep sequencing of subcellular RNA fractions shows splicing to be predominantly co-transcriptional in the human genome but inefficient for lncRNAs. Genome Res. 2012;22:1616–25.
10. Robertson G, Schein J, Chiu R, Corbett R, Field M, Jackman SD, Mungall K, Lee S, Okada HM, Qian JQ, Griffith M, Raymond A, Thiessen N, Cezard T, Butterfield YS, Newsome R, Chan SK, She R, Varhol R, Kamoh B, Prabhu A-L, Tam A, Zhao Y, Moore RA, Hirst M, Marra MA, Jones SJM, Hoodless PA, Birol I. De novo assembly and analysis of RNA-seq data. Nat Methods. 2010;7(11):909–12.
11. Peng Y, Leung H, Yiu S, Lv M, Zhu X, Chin F. IDBA-tran: a more robust de novo de Bruijn graph assembler for transcriptomes with uneven expression levels. Bioinformatics. 2013;29(13):i326–34.
12. Sacomoto G, Kielbassa J, Chikhi R, Uricaru R, Antoniou P, Sagot M-F, Peterlongo P, Lacroix V. KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. BMC Bioinform. 2012;13(S–6):5.
13. Bao W, Kojima KK, Kohany O. Repbase update, a database of repetitive elements in eukaryotic genomes. Mobile DNA. 2015;6(1):11.
14. Lopez-Maestre H, Brinza L, Marchet C, Kielbassa J, Bastien S, Boutigny M, Monnin D, El Filali A, Carareto CM, Vieira C, et al. SNP calling from RNA-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. Nucl Acids Res. 2016;44(19):148.
15. Carroll ML, Roy-Engel AM, Nguyen SV, Salem A-H, Vogel E, Vincent B, Myers J, Ahmad Z, Nguyen L, Sammarco M, Watkins WS, Henke J, Makalowski W, Jorde LB, Deininger PL, Batzer MA. Large-scale analysis of the Alu Ya5 and Yb8 subfamilies and their contribution to human genomic diversity. J Mol Biol. 2001;311(1):17–40.
16. Jurka J, Bao W, Kojima K. Families of transposable elements, population structure and the origin of species. Biol Direct. 2011;6(1):44.
17. Bern M, Plassmann P. The steiner problem with edge lengths 1 and 2. Inf Process Lett. 1989;32(4):171–6.
18. Lappalainen T, Sammeth M, Friedlander MR. /'t Hoen, PAC, Monlong J, Rivas MA, Gonzalez-Porta M, Kurbatova N, Griebel T, Ferreira PG, Barann M, Wieland T, Greger L, van Iterson M, Almlof J, Ribeca P, Pulyakhina I, Esser D, Giger T, Tikhonov A, Sultan M, Bertier G, MacArthur DG, Lek M, Lizano E, Buermans HPJ, Padioleau I, Schwarzmayr T, Karlberg O, Ongen H, Kilpinen H, Beltran S, Gut M, Kahlem K, Amstislavskiy V, Stegle O, Pirinen M, Montgomery SB, Donnelly P, McCarthy MI, Flicek P, Strom TM, Consortium TG, Lehrach H, Schreiber S, Sudbrak R, Carracedo A, Antonarakis SE, Hasler R, Syvanen A-C, van Ommen G-J, Brazma A, Meitinger T, Rosenstiel P, Guigo R, Gut IG, Estivill X, Dermitzakis ET. Transcriptome and genome sequencing uncovers functional variation in humans. Nature. 2013;501(7468):506–11.
19. Robinson JT, Thorvaldsdottir H, Winckler W, Guttman M, Lander ES, Getz G, Mesirov JP. Integrative genomics viewer. Nat Biotechnol. 2011;29(1):24–6.
20. Griebel T, Zacher B, Ribeca P, Raineri E, Lacroix V, Guigó R, Sammeth M. Modelling and simulating generic RNA-seq experiments with the flux simulator. Nucl Acids Res. 2012;40(20):10073.
21. Kent WJ. BLAT—the BLAST-like alignment tool. Genome Res. 2002;12:656–64.
22. Freyermuth F, Rau F, Kokunai Y, Linke T, Sellier C, Nakamori M, Kino Y, Arandel L, Jollet A, Thibault C, Philipps M, Vicaire S, Jost B, Udd B, Day JW, Duboc D, Wahbi K, Matsumura T, Fujimura H, Mochizuki H, Deryckere F, Kimura T, Nukina N, Ishiura S, Lacroix V, Campan-Fournier A, Navratil V, Chautard E, Auboeuf D, Horie M, Imoto K, Lee K-Y, Swanson MS, de Munain AL, Inada S, Itoh H, Nakazawa K, Ashihara T, Wang E, Zimmer T, Furling D, Takahashi MP, Charlet-Berguerand N. Splicing misregulation of SCN5A contributes to cardiac-conduction delay and heart arrhythmia in myotonic dystrophy. Nat Commun. 2016;7:11067.

Chapter 4

Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data

Preamble

Key points

- Tools for *de novo* assembly of alternative splicing events allow to discover novel variants even when a good reference genome and annotations are available;
- Assembly-first approaches can better detect events that: i) involve novel exons or novel combinations of existing exons; ii) stem from paralogous genes;
- Mapping-first approaches can better detect events that: i) are lowly-expressed; ii) contain repeats; iii) are complex;
- No approach is exhaustive. Mapping-first and assembly-first approaches each systematically misses some types of alternative splicing events. Furthermore, many events that were found only by one approach are differentially regulated across conditions. As such, these events should not be discarded from the analysis. Therefore, the diversity of splicing variants can be better explored by a combination of both approaches;
- Better methods are needed for the detection of events that: i) are lowly-expressed; ii) are exonized repeats; iii) are complex splicing variants; iv) stem from paralogous genes.

Status

Published in journal *Scientific Reports* [11].

Author contributions

The first author is *Clara Benoit-Pilven*. *L.* is second author and was involved in the tasks of:

- Improving the scalability of KisSplice;
- Comparing the two pipelines and classifying the instance types;
- Developing the supporting webpage.

SCIENTIFIC REPORTS

OPEN

Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data

Received: 15 September 2017
Accepted: 30 January 2018
Published online: 09 March 2018

Clara Benoit-Pilven[1], Camille Marchet[3], Emilie Chautard[1,2], Leandro Lima[2], Marie-Pierre Lambert[1], Gustavo Sacomoto[2], Amandine Rey[1], Audric Cologne[2], Sophie Terrone[1], Louis Dulaurier[1], Jean-Baptiste Claude[1], Cyril F. Bourgeois[1], Didier Auboeuf[1] & Vincent Lacroix[2]

Genome-wide analyses estimate that more than 90% of multi exonic human genes produce at least two transcripts through alternative splicing (AS). Various bioinformatics methods are available to analyze AS from RNAseq data. Most methods start by mapping the reads to an annotated reference genome, but some start by a *de novo* assembly of the reads. In this paper, we present a systematic comparison of a mapping-first approach (FARLINE) and an assembly-first approach (KISSPLICE). We applied these methods to two independent RNAseq datasets and found that the predictions of the two pipelines overlapped (70% of exon skipping events were common), but with noticeable differences. The assembly-first approach allowed to find more novel variants, including novel unannotated exons and splice sites. It also predicted AS in recently duplicated genes. The mapping-first approach allowed to find more lowly expressed splicing variants, and splice variants overlapping repeats. This work demonstrates that annotating AS with a single approach leads to missing out a large number of candidates, many of which are differentially regulated across conditions and can be validated experimentally. We therefore advocate for the combined use of both mapping-first and assembly-first approaches for the annotation and differential analysis of AS from RNAseq datasets.

In the last 10 years, the prevalence of alternative splicing has been completely re-evaluated. Recent reports claim that more than 90% of multi-exon genes produce at least two splicing variants[1,2]. The depth at which transcriptomes can be sampled with next generation sequencing techniques opens the possibility not only to annotate splicing variants in various conditions, but also to detect which transcripts are differentially spliced across pathological and physiological conditions.

This growing interest in splicing both as a fundamental process and because of its implication in pathologies[3–5] has been accompanied by an increasing number of methods aiming at analyzing RNAseq datasets[6–8]. The ultimate goal of these methods is to identify and quantify full-length transcripts from short sequencing reads. This task is particularly challenging and recent benchmarks show that all methods still make a lot of mistakes[9]. The difficulty of reconstructing full-length transcripts (isoform-centric approaches) also prompted a number of authors to focus on identifying exons that are differentially included within transcripts (exon-centric approaches)[10–13].

Whether they are exon- or isoform-centric, methods to study splicing from RNAseq data can further be divided in two main categories[14]. The mapping-first approaches first map the reads to the reference genome and

[1]Université de Lyon, ENS de Lyon, Université Claude Bernard, CNRS UMR 5239, INSERM U1210, Laboratory of Biology and Modelling of the Cell, 46 Allée d'Italie Site Jacques Monod, F-69007, Lyon, France. [2]Université de Lyon, F-69000, Lyon; Université Lyon 1; CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, F-69622, Villeurbanne, EPI ERABLE - Inria Grenoble, Rhône-Alpes, France. [3]IRISA Inria Rennes Bretagne Atlantique CNRS UMR 6074, Université Rennes 1, GenScale team, Rennes, 263 Avenue Général Leclerc, Rennes, France. Correspondence and requests for materials should be addressed to D.A. (email: Didier.auboeuf@inserm.fr) or V.L. (email: Vincent.lacroix@univ-lyon1.fr)

the mapped reads are then assembled into exons and eventually transcripts. In contrast, assembly-first approaches first assemble the reads based on their overlaps. The assembled sequences (corresponding to sets of exons) are then aligned to the reference genome.

Mapping-first approaches have been the most used so far, essentially because they were the first to be developed and because they initially required less computational resources. *De novo* assembly methods were also thought to be restricted to non-model species, where no (good) reference genome is available, and they seemed to be inadequate when an annotated reference genome is available.

Recent progress in *de novo* transcriptome assembly is clearly changing this view, and the argument of the heavier computational burden does not hold anymore.

The application of *de novo* assembly to human RNAseq datasets however still remains rare, although some studies have already shown its potential to detect novel biologically relevant splicing variants[15,16].

The generalization of *de novo* assembly approaches for studying splicing in human seems to be mostly impeded by the lack of a clear evaluation of its potential interest in comparison to more traditional mapping-based approaches.

This is the gap we aim at filling with the work presented here.

To achieve this goal, we performed a systematic evaluation of an assembly-first and a mapping-first approach on two RNAseq datasets.

As a first step, we compared pipelines that we developed in parallel, namely KisSplice and FaRLine, because we could easily control their parameters. Any difference between the predictions that is solely due to a parameter setting could be fixed easily, which enabled us to obtain a precise understanding of the irreducible differences between the two approaches.

In a second step, we confirmed the generality of our findings by benchmarking our methods against Cufflinks[6], MISO[11] and Trinity[17], which are widely used pipelines.

A significant part of our work has been to manually dissect a number of cases found by only one of the two methods. This enabled us to go beyond a simple qualitative description and provide the community with a precise understanding of which cases are overlooked by each type of method, and where new methods are needed.

All the software and step-by-step protocols presented in this work are freely available at http://kissplice.prabi.fr/pipeline_ks_farline. This should facilitate the reproducibility of our work, and applications to other datasets.

From a general point of view, the combination of approaches we propose should enable to improve splicing-related transcriptomic analyses in physiological and pathological situations.

Results

KisSplice and FaRLine. Figure 1 presents schematically the two pipelines that we developed and compared. A detailed description of each step is given in the Methods section. In the assembly-first approach, a De Bruijn graph is built from the reads. Alternative splicing events, which correspond to bubbles in this graph are enumerated and quantified by KisSplice. Each path is then mapped on the reference genome using STAR and the event is annotated by KisSplice2RefGenome, using the EnsEMBL r75 annotations as an evidence. Importantly, exons not present in the annotations can be identified by this approach. In the mapping-first approach, reads are aligned to the reference genome using TopHat2. Mapped reads are then analyzed by FaRLine, using the EnsEMBL r75 annotations as a guide.

We also tested STAR instead of TopHat2 for the mapping-first pipeline, and found that our main results were essentially unchanged (see Methods).

Quantification of splicing variation is performed similarly in the two pipelines. Only junction reads are considered. Exonic reads are not considered, for reasons exposed in Methods. For the inclusion isoform, there are two junctions to consider. We calculate the mean of the counts of these two junctions.

The differential analysis is performed by a common method for the two approaches: kissDE, which tests if the relative abundance of the inclusion isoform has changed significantly across conditions.

Overall, we developed and adapted jointly these two pipelines in order to minimize the discrepancies that could complicate the comparison.

The majority of frequent isoforms are identified by both approaches. Applying KisSplice and FaRLine to the same RNAseq datasets generated by the ENCODE consortium (SK-N-SH cell lines treated or not with retinoic acid), we noticed that 68% of the alternatively skipped exons (ASE) identified by KisSplice were also identified by FaRLine and that 24% of ASEs identified by FaRLine were also identified by KisSplice (Fig. 2A). This observation highlights that the mapping-first approach predicts a much larger number of events. This difference in sensitivity is due to the fact that while mapping-first approaches require that each exon junction is covered by at least one read, assembly-first approaches require overlapping reads across the entire skipped exon. Therefore, it can be anticipated that low abundant isoforms, that are covered by few reads, will be reported by mapping, but not by the assembly-first approach. Supporting this prediction, we observed that for ASEs reported only by FaRLine, the number of reads supporting the minor isoform is much lower than in the other categories (Fig. 2 B). The same results were obtained using another RNAseq dataset representing MCF-7 cells expressing or not the DDX5 and DDX17 splicing factors (Supplementary Figure S1).

Having clarified that rare variants are better handled by the mapping-first approach, we decided to filter them out, in order to analyse other differences between the two approaches. Experimental validations by RT-PCR that we performed on rare variants stratified by read support enabled us to clarify that both an absolute and a relative cutoff on the number of reads are required to discriminate variants which can be validated from those which cannot. Indeed, out of the 48 tested cases, we were able to validate 41 (Supplementary Figure S9). The non validated cases indeed corresponded to cases supported by fewer reads. However, what really departed them from the validated cases was their lower relative abundance (Supplementary Figure S10, Supplementary Table 1). In the

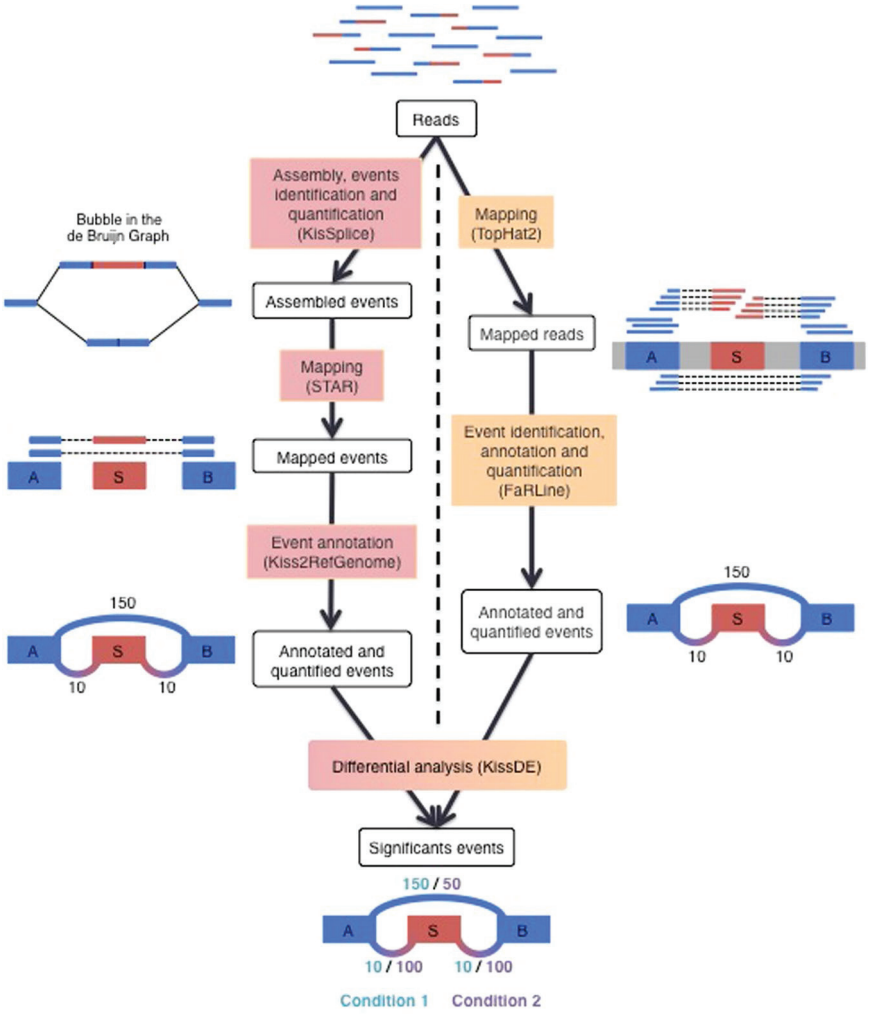


Figure 1. The two pipelines compared in this study: KisSplice and FaRLine. The first step of KisSplice is to assemble the reads and extract the splicing events. These events are then mapped back to the reference genome and classified by event type. The annotated and quantified events are then used for the differential analysis between the biological conditions. In contrast, the first step of FaRLine is to map the reads on the reference genome. From this mapping, annotated and quantified events are extracted. Finally, the differential analysis is done with the same method as in the KisSplice pipeline.

remaining of our work, we chose to use both criteria and we filtered variants supported by less than 5 reads, and less than 10% compared to the major isoform.

As expected, the proportion of candidates reported simultaneously by both methods increased significantly. Approximately 70% of predicted skipped exons were indeed found by both approaches after filtering lowly expressed isoforms. (Fig. 2C, Supplementary Figure S1C).

Furthermore, the estimation of their inclusion rates was consistent across the two approaches ($R^2 > 0.9$)).

Beyond the overall concordance of the two approaches in detecting common splicing events, a number of candidates remained reported by only one approach. Since many of them have a highly-expressed minor isoform (supported by more than 100 reads) (Fig. 2D, Supplementary S1D), the failure of one approach to detect them is likely not due to a lack of coverage.

For events only found by one approach, we patiently dissected the reasons why they could have been missed out by the other approach. This enabled us to define 4 main categories which cover 70% of the cases (Fig. 3A) The remaining 30% of cases did not fit into clearly defined biological categories. We however classified them using methodological criteria. The full list of categories is presented in Supplementary Table 2. For each of the 4 main categories, we selected cases to validate experimentally. All 34 RT-PCR validations were successful and are presented in Supplementary Figure S11 confirming that these events are not false positives.

Some isoforms are systematically missed by one approach. The first category corresponds to cases that were missed out by the mapping-first approach and corresponds to alternative splicing events involving novel exons or novel combinations of existing exons.

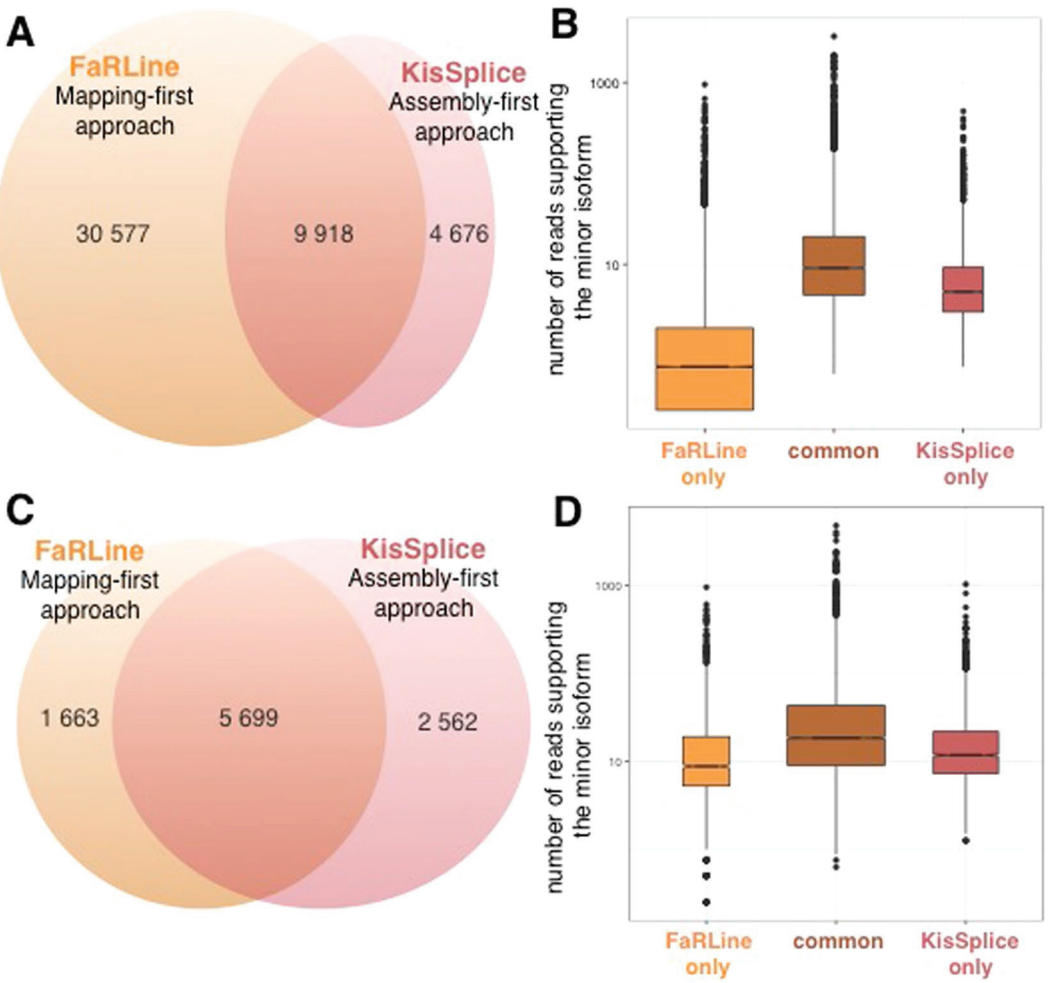


Figure 2. Comparison of the ASE identified by the assembly-first and mapping-first pipelines. (**A**) Venn diagram of ASEs identified by the two pipelines. FaRLine detected many more events than KisSplice. 68% of ASE found by KisSplice were also found by FaRLine and 24% of ASE detected by FaRLine were also found by KisSplice. (**B**) Boxplot of the expression of the minor isoform in the 3 categories defined in the Venn diagram of panel A: ASE identified only by FaRLine, ASE identified by both pipelines and ASE identified only by KisSplice. The number of reads supporting the minor isoform of the ASE identified by FaRLine is overall much lower. Many isoforms are supported by less than 5 reads. (**C**) Venn diagram of ASEs identified by the two pipelines after filtering out the poorly expressed isoforms (less than 5 reads, or less than 10% of the number of reads supporting both isoforms). The common events represent a larger proportion than before filtering: 77% of the ASE identified by FaRLine and 69% of the ASE identified by KisSplice. (**D**) Boxplot of the expression of the minor isoform in the 3 categories defined in the Venn diagram of panel C: ASE identified only by FaRLine, ASE identified by both pipelines and ASE identified only by KisSplice. The distribution of the number of reads supporting the minor isoform is similar for the 3 categories with highly expressed variants in each category.

There are two reasons to explain why the mapping-first approach does not detect these events. First the mapper may fail to map the reads, or map them to an incorrect location, as junction discovery using short reads is a challenging task. Second, even in the case where the mapper succeeds, FaRLine may fail to report the event because it relies on annotations. Among these 1864 cases, we distinguished 3 sub-categories of errors due to the annotation. Either the exon is unannotated (30%), one of its flanking exon is unannotated (13%) or both exons are annotated but no transcript combining them was annotated (57%).

The assembly-first approach, KisSplice, does not consider annotations, and an interesting resulting advantage is that novel junctions have the same chance to be assembled as known junctions. Mapping assembled novel junctions to the genome is indeed less challenging than read mapping because the assembled sequences are longer.

More importantly, the ability of KisSplice to identify novel splicing events comes from the fact that it introduces known annotations as late as possible in its pipeline (see Methods). Annotations are used as an evidence, not as a filter. AS events involving novel splice sites are clearly identified as such, and can be specifically tested and experimentally validated. More than 99% of the novel splice sites were canonical splice sites (GT-AG).

As an example, the *HIRA* gene contains a novel exon, whose inclusion is supported by at least 20 reads on each junction (Fig. 3B, Supplementary Figure S8A). This case was overseen by the mapping-first approach, FaRLine.

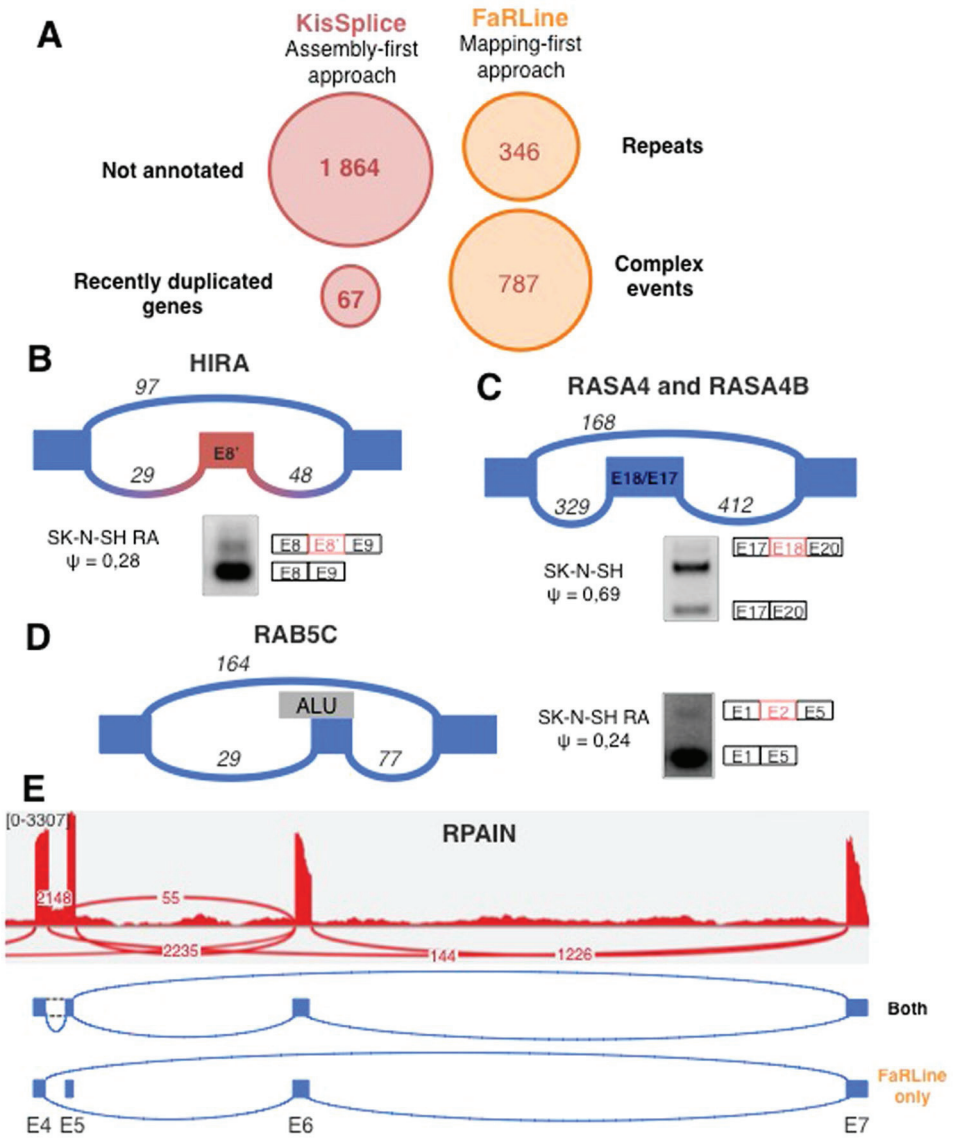


Figure 3. (**A**) Main categories explaining why some exons are detected by only one method. (**B**) The exon in intron 8 of the *HIRA* gene is an example of an exon not annotated in EnsEMBL r75. This event was identified by KisSplice but not by FaRLine. (**C**) *RASA4* and *RASA4B* are 2 paralog genes. KisSplice detected 2 isoforms that could be produced by these 2 genes. FaRLine did not detect any event in either of these genes. The exon skipped is exon 18 in *RASA4* (corresponding to exon 17 in *RASA4B*). The third band on the RT-PCR is the inclusion of another exon in the intron 18 of *RASA4*. (**D**) Exon 2 of the *RAB5C* gene is an example of exon skipping overlapping an Alu element identified only by FaRLine. The events in panel B to C were validated by RT-PCR. (**E**) The *RPAIN* gene contains a complex event with a lowly expressed isoform. This weakly expressed isoform was not identified by KisSplice, while the other isoforms were identified by both approaches.

The panel B of the Supplementary Figure S8 shows an example of an ASE not reported by FaRLine because the included exon was not present in the transcripts.

The second category of splicing events identified by only one approach corresponds to recent gene duplications. Untangling the relation between alternative splicing and gene duplication is a difficult topic, subject to debate[18,19]. It is indeed difficult to assess the amount of alternative splicing that occurs within paralogous genes. With the mapping-first approach, the reads stemming from recent paralogs are classified as multi-mapping reads. FaRLine, like the vast majority of mapping-first pipelines, discards these reads for further analysis, as their precise location cannot be clearly established. This results in silently underestimating alternative splicing in recent paralog genes. Note that setting the mapper to keep multi-mapping reads in the analysis leads to overestimating alternative splicing, as all members of the family will be predicted as alternatively spliced. In opposition, *de novo* assembly can faithfully state that a family of recent paralogs collectively produce two isoforms that vary in their sequence. However, whether the two isoforms are produced from the same locus or from different loci remains undetermined. KisSplice detects these cases of putative AS in paralog genes. Figure 3C illustrates the case with

genes *RASA4* and *RASA4B*. Exon 18 in *RASA4* (denoted as exon 17 in *RASA4B*) was detected to be skipped. The exclusion isoform is supported by 160 reads, while the inclusion isoform is supported by 400 reads. The mapping-first approach did not detect either of these isoforms at all. Another example from this category is presented in Supplementary Figure S2C.

The third category of splicing events identified by only one approach corresponds to cases that are missed out by the assembly-first approach. Out of the 1663 cases belonging to this category, a large fraction (21%) corresponds to cases where the skipped exon overlaps a repeat, notably Alu elements. Alu are transposable elements present in a very large number of copies in the human genome[20]. Most of these copies are located in introns and a number of them have been exonised[21,22]. The reason why the mapping-first approach is able to identify these cases is because even though the reads partially map to repeated sequences, the boundaries of these exons are unique and annotated. Hence the mapper, if set properly, can map these reads to unique annotated exon junctions and is not confused by multiple mappings. Importantly, if the annotations are not provided to the mapper, it will be confused by multiple mappings and will not be able to map the read to the correct location (Supplementary Figure S7). Figure 3D and Supplementary Figure S2D represent two RT-PCR validated Alu-derived exons identified by the mapping-first approach. The assembly-based approach fails to detect most of these events. The reason is that, although they do form bubbles in the DBG generated by the reads, these bubbles are highly branching (supplementary figure http://kissplice.prabi.fr/sknsh/graph_RAB5C_distance_3.html23). Enumerating branching bubbles is computationally very challenging, and may take a prohibitive amount of time. In practice, we restrict our search to the enumeration of bubbles with at most 5 branches (Supplementary Figure S12A).

The fourth category of splicing events identified by only one approach corresponds to cases where more than two splicing isoforms locally coexist, and one of them is poorly expressed compared to the others. The *RPAIN* gene is a good illustration of such cases (Fig. 3E), as exons 5 and 6 of *RPAIN* may be skipped and the intron between exons 4 and 5 may be retained. While both methods successfully reported the skipping of exon 6, with exons 5 and 7 as flanking, FaRLine additionally reported the skipping of the same exon, but with exons 4 and 7 as flanking exons. The reason why KisSplice did not report this case is because the junction between exons 4 and 6 is relatively weakly supported. More specifically, this junction is supported by only 55 reads, which accounts for less than 2% of the total number of reads branching out from exon 4. Transcriptome assemblers, like KisSplice, usually interpret such relatively weakly supported junctions as sequencing errors or spurious junctions in highly-expressed genes, therefore disregarding them in the assembly phase (see Supplementary Methods). Supplementary Figure S2E shows another example of a complex event not correctly handled by KisSplice because there were locally more than 5 branches.

Comparison of the approaches after differential analysis. Beyond the tasks of identifying exon skipping events, a natural question which arises when two conditions are compared is to assess if the exon inclusion rate significantly changed across conditions.

In order to test this, we took advantage of the availability of replicates for both the SK-N-SH cell line and the same cell line treated with retinoic acid. For each detected event, we tested with kissDE[24], whether we could detect a significant association between one isoform and one condition. Focusing on those condition-specific events, we again partitioned them in events reported by both methods, by FaRLine only and by KisSplice only. As shown in Fig. 4, the majority of condition-specific events were detected by both approaches. This is the case for instance of exon 22 of gene *ADD3* which is clearly more included upon retinoic acid treatment (Fig. 4C), with a DeltaPSI of 27%. The estimation of the DeltaPSI is overall very similar across the two approaches (Fig. 4B) with a correlation of 0.94. The outliers essentially correspond to ASE with several alternative donor/acceptor sites. KisSplice considers these events as different exons while FaRLine considers them as an unique exon, and sums up all the incoming (resp. outgoing) junction counts. Hence, the read counts will differ. Supplementary Figure S8D gives an example.

When focusing on condition-specific events, the proportion of events predicted by only one method increased, for two main reasons. First, some ASE annotated by both approaches were predicted to be differentially included only by one method. This is again due to differences in the quantification of the inclusion rate, especially for ASE with multiple 5′ and 3′ splice sites. Second, some of the exons that were missed out by one method at the identification step happened to be condition specific. This is the case of an exon in *NINL* intron 5 (Fig. 4D), only identified by KisSplice because it was not annotated. This is also the case of *SAR1B* exon 3 (Fig. 4E), only identified by FaRLine because it overlaps with an Alu element. The analysis of the MCF-7 RNAseq dataset gave very similar results (Supplementary Figure S3).

The observation that many of the AS events that were annotated only by one method are differentially regulated across conditions confirms that these AS events should not be discarded from the analysis. Focusing only on AS events annotated by one approach may lead to miss splicing events which are central in the biological context.

Overlap with other methods. In a first step, we picked FaRLine and KisSplice as examples of a mapping-first and an assembly-first approach respectively. Clearly, there are other published methods in both categories. MISO is probably the most widely used to annotate AS events. We therefore ran it on the same datasets to check how its predictions overlapped with ours. As shown in Fig. 5A (SK-N-SH dataset), 77% of predictions made by MISO were common to both FaRLine and KisSplice, 18% were only common with FaRLine, 2% were only common to KisSplice and the remaining 3% were specific to MISO. The overlap between the different methods was very similar when the MCF-7 RNAseq dataset was used (Supplementary Figure S4A). Overall, almost all candidates predicted by MISO were also predicted by FaRLine. This large overlap with FaRLine was expected, because both are mapping-first approaches. This also shows that the differences between mapping- and assembly-first approaches reported above are not limited to one mapping-first approach.

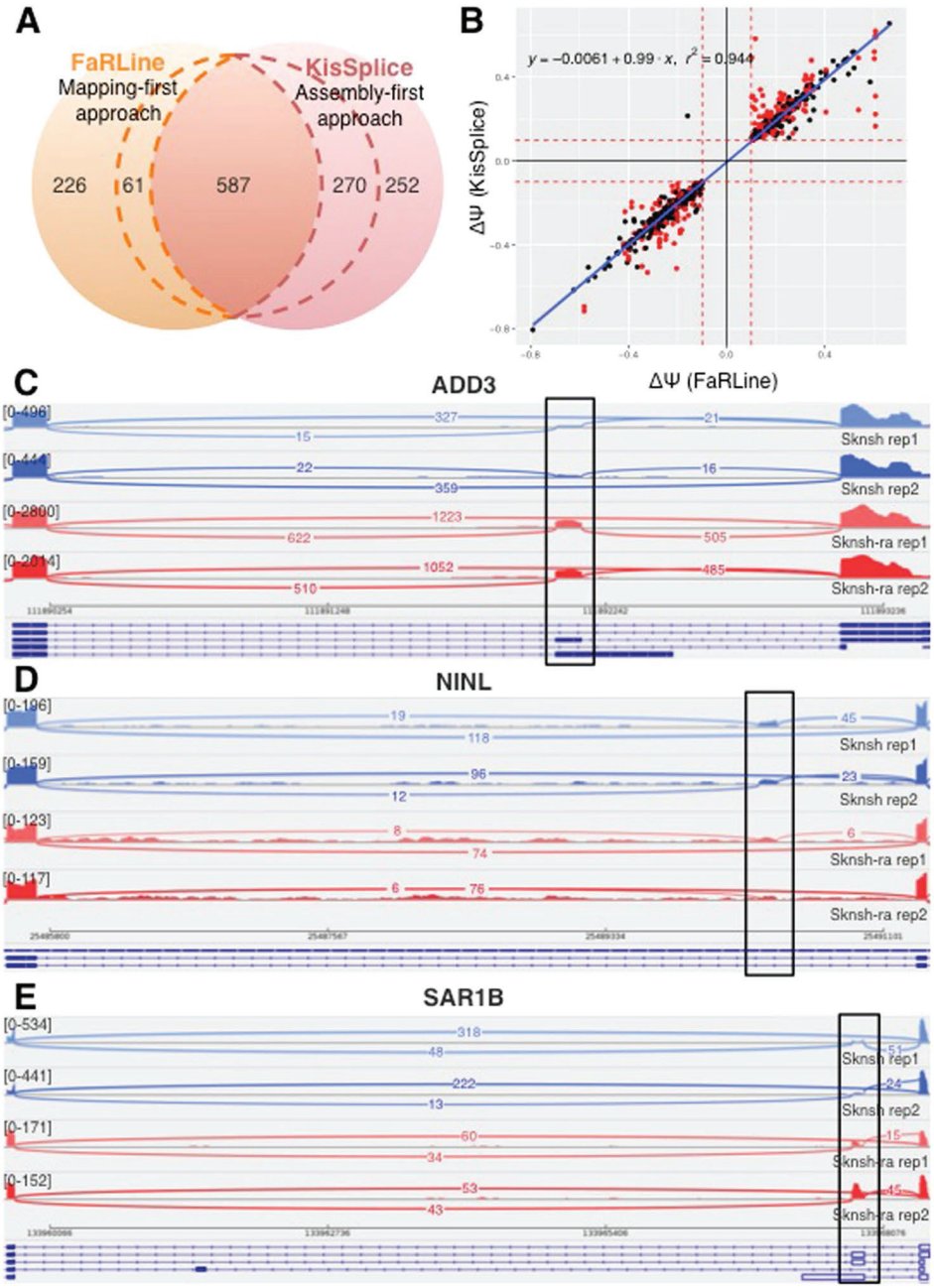


Figure 4. (**A**) Condition-specific variants identified by FaRLine, KisSplice or both methods. Within dashed lines are events identified by both approaches but detected as condition-specific by only one approach. (**B**) DeltaPSI as estimated by KisSplice and FaRLine, for events identified by both methods. The red dots represent complex events for which KisSplice found at least 2 'bubbles'. (**C**) Exon 22 of the *ADD3* gene is an example of regulated ASE identified by both approaches. (**D**) A new exon in intron 5 of *NINL* gene is identified only by KisSplice. The inclusion of this exon is differentially regulated between the 2 experimental conditions. (**E**) Because exon 3 of the *SAR1B* gene is an exonised Alu element, only FaRLine identified this event. Moreover this exon is significantly more included in the treated cells (SK-N-SH RA) compared to the control cells.

Besides exon-centric approaches, which aim at finding the differentially spliced exons, there is also a number of published methods which are isoform-centric and have the more ambitious goal to reconstruct full-length transcripts at the expense of underestimating alternative splicing.

The most widely used mapping-first and isoform-centric approach is Cufflinks[6] that we compared to FaRLine using the same dataset. As shown in Fig. 5B (and Supplementary Figure S4B), we found that the vast majority of ASE were predicted by both approaches.

Finally, we compared KisSplice to one of the most widely used de-novo transcriptome assembler, Trinity[17]. As shown in Fig. 5D (and Supplementary Figure S4D), most ASE found by Trinity were also found by KisSplice. However, KisSplice was significantly more sensitive. The goal of Trinity is to assemble the major isoforms

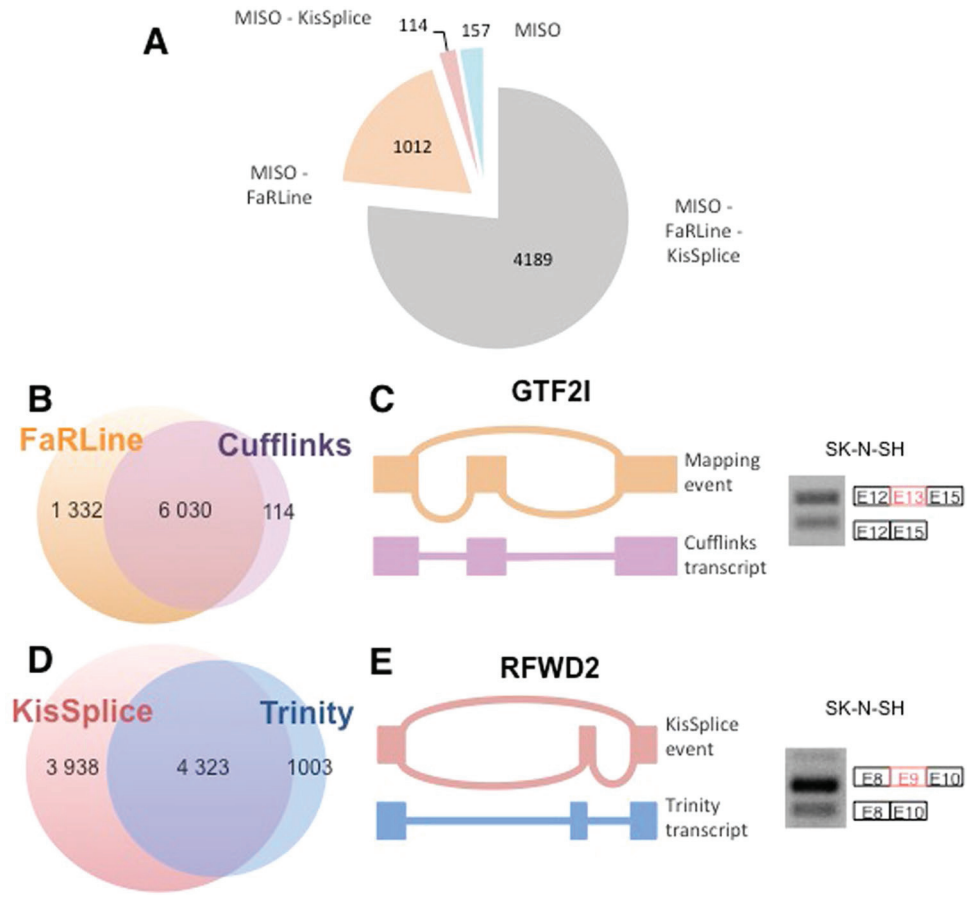


Figure 5. (**A**) 77% of ASE identifed by MISO are also annotated by FaRLine and KisSplice. 18% of MISO's ASE are also annotated by FaRLine while only 2% of MISO's ASE are also annotated by KisSplice. Finally, only 3% of these ASEs are only annotated by MISO. (**B**) Most of the events annotated by Cufflinks are identified by FaRLine. (**C**) *GTF2I* exon 13 is an example of an ASE annotated by FaRLine but not by Cufflinks. Indeed, Cufflinks only identified the isoform corresponding to the exon inclusion. (**D**) Most of the events annotated by Trinity are also annotated by KisSplice. But half of the ASE annotated by KisSplice are not annotated by the global assembler Trinity. (**E**) KisSplice annotates an ASE in the *RFWD2* gene, while Trinity only identified the isoform corresponding to the exon inclusion. The events in panels C and E have been validated by RT-PCR.

for each gene, it therefore largely under-estimates alternative splicing, especially inclusion/exclusion of short sequences.

For completeness sake, we also provide an all-vs-all comparison (Supplementary Figure S5). An interactive version of this Figure is available at http://kissplice.prabi.fr/pipeline_ks_farline/. The list of events found by any used method can be retrieved from this interactive figure and analysed in IGV, to reproduce the sashimi plots of the paper. The general conclusions from these comparisons is that there is a clear distinction between mapping-first and assembly-first approaches, and between exon-centric and isoform-centric approaches, the latter being less sensitive.

Discussion

De novo assembly is usually applied to non-model species where no (good) reference genome is available. We show here that even when an annotated reference genome is available, using assembly offers a number of advantages. We named this approach "assembly-first" because it does use a reference genome, but as late as possible in the process, in order to minimize the *a priori* on which exons should be identified.

Using this strategy, we identified novel alternatively skipped exons, which were not identified by traditional read mapping approaches (Fig. 3 and Supplementary Figure S2). While it is believed that the human genome is fully annotated, it is important to underline that we have not yet established a final map of the parts of the genome that can be expressed. It can be anticipated that sequencing of single-cells from different parts of the body will lead to the discovery of a huge diversity of transcripts and that a substantial number of new exons will be discovered. An example is the case of unannotated skipped exons which overlap with repeat elements. We cannot exclude that this category is currently largely under-annotated.

We also showed that assembly-first approach has the ability to detect splicing variants within recently duplicated genes (Fig. 3 and Supplementary Figure S2). This is because mapping approaches discard reads which map to multiple genomic locations. Identification of such splicing variants produced from different genomic regions

sharing sequence similarities (e.g. paralog genes, pseudogenes) is however very important, since splicing variants generated from paralogous genes but also from pseudogenes may have different biological functions[25].

Conversely, we showed that some ASE were detected only by the mapping-first approach. As shown in Fig. 2 (and Supplementary Figure S1), we observed that the mapping-first approach has a better ability to detect lowly-expressed splicing variants. Although such lowly-expressed splicing variants are often considered as "noise" or biologically non relevant, caution must be taken with such assumptions for several reasons. First, mRNA expression level is not necessarily correlated with protein expression level. Second, as observed from single-cell transcriptome analyses, some mRNAs can be expressed in few cells, within a cell population (e.g. they are expressed at a specific cell cycle step) and may therefore appear to be expressed at a low level in total RNAs extracted from a mixed cell population[26]. Therefore, computational analysis should not systematically discard lowly-expressed splicing variants and filtering these events should depend on the biological questions to be addressed.

We also observed that the mapping-first approach better detects exons corresponding to annotated-repeat elements (Fig. 3 and Supplementary Figure S2). While it has been assumed for a long time that repeat elements are "junk", increasing evidences support important biological functions for such elements. For example, repeat elements like Alu can evolve as exons and the presence of Alu exons in transcripts has been shown to play important regulatory functions[22,27].

When two methods give non-overlapping predictions, the temptation could be to focus on exons found by both approaches and to discard the others. We argue that this is not the best option, because approach-specific cases can be validated experimentally, and also because many of them correspond to regulated events, i.e. the inclusion isoform is significantly up or down regulated depending on the experimental condition.

In conclusion, combining mapping- and assembly-first approaches allows to detect a larger diversity of splicing variants. This is very important towards the in depth characterization of cellular transcriptome although other approaches are further required to analyze their biological functions.

From a computational perspective, a number of challenges are still ahead. The co-development of two approaches enabled us to narrow down the list of difficult instances not properly dealt with by at least one approach, but we cannot exclude that some categories are still missed out by both approaches. The categories of challenging cases that we defined in Fig. 3: lowly-expressed variants, exonised Alu, complex splicing variants, paralogs have been overlooked up to now. Possibly because they are much harder to detect, they have been assumed to play a minor role in transcriptomes, but more recent studies however argues the opposite.

For exonised ALUs, paralog genes and genes with complex splicing patterns, the possibility to sequence longer reads with third generation techniques[28,29] should prove very helpful. The number of reads obtained with these techniques is however currently much lower than with Illumina, thereby preventing their widespread use for differential splicing, for which the sequencing depth, and not so much the length of the reads, is the critical parameter which conditions the statistical power of the tests. In the coming years, methods combining second and third generation sequencing should enable to obtain significant advances in RNA splicing.

Material and Methods

FaRLine and KisSplice. Figure 1 shows the two pipelines that we are comparing. While STAR and TopHat are third-party softwares, we developed the other methods ourselves. KisSplice was first introduced in Sacomoto *et al.*[13]. The novelty here is that its usage is now possible in the case where a reference genome is available, which required specific methodological developments implemented in the newly released KisSplice2RefGenome software. kissDE was first introduced in Lopez-Maestre *et al.*[24] in the context of SNPs for non-model species. We present here its extension for alternative splicing. FaRLine is a new mapping-first pipeline, that we introduce in this paper. It is the RNAseq pipeline associated to the FasterDB database[30] and was already successfully applied to the analysis of the effect of metformin treatment on myotonic dystrophy type I (DM1) with a validation rate of 95%[31]. Specifically, 20 cases of ASE regulated by the metformin treatment were tested, and 19 were validated. In this paper, we provide additional validations of FaRLine with similar validation rates (36 out of 38), Supplementary Figure S19.

For the sake of self-containment, we explain all methods here.

KisSplice. KisSplice is a local transcriptome assembler. As most short reads transcriptome assemblers[8,17,32], it relies on a De Bruijn graph (DBG). Its originality lies in the fact that it does not try to assemble full-length transcripts. Instead, it assembles the parts of the transcripts where there is a variation in the exon content. By aiming at a simpler goal, it can afford to be more exhaustive and identify more splicing events. The key concept on which KisSplice is built is that variations in the nucleotide content of the transcripts will correspond to specific patterns in the DBG called bubbles (Supplementary Figure S13). KisSplice's main algorithmic step therefore consists in enumerating all the bubbles in the graph built from the reads. Examples of bubbles in the DBG and explanation of the parameters used to filter out sequencing errors and repeat-induced bubbles are given in Supplementary Methods.

Annotating the events with KisSplice2RefGenome. KisSplice outputs bubbles in the form of a pair of fasta sequences. Clearly, such information is insufficient to analyse alternative splicing for model species. KisSplice2RefGenome enables to provide for each bubble: the gene name, the AS event type, the genomic coordinates and the list of splice sites used (novel or annotated).

Bubbles found by KisSplice are mapped to the reference genome using STAR, with its default settings, which means that in the case of multi-mappings, STAR reports all equally best matches. The mapping results are then analysed by KisSplice2RefGenome. Bubbles are classified in sub-types depending on the number of blocks obtained when mapping each path of the bubble to the genome (Supplementary Figure S14). For exon skipping,

the longer path of the bubble corresponds to 3 blocks, while the lower path corresponds to 2 blocks. The splice sites are located and compared to the annotations. Events with novel splice sites are reported explicitly as such in the output of the program.

In the case where the bubble corresponds to a genomic insertion or deletion, it exhibits a specific pattern in terms of block numbers (one block for one path and two blocks for the other) and is reported separately.

The criterion of the number of blocks is discriminative in most cases. However, there is a possible confusion between intron retentions and genomic deletions, since in both cases, the longer path will map into one block and the lower path in two blocks. In this case, we also use the distance between the blocks, and introduce a user-defined threshold, which we set to 50nt, below which the bubble is classified as a genomic deletion, and above which it is classified as an intron retention.

In the special case where the exon flanking the AS event is very short (less than k nt), the number of blocks is increased for both paths, but the difference of number of blocks remains unchanged.

In the special case where there is a genomic polymorphism located less than k nt apart from the AS event, KisSplice will report several bubbles (possibly all combinations of genomic and transcriptomic variants). This redundancy is removed in KisSplice2RefGenome where the primary focus in on splicing.

In the case where the bubble maps to two locations on the genome, a distinction is made between the case of exact repeats where both paths map to both locations and inexact repeats where each path maps to a distinct location (Supplementary Figure S12B). The cases of exact repeats correspond to recent gene duplications.

FaRLine. FasterDB EnsEMBL r75 annotation. FasterDB RNAseq Pipeline, FaRLine, uses the FasterDB-based EnsEMBL r75 annotation database. FasterDB is a database containing all annotated human splicing variants[30].

Each transcripts present in the FasterDB, is composed of a succession of exons, that we call transcript exons (represented in blue in Supplementary Figure S15). The genomic exons (represented in red in Supplementary Figure S15) are defined by projecting the transcript exons. First, the transcript exons are grouped by position. Then each group of exons defines a projected exon with the following rules:

- The start is the leftmost start of the non-first-exon of the group.
- The end is the rightmost end of the non-last-exon of the group that ends before the start of the next group of exons.

When the most frequent event annotated in the transcripts is an intron retention, the projected genomic exon is defined as a combination of the two exons flanking the retained intron. In Supplementary Figure S15, the exons 5 and 6 and the intron 5 are considered as one unique exon. As events included within one exon are not tested, this results in some events being missed.

Mapping. The first step of FaRLine is to map the reads to a reference genome. This step is done using Tophat-2.0.11[6]. tophat–min-intron-length 30–max-intron-length 1200000\-p 8 [–solexa1.3-quals for Sknsh_rep1 and Sknsh_rep2]\–transcriptome-index

A transcriptome index has been built by TopHat using EnsEMBL r75 annotations in gtf format. When a transcriptome index is used, the mapping steps are modified: instead of aligning first to the genome, which is the default behavior, TopHat uses Bowtie to align the reads to the transcript sequences first, then align the remaining unmapped reads to the genome. Minimal and maximal intron lengths have been modified (default 70 and 500000) to maximize the number of junctions detected, according to the statistics provided by FasterDB EnsEMBL r75 annotations.

The resulting alignment files have been filtered using samtools 0.1.19[33].

Samtools view -F 260 -f 1 -q 10 -b

With this step, only the primary alignments are kept. The minimum read alignment quality was set up so that multi-mapping reads were removed from the alignment file.

Annotation and quantification of alternative splicing events. For each gene, all the reads with at least one base overlapping the gene from the start to the end coordinates are retrieved. CIGAR strings are then used to find the alignments blocks. Junction reads are identified by the presence of at least one'N' letter in the CIGAR. Junction reads were filtered if:

- More than 10% of soft-clipping was detected in the alignment (it should not be the case with TopHat).
- An indel was close to the junction site, as it would make the junction position uncertain.

Junction read alignments are then processed block by block sequentially from left to right. Alignment blocks under 4 bp on read extremities are removed from the reads as we considered it is not sufficient to identify correctly the mapping localization. Then each block is compared to FasterDB annotations to check if the block boundaries correspond to known exons annotated in FasterDB, or to a putative new acceptor or donor site. First and last alignment blocks for each read must overlap one and only one exon for a read to be considered. For the inner blocks, if alignment blocks map to a succession of exons and introns, it is considered as an intron retention. For the acceptors and donors, we also added a supplementary filter. If a new donor is identified within a junction, we check if the junction also has an acceptor identified of the same length $+/-1$bp on the other side of the junction, showing most probably a problem of mapping. Once all the blocks are identified, the block annotations are used to annotate putative alternative splicing events: alternative skipped exon, multiple exon skipping, acceptor, or donor sites.

Once all the junction reads are processed, the alternative splicing events identified are pooled and the reads participating to each event are quantified, as well as the known exon-exon junction. If an exon-exon junction is annotated with multiple known acceptors and/or donors, all the possible junction reads are quantified and summed up. To fasten the quantification step, a junction coordinate file with the corresponding read numbers is produced from the read alignment using the same filters than described above and will be used for all the quantification tools: junction, exon skipping, acceptor and donor.

A challenge in defining the alternative skipped exon events is to identify the flanking exons. In the first version of FaRLine, these flankings exons were defined as the closest annotated genomic exons. This rule led to miss a lot of ASE events. Therefore, to define the flanking exons, we now use the information contained in the transcripts and in the reads. We consider each junction which skips an exon and is covered by at least one read. If this junction is annotated in the transcripts, we extract all annotated events containing this junction. Else, we annotate the event with the longest covered inclusion isoform. It allows FaRLine to be more robust to the incompleteness of the annotation compared to other methods, like MISO (Supplementary Figure S6). Panel C of Supplementary Figure S8 gives an example of an ASE reported by FaRLine but not by MISO because the exclusion isoform is not annotated in the transcripts.

Comparison with STAR. We also mapped the reads with STAR, ran FaRLine on these alignments and compared the predicted skipped exons with KisSplice. The main results are similar to what we found with TopHat. Indeed, without any filter, 69% of ASE annotated by KisSplice are also found by FaRLine and 24% of FaRLine's event by KisSplice (compared to 68% and 24% respectively for the mapping with TopHat). When we filter out the events with an unfrequent variant, we show that approximately 70% of predicted ASE are found by both approaches.

Quantification and differential analysis. Both pipelines perform ASE detection and quantification. The quantification step was done similarly in the two pipelines where only the junction reads were taken into account. To evaluate if using exonic reads in the quantification could increase the accuracy of our methods, we ran KisSplice on the MCF-7 dataset with the option –exonic reads set to on. In doing so, only the inclusion rate of the AS events changes. When comparing usage of only junction reads to usage of both junction and exonic reads, we observed that the p-values calculated strongly correlate as shown in Supplementary Figure S16. We found that some AS events became significant upon the addition of exonic reads but the opposite also happened. Inspection of these events revealed that many are borderline cases, where the p-value is close, but slightly above 5%. A manual inspection of the AS events with a very different p-value upon addition of exonic reads revealed that they correspond to exons overlapping alternative first or last exons (see *STARD4*, Supplementary Figure S17A) or novel exons located in poorly spliced introns (see *PANK2* and *PRRC2B*, Supplementary Figure S17 B and C). Overall, we concluded that exonic reads can bring some statistical power in cases where the skipped exon does not overlap with any other event. In case of more complex events, exonic reads tend to "pollute" the pairwise comparison.

The last step of the pipelines is the differential analysis of the expression levels of the variants. This task is performed using the kissDE[24] R package, which takes as input a table of read counts as in Supplementary Figure S18, and outputs a p-value and a DeltaPSI (Percent Spliced In).

Our statistical analysis adopted the framework of count regression with Negative Binomial distribution. We considered a 2-way design with interaction, with *isoforms* and *experimental conditions* as main effects. Following the Generalized Linear Model framework, the expected intensity of the signal was denoted by λ_{ijk} and was decomposed as:

$$\log \lambda_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} \tag{1}$$

where μ is the local mean expression of the gene, α_i the contribution of splicing variant i on the expression, β_j the contribution of condition j to the total expression, and $(\alpha\beta)_{ij}$ the interaction term. The target hypothesis was H_0: $\left\{(\alpha\beta)_{ij} = 0\right\}$ *i.e.* no interaction between the variant and the condition. If this interaction term is not null, a differential usage of a variant across conditions occurred. The test was performed using a Likelihood Ratio Test with one degree of freedom. To account for multiple testing, p-values were adjusted with a 5% false discovery rate (FDR) following a Benjamini-Hochberg procedure[34].

In addition to adjusted p-values, we report a measure of the magnitude of the effect. The measure we provide is based on the Percent Spliced In (PSI):

$$PSI_{condition} = \frac{counts_{variant1}}{counts_{variant1} + counts_{variant2}} \tag{2}$$

If counts for a variant are below a threshold, then the PSI is not calculated. This prevents from over-interpreting large magnitudes derived from low counts. When several replicates are available for a condition, then a PSI is computed for each replicate, and we calculate their mean.

Finally, we output the DeltaPSI:

$$DeltaPSI = PSI_{condition1} - PSI_{condition2} \tag{3}$$

unless one of the mean PSI of a condition could not be estimated. The higher the DeltaPSI, the stronger the effect. In practice, we consider only DeltaPSI larger than 0.1, a threshold below which it is difficult to perform any experimental validation.

SK-N-SH dataset. We downloaded a total of 959 M reads from http://genome.crg.es/encode_RNA_dashboard/hg19/35. They correspond to long polyA+ RNAs generated by the Gingeras lab, and are also accessible

with the following accession numbers (ENCSR000CPN - SRA: SRR315315, SRR315316 and ENCSR000CTT -SRA: SRR534309, SRR534310). For cell lines treated by retinoic acid, the reads were 76nt long, while they were 100nt long for the non treated cells. Hence we trimmed all reads to 76nt.

MCF-7 dataset. MCF-7 were transfected (two biological replicates) with siRNA targeting both DDX5 and DDX17 RNA helicases, and total RNA were extracted as described previously[36]. cDNA synthesis was made using the TruSeq Stranded Total RNA protocol after Ribo-Zero Gold-mediated elimination of ribosomal RNA (Beckman Coulter Genomics). High throughput sequencing (2×125 bp) was carried out on an Illumina HiSeq 2500 platform (Beckman Coulter Genomics), generating between 45 and 50 millions of paired-end pairs of reads. Raw datasets are available on GEO under the accession number GSE94372.

Reads were trimmed according to standard quality control filters using prinseq[37] and adapter were removed using cutadapt[38]. The resulting reads had length between 25 and 125nt. Because MISO is unable to deal with reads of unequal length, we selected only reads with length larger than 100nt (87% of the reads) and trimmed longer reads to 100nt.

Computational requirements, software availability and reproducibility of the results. FaRLine took 45 hours and 10 Go of RAM. The time-limiting step was TopHat2, which took 41 hours, even parallelised on 8 cores. When STAR was tested instead of TopHat2, it took 4 hours, but 30 Go of RAM. KisSplice took 30 hours and 10 Go of RAM. The RAM-limiting step was STAR which took 30Go of RAM. All the steps of the pipelines can be reproduced using the following tutorial:

http://kissplice.prabi.fr/pipeline_ks_farline.

Experimental Validation. SK-N-SH cells were purchased from the American Type Culture Collection (ATCC) and cultured using EMEM medium (ATCC) complemented with 10% FBS (Thermo Fisher Scientific). Cells were differentiated for 48 h using 6 μM of all-trans retinoic acid (Sigma-Aldrich).

After harvesting, total RNA were extracted using Tripure isolation reagent (Sigma-Aldrich), treated with DNase I (DNAfree, Ambion) for 30 min at 37 °C and reverse-transcribed (RT) using M-MLV reverse transcriptase and random primers (Invitrogen). Before PCR, all RT reaction mixtures were diluted at 2.5 ng μL of initial RNA. PCR reactions were performed using GoTaq polymerase (Promega).

MCF7 cells were cultured as described in[36]. RT-PCRs were performed using the same protocol as for SK-N-SH cells.

References

1. Pan, Q., Shai, O., Lee, L. J., Frey, B. J. & Blencowe, B. J. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nat Genet* **40**, 1413–1415 (2008).
2. Wang, E. T. *et al*. Alternative isoform regulation in human tissue transcriptomes. *Nature* **456**, 470–476 (2008).
3. Scotti, M. M. & Swanson, M. S. Rna mis-splicing in disease. *Nature Reviews Genetics* **17**, 19–32 (2016).
4. Edery, P. *et al*. Association of tals developmental disorder with defect in minor splicing component u4atac snrna. *Science* **332**, 240–243 (2011).
5. David, C. J. & Manley, J. L. Alternative pre-mrna splicing regulation in cancer: pathways and programs unhinged. *Genes & development* **24**, 2343–2364 (2010).
6. Trapnell, C. *et al*. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols* **7**, 562–578 (2012).
7. Wang, K. *et al*. Mapsplice: accurate mapping of rna-seq reads for splice junction discovery. *Nucleic acids research* **38**, e178–e178 (2010).
8. Robertson, G. *et al*. *De novo* assembly and analysis of rna-seq data. *Nature methods* **7**, 909–912 (2010).
9. Steijger, T. *et al*. Assessment of transcript reconstruction methods for rna-seq. *Nature methods* **10**, 1177–1184 (2013).
10. Anders, S., Reyes, A. & Huber, W. Detecting differential usage of exons from RNA-seq data. *Genome research* **22**, 2008–17 (2012).
11. Katz, Y., Wang, E. T., Airoldi, E. M. & Burge, C. B. Analysis and design of rna sequencing experiments for identifying isoform regulation. *Nature methods* **7**, 1009–1015 (2010).
12. Shen, S. *et al*. MATS: a Bayesian framework for flexible detection of differential alternative splicing from RNA-Seq data. *Nucleic Acids Research* e61–e61 (2012).
13. Sacomoto, G. A. T. *et al*. KISSPLICE: *de-novo* calling alternative splicing events from RNA-seq data. *BMC bioinformatics* **13**(Suppl 6), S5 (2012).
14. Martin, J. A. & Wang, Z. Next-generation transcriptome assembly. *Nature Reviews Genetics* **12**, 671–682 (2011).
15. Dargahi, D. *et al*. A pan-cancer analysis of alternative splicing events reveals novel tumor-associated splice variants of matriptase. *Cancer informatics* **13**, 167 (2014).
16. Freyermuth, F. *et al*. Splicing misregulation of scn5a contributes to cardiac-conduction delay and heart arrhythmia in myotonic dystrophy. *Nature communications* **7** (2016).
17. Grabherr, M. G. *et al*. Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data. *Nature biotechnology* **29**, 644 (2011).
18. Kopelman, N. M., Lancet, D. & Yanai, I. Alternative splicing and gene duplication are inversely correlated evolutionary mechanisms. *Nat Genet* **37**, 588–589 (2005).
19. Roux, J. & Robinson-Rechavi, M. Age-dependent gain of alternative splice forms and biased duplication explain the relation between splicing and duplication. *Genome research* **21**, 357–363 (2011).
20. Batzer, M. A. & Deininger, P. L. Alu repeats and human genomic diversity. *Nature Reviews Genetics* **3**, 370–379 (2002).
21. Lev-Maor, G., Sorek, R., Shomron, N. & Ast, G. The birth of an alternatively spliced exon: 3'splice-site selection in alu exons. *Science* **300**, 1288–1291 (2003).
22. Sorek, R. *et al*. Minimal conditions for exonization of intronic sequences: 5′ splice site formation in alu exons. *Molecular cell* **14**, 221–231 (2004).
23. Franz, M. *et al*. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics* **32**, 309–311, https://doi.org/10.1093/bioinformatics/btv557/oup/backfile/content_public/journal/bioinformatics/32/2/10.1093_bioinformatics_btv557/3/btv557.pdf (2016).
24. Lopez-Maestre, H. *et al*. SNP calling from RNA-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. *Nucleic Acids Research* **44**, e148–e148 (2016).

25. Poursani, E. M., Soltani, B. M. & Mowla, S. J. Differential expression of oct4 pseudogenes in pluripotent and tumor cell lines. *Cell Journal (Yakhteh)* **18**, 28 (2016).
26. Bacher, R. & Kendziorski, C. Design and computational analysis of single-cell rna-sequencing experiments. *Genome biology* **17**, 1 (2016).
27. Shen, S. *et al.* Widespread establishment and regulatory impact of alu exons in human genes. *Proceedings of the National Academy of Sciences* **108**, 2837–2842 (2011).
28. Tilgner, H., Grubert, F., Sharon, D. & Snyder, M. P. Defining a personal, allele-specific, and single-molecule long-read transcriptome. *Proceedings of the National Academy of Sciences of the United States of America* **111**, 9869–74 (2014).
29. Bolisetty, M. T., Rajadinakaran, G. & Graveley, B. R. Determining exon connectivity in complex mRNAs by nanopore sequencing. *Genome biology* **16**, 204 (2015).
30. Mallinjoud, P. *et al.* Endothelial, epithelial, and fibroblast cells exhibit specific splicing programs independently of their tissue of origin. *Genome research* **24**, 511–521 (2014).
31. Laustriat, D. *et al. In Vitro* and *In Vivo* Modulation of Alternative Splicing by the Biguanide Metformin. *Molecular Therapy. Nucleic Acids* **4**, e262 (2015).
32. Schulz, M. H., Zerbino, D. R., Vingron, M. & Birney, E. Oases: robust *de novo* rna-seq assembly across the dynamic range of expression levels. *Bioinformatics* **28**, 1086–1092 (2012).
33. Li, H. *et al.* The sequence alignment/map format and samtools. *Bioinformatics* **25**, 2078–2079 (2009).
34. Benjamini, Y. & Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)* 289–300 (1995).
35. Djebali, S. *et al.* Landscape of transcription in human cells. *Nature* **489**, 101–108 (2012).
36. Dardenne, E. *et al.* RNA Helicases DDX5 and DDX17 Dynamically Orchestrate Transcription, miRNA, and Splicing Programs in Cell Differentiation. *Cell Reports* (2014).
37. Schmieder, R. & Edwards, R. Quality control and preprocessing of metagenomic datasets. *Bioinformatics (Oxford, England)* PMID: 21278185. **27**, 863–864 (2011).
38. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet. journal* **17** (2011).

Acknowledgements

This work was performed on the computing facilities of the computing center LBBE/PRABI and the PSMN (Pole Scientifique de Modelisation Numerique) computing center of ENS de Lyon. This work was funded by the ANR-12-BS02-0008 (Colib'read) by the ABS4NGS ANR project (ANR-11-BINF-0001-06), Action n3.6 Plan Cancer 2009–2013, Fondation ARC (Programme Labellisé Fondation ARC 2014, PGA120140200853) and INCa (2014-154). Doctoral fellowships from ARC 1 - Région Rhône-Alpes (C.B.P), Science Without Borders - CNPq - Brazil (L.L. - grant process number 203362/2014-4), ARS RhÃ´ne-Alpes (A.R.) and post-doctoral fellowships from Fondation ARC (M.P.L).

Author Contributions

V.L. and D.A. designed the study. C.B.P., E.C. and J.B.C. developed FaRLine. L.L. and G.S. significantly improved the scalability of KisSplice, C.M., A.C. and V.L. developed KisSplice2RefGenome. C.B.P., L.L. and V.L. compared the two pipelines and classified the instance types. L.L. developed the supporting webpage. C.B.P. and C.F.B. planned the experimental validations. M.P.L., A.R., S.T., L.D. performed the experimental validations. C.B.P., D.A. and V.L. wrote the manuscript. All authors read and approved the final manuscript.

Additional Information

Supplementary information accompanies this paper at https://doi.org/10.1038/s41598-018-21770-7.

Competing Interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Chapter 5

On Bubble Generators in Directed Graphs

Preamble

Key points

- Theoretically, the number of bubbles in a graph can be exponential in the size of the graph. In practice, DBGs built from real datasets tend to be huge, usually containing millions of vertices and bubbles. Exploring the complete bubble space is thus unfeasible, and therefore relevant events described by hard-to-find bubbles may be lost;

- Here we describe an efficient and compact description of all bubbles in a graph G, called bubble generator $\mathcal{G}(G)$, which can be constructed in polynomial time. This is a suitable representative set of the complete bubble space;

- We further show a decomposition algorithm: any bubble B in a graph G can be represented as a sum of $O(n^2)$ bubbles belonging to $\mathcal{G}(G)$. This decomposition can be found in a total of $O(n^3)$ time, where n is the number of vertices of G;

- The practical potential of the bubble generator is demonstrated by applying it in two different directions in the analysis of a real RNA-seq dataset. First, we employed the generator as a preprocessing step to algorithms that find bubbles, by "cleaning" from the graph all unnecessary arcs. Second, we use it to find alternative splicing events in a reference-free context, showing that it can be complementary to current methods;

- These applications, however, remain only as proofs-of-concept. More work is needed to develop fully-fledged methods based on the bubble generator.

Status

Presented by *L.* in the *43rd International Workshop on Graph-Theoretic Concepts in Computer Science* (*WG 2017*), and published in *LNCS* [1]. Journal extension submitted to *Algorithmica*. The paper presented next is this journal version.

Author contributions

As this paper is in theoretical computer science, the author list is sorted by alphabetical order. All authors contributed equally to the paper.

Noname manuscript No.
(will be inserted by the editor)

On Bubble Generators in Directed Graphs

V. Acuña · R. Grossi · G. F. Italiano · L. Lima · R. Rizzi · G. Sacomoto · M.-F. Sagot · B. Sinaimeri

Received: date / Accepted: date

Abstract Bubbles are pairs of internally vertex-disjoint (s, t)-paths in a directed graph, which have many applications in the processing of DNA and RNA data. Listing and analysing all bubbles in a given graph is usually unfeasible in practice, due to the exponential number of bubbles present in real data graphs. In this paper, we propose a notion of bubble generator set, *i.e.,* a polynomial-sized subset of bubbles from which all the other bubbles can be obtained through a suitable application of a specific symmetric difference operator. This set provides a compact representation of the bubble space of a graph. A bubble generator can be useful in practice, since some pertinent information about all the bubbles can be more conveniently extracted from this compact set. We provide a polynomial-time algorithm to decompose any

V. Acuña
Center for Mathematical Modeling (UMI 2807 CNRS), University of Chile, Santiago, Chile.
E-mail: viacuna@dim.uchile.cl

R. Grossi
Università di Pisa, Pisa, Italy and Erable, INRIA, France.
E-mail: grossi@di.unipi.it

G. F. Italiano
LUISS University, Roma, Italy and and Erable, INRIA, France..
E-mail: gitaliano@luiss.it

R. Rizzi
Università di Verona, Verona, Italy.
E-mail: Romeo.Rizzi@univr.it

L. Lima
Erable INRIA Grenoble Rhône-Alpes, Université Lyon 1; CNRS, UMR5558, LBBE, Villeurbanne, France and Università di Roma "Tor Vergata", Roma, Italy.
E-mail: leandro.ishi-soares-de-lima@inria.fr

G. Sacomoto · M.-F. Sagot · B. Sinaimeri
Erable INRIA Grenoble Rhône-Alpes, Université Lyon 1; CNRS, UMR5558, LBBE, Villeurbanne, France.
E-mail: gustavo.sacomoto@gmail.com marie-france.sagot@inria.fr blerina.sinaimeri@inria.fr

bubble of a graph into the bubbles of such a generator in a tree-like fashion. Finally, we present two applications of the bubble generator on a real RNA-seq dataset.

Keywords Bubbles · Bubble generator set · Decomposition algorithm

1 Introduction

Bubbles are pairs of internally vertex-disjoint (s,t)-paths in a directed graph, which find many applications in the processing of DNA and RNA data. For example, in the genomic context, genome assemblers usually identify and remove bubbles in order to linearise the graph [16,21,25]. However, bubbles can also represent interesting biological events, *e.g.,* allelic differences (SNPs and indels) when processing DNA data [9,23,24], and alternative splicing events in RNA data [18,17,12,19]. Due to their practical relevance, several theoretical studies concerning bubbles were carried out in the past few years [2,4,15,18, 22], usually related to bubble-enumeration algorithms.

Although the enumeration of bubbles could be important to describe biological events appearing in the sequences, this approach has a significant disadvantage. Indeed, while many biological events can be represented by bubbles in a de Bruijn graph (see *e.g.* [19,14,17]) (the graph build from the reads provided by a sequencing process), the opposite is not true: most of the bubbles do not correspond to any biological phenomena and appear just because of a combination of other events [12,17]. In practice, due to the high throughput of second-generation sequencing machines, the genomic and transcriptomic De Bruijn graphs tend to be huge, usually containing from millions to billions of vertices. As expected, the number of bubbles also tends to be huge, in the worst case exponential in the number of vertices. As a consequence, algorithms that deal with bubbles either tend to simplify the graph by removing them, or just enumerate a small subset of the bubbles. Such subsets usually correspond to bubbles with some predefined characteristics, and may not be the best representatives of the biological phenomena under study. More worrying is the fact that, by focusing only on these particular bubbles, all the relevant events described by bubbles that do not satisfy the constraints may be lost. On the other hand, any algorithm that tries to be more exhaustive, say by enumerating a large portion of the bubbles, will certainly spend a prohibitive amount of time in real data graphs and thus it is not likely to be practical [12, 17]. This motivates further work for finding efficient ways to recognise bubbles that correspond to relevant events and/or to represent the set of bubbles in a more concise way.

In this paper, we propose an elementary bubble generator, *i.e.,* a subset of bubbles that is able to generate any other bubble in the graph. More specifically, we show how to identify, for any given directed graph G, a generator set $\mathcal{G}(G)$ of bubbles which is of polynomial size in the input graph, and such that any bubble in G can be obtained in a polynomial number of steps by properly combining the bubbles in the generator $\mathcal{G}(G)$ through a symmetric difference

operator. In several biological applications, it is desirable to decompose a bubble into elementary bubbles in such a way that only bubbles can be generated at each step of the decomposition. This happens, for instance, when one wishes to decompose complex alternative splicing events [19] into several elementary alternative splicing events. Our bubble generator enjoys this property: in order to take this into account, we consider a constrained version of the symmetric difference operator, where two bubbles are combinable only if the output is also a bubble (*i.e.,* the operator is undefined if the output is not a bubble). Moreover, we present a polynomial-time decomposition algorithm that, given a bubble B in the graph G, finds a sequence of bubbles from the generator $\mathcal{G}(G)$ whose combination results in B. Our algorithm can be applied when one needs to know how to decompose a bubble into its elementary parts, *e.g.,* when one is interested in identifying and decomposing complex alternative splicing events [19] into several elementary alternative splicing events.

At first sight, a bubble generator might seem related to a cycle basis, which represents a compact description of all Eulerian subgraphs in a graph. The study of cycle bases started a long time ago [13] and has attracted much attention in the last fifteen years, leading to many interesting results, such as the classification of different types of cycle bases, the generalisation of these notions to weighted and to directed graphs, as well as to several complexity results for constructing bases. We refer the interested reader to the books of Deo [6] and Bollobás [3], and to the survey of Kavitha *et al.* [10] for an in-depth coverage of cycle bases. Unfortunately, problems related to bubble generators appear to be very different (and more difficult) from their counterparts in cycle bases, so that it does not seem possible to apply directly to bubble generators all the techniques developed for cycle bases. Indeed, a cycle basis in a directed graph contains subgraphs that are *not* necessarily directed cycles in the original graph, but more generally cycles in the underline undirected graph [11]. As a consequence, the techniques developed for cycle bases in undirected and directed graphs cannot be applied to our problem, since they do not guarantee a decomposition into elementary bubbles, which generates only bubbles at each step.

To test the practical effectiveness of our generator set of bubbles, we applied it in two different directions in the analysis of a real RNA-seq dataset. First, we employed the generator as a preprocessing step in all algorithms that find bubbles, by "cleaning" from the graph all unnecessary arcs (*i.e.* arcs that do not belong to any bubble). Second, we use it to find alternative splicing (henceforth denoted by AS) events in a reference-free context. In particular, some bubbles in our generator set correspond to AS events that are hard to find by the state-of-art algorithm for AS events enumeration [12]. However, this application should still be seen just as a proof-of-concept on the practical potential of the bubble generator or as complementary to current methods, since it is still limited for the exhaustive enumeration of AS events. The latter would require a non-trivial procedure to enumerate AS-associated bubbles by combining generator bubbles and would be beyond the scope of this paper (see Section 6).

The remainder of this paper is organised as follows. Section 2 presents some definitions that will be used throughout the paper. Section 3 introduces our bubble generator. Section 4 presents a polynomial-time algorithm for decomposing any bubble in a graph into elements of our bubble generator. Section 5 presents two applications of the bubble generator in processing and analysing RNA data. Finally, we conclude with open problems in Section 6.

2 Preliminaries

Throughout the paper we assume that the reader is familiar with the standard graph terminology, as contained for instance in [5]. A graph is a pair $G = (V, E)$, where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges. For convenience, we may also denote the set of vertices V of G by $V(G)$ and its set of edges E by $E(G)$. We further set $n = |V(G)|$ and $m = |E(G)|$. A graph may be *directed* or *undirected*, depending on whether its edges are directed or undirected. In this paper, will deal with graphs that are directed, unweighted, finite and without parallel edges. An edge $e = (u, v)$ is said to be *incident* to the vertices u and v, and u and v are said to be the endpoints of $e = (u, v)$. For a directed graph, edge $e = (u, v)$ is said to be leaving vertex u and entering vertex v. Alternatively, $e = (u, v)$ is an outgoing edge for u and an incoming edge for v. The *in-degree* of a vertex v is given by the number of edges entering v, while the *out-degree* of v is the number of edges leaving v. The *degree* of v is the sum of its in-degree and out-degree.

We say that a graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given a subset of vertices $V' \subseteq V$, the subgraph of G *induced* by V', denoted by $G_{V'}$, has V' as vertex set and contains all edges of G that have both endpoints in V'. Given a subset of edges $E' \subseteq E$, the subgraph of G *induced* by E', denoted by $G_{E'}$, has E' as edge set and contains all vertices of G that are endpoints of edges in E'. Given a subset of vertices $V' \subseteq V$ and a subset of edges $E' \subseteq E$, we denote by $G \setminus V'$ the graph induced by $V \setminus V'$ and by $G \setminus E'$ the graph induced by $E \setminus E'$. Given a set S of subgraphs of G, G_S denotes the graph induced by the edges in $\cup_{s \in S} E(s)$. Given two subgraphs G and H, their union $G \cup H$ is the graph F for which $V(F) = V(G) \cup V(H)$ and $E(F) = E(G) \cup E(H)$. Their intersection $G \cap H$ is the graph F for which $V(F) = V(G) \cap V(H)$ and $E(F) = E(G) \cap E(H)$.

Let s, t be any two vertices in G. A (*directed*) *path* from s to t in G is a sequence of vertices and edges $s = v_1$, e_1, v_2, e_2, …, v_{k-1}, e_{k-1}, $v_k = t$, such that $e_i = (v_i, v_{i+1})$ for $i = 1, 2, \ldots, k-1$. Since there is no danger of ambiguity, in the remainder of the paper we will also denote a path simply as $s = v_1$, v_2, …, v_{k-1}, $v_k = t$ (*i.e.,* as a sequence of vertices). A path is *simple* if it does not contain repeated vertices, except possibly for the first and the last vertex. Throughout this paper, all the paths considered will be simple and referred to as paths. A path from s to t is also referred to as an (s, t)-path. The length of a path p is the number of edges in p and will be denoted by $|p|$. Note that, as a special case, we also allow a single vertex to be a path, *i.e.,* a path of length

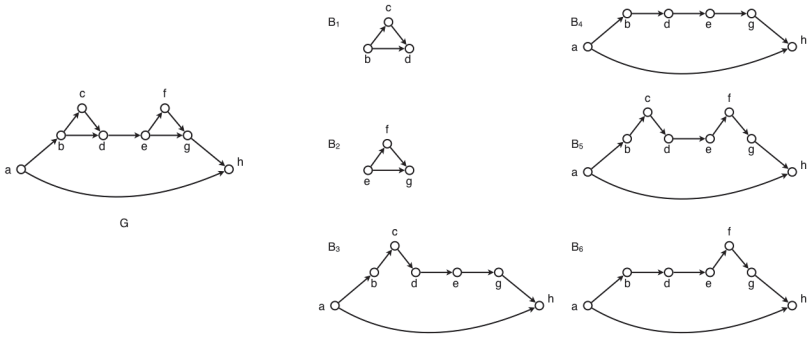


Fig. 1: An example of a graph G and the set $B(G)$ of all the bubbles in G. The set $\mathcal{G}(G) = \{B_1, B_2, B_4\}$ is a generator set that satisfies conditions of Theorem 1.

0. If p and q are paths, we say that p is a subpath of q if p is contained in q, and we denote this $p \subseteq q$. Given a path p_1 from x to y and a path p_2 from y to z, we denote by $p_1 \cdot p_2$ their concatenation, *i.e.,* the path from x to z defined by the path p_1 followed by p_2. A path q is a prefix of a path p if there exists a path r such that $p = q \cdot r$. Similarly, a path q is a suffix of a path p if there exists a path r such that $p = r \cdot q$. A *(directed) cycle* is a simple path (of length greater than zero) starting and ending on the same vertex.

Definition 1 Given a directed graph G and two (not necessarily distinct) vertices $s, t \in V(G)$, an (s,t)-*bubble* consists of two directed (s,t)-paths that are internally vertex disjoint. Vertex s is the source and t is the target of the bubble. If $s = t$ then exactly one of the paths of the bubble has length 0, and therefore B corresponds to a directed cycle. In this case, we say that B is a *degenerate bubble*.

In Fig. 1 we show an example of a graph and all the bubbles in it. We denote by $B(G)$ the set of all bubbles in G. Before giving formally the definition of bubble generator of G, we recall some basic definitions of cycle bases in undirected graphs.

Let G be an undirected graph. Two subgraphs G_1, G_2 of G can be combined by the operator Δ that simply consists in the symmetric difference of the set of edges. More formally, $G_1 \Delta G_2 = (G_1 \cup G_2) \setminus (E(G_1) \cap E(G_2))$ where $E(G_i)$ is the set of edges of G_i. With this operation, it can be shown that the space of all Eulerian subgraphs of G (called the *cycle space* of G) is a vector space [8, 10,11,13]. In the theory of vector spaces, a set of vectors is said to be *linearly dependent* if one of the vectors in the set can be defined as a linear combination of the others; if no vector in the set can be written in this way, then the vectors are said to be *linearly independent* [20]. A *basis* is a minimum set of vectors, such that any vector in the space is a linear combination of this set. Clearly a basis is a set of linearly independent vectors. Furthermore, given a vector

space and a set of k linearly independent vectors F, the subspace of vectors generated starting from elements in $\mathcal{F}$ is called the *span* of $\mathcal{F}$ and its dimension is k. It is well-known that a cycle basis for a connected undirected graph G, denoted by $\mathcal{C}(G)$, has dimension $m-n+1$. If the graph G is not connected this is generalised to $m-n+c$, where c is the number of connected components (see, *e.g.,* [8,10,11,13]) .

As mentioned in Section 1, we are interested in decomposing a bubble into elementary bubbles in such a way that, at each step of the decomposition, only bubbles are generated. To ensure this property, we define next a suitable symmetric difference operator which takes as input two bubbles and produces one bubble as output. Given two bubbles B_1 and B_2, *the constrained symmetric difference* operator Δ is such that $B_1 \Delta B_2$ is defined if and only the subgraph induced by $(E(B_1) \cup E(B_2)) \setminus (E(B_1) \cap E(B_2))$ is a bubble. Otherwise, we say that $B_1 \Delta B_2$ is undefined. If $B_1 \Delta B_2$ is defined, we also say that B_1 and B_2 are *combinable*. Given two combinable bubbles B_1 and B_2, we refer to $B_1 \Delta B_2$ as the *sum of* B_1 *and* B_2, and denote it also by $B_1 + B_2$. We also say that the bubble $B_1 + B_2$ is *generated* from bubbles B_1 and B_2, or alternatively that it can be *decomposed* into the bubbles B_1 and B_2. Let $\mathcal{B}$ be a set of bubbles in G. We say that a bubble B is *spanned by* $\mathcal{B}$ if it can be generated starting from bubbles in $\mathcal{B}$. The set of all the bubbles spanned by $\mathcal{B}$ is called the *span of* $\mathcal{B}$. $\mathcal{B}$ is a *bubble generator* if each bubble in G is spanned by $\mathcal{B}$, *i.e.,* each bubble in G can be generated by starting from the bubbles in $\mathcal{B}$.

Due to our constrained symmetric difference operator Δ, all subgraphs generated by the elements in $\mathcal{B}$ are necessarily bubbles. Since not all pairs of bubbles of G are combinable, the bubble space is not closed under Δ, and therefore it does not form a vector space (over $\mathbb{Z}_2$). Hence, the techniques developed for cycle bases cannot be applied directly to bubble generators.

A generator is *minimal* if it does not contain a proper subset that is also a generator; and a generator is *minimum* if it has the minimum cardinality. We are interested in finding a minimum bubble generator of a given directed graph G.

3 The bubble generator

In this section, we present a bubble generator for a directed graph G. Throughout, we assume that shortest paths in G are unique. This is without loss of generality, since there are many standard techniques for achieving this, including perturbing edge weights by infinitesimals. However, for our goal, it suffices to use a "lexicographic ordering". Namely, we define an arbitrary ordering $v_1, \ldots, v_n$ on the vertices of G. A path p is considered lexicographically shorter than a path q if the length of p is strictly smaller than the length of q, or, if p and q have the same length, the sequence of vertices associated with p is lexicographically smaller than the sequence associated with q. We denote this by $p <_{lex} q$.

We denote by $B = (p, q)$ the bubble having p, q as its two internally vertex-disjoint paths, referred to as *legs*. We denote by $\ell(B)$ (resp., by $\mathcal{L}(B)$) the shorter (resp., longer) between the two legs p, q of B. Note that, because of the lexicographic order, there are no ties. We also denote by $|B|$ the number of edges of bubble B. Note that $|B| = |\ell(B)| + |\mathcal{L}(B)|$. Next, we define a total order on the set of bubbles.

Definition 2 Let B_1 and B_2 be any two bubbles. B_1 is *smaller* than B_2 (in symbols, $B_1 < B_2$) if one of the following holds: either (i) $\mathcal{L}(B_1) <_{lex} \mathcal{L}(B_2)$; or (ii) $\mathcal{L}(B_1) = \mathcal{L}(B_2)$ and $\ell(B_1) <_{lex} \ell(B_2)$.

Definition 3 A bubble B is *composed* if it can be obtained as a sum of two smaller bubbles. Otherwise, the bubble B is called *simple*.

For a directed graph G, we denote by $\mathcal{S}(G)$ the set of simple bubbles of G. It is not difficult to see that $\mathcal{S}(G)$ is a generator. We are not able for now to prove that any bubble in G can be obtained in a polynomial number of steps from bubbles in $\mathcal{S}(G)$. Nevertheless, to achieve the latter goal, we will introduce next another generator $\mathcal{G}(G) \supseteq \mathcal{S}(G)$. Let $p : s = x_0, x_1, \ldots, x_h = t$ be a path from s to t and let $0 \leq i \leq j \leq h$. To ease the notation, we denote by $p_{i,j}$ the subpath of p from x_i to x_j, and refer also to $p_{0,j}$ as $p_{s,j}$ and to $p_{i,h}$ as $p_{i,t}$. The next theorem provides some properties of simple bubbles.

Theorem 1 *Let B be a simple (s, t)-bubble in a directed graph G. The following holds:*

(1) $\ell(B)$ is the shortest path from s to t in G;
(2) Let $\mathcal{L}(B) = s, v_1, \ldots, v_r, t$. Then $s, v_1, \ldots, v_r$ is the shortest path from s to v_r in G.

Proof Let B be a simple (s, t)-bubble: we show that both conditions (1) and (2) must hold.

We first consider condition (1). If B is degenerate, then it trivially satisfies condition (1). Therefore, assume that B is non-degenerate and, by contradiction, that $\ell(B)$ is not the shortest path from s to t. Let $p^* : s = x_0, x_1, \ldots, x_h = t$ be the shortest path from s to t in G. For $0 \leq i \leq j \leq h$, by subpath optimality, $p^*_{i,j}$ is the shortest path from x_i to x_j. Let k be the smallest index, $0 \leq k < h$, for which the edge (x_k, x_{k+1}) does not belong to either one of the legs of B. Such an index k must exist, as otherwise p^* would coincide with a leg of B. Furthermore, let l, $k < l \leq h$, be the smallest index greater than k for which $x_l \in V(B)$. Such a vertex x_l must also exist, since $x_h = t \in V(B)$. In other words, x_k is the first vertex of the bubble B where p^* departs from B and x_l, $l > k$, is the first vertex where the shortest path p^* intersects again the bubble B. By definition of x_k and x_l, $p^*_{k,l}$ is internally vertex-disjoint with both legs of B. We now claim that B can be obtained as the sum of two smaller bubbles, thus contradicting our assumption that B is a simple bubble.

To prove the claim, we distinguish two cases, depending on whether x_k and x_l are on the same leg of B or not. Consider first the case when x_k and x_l are

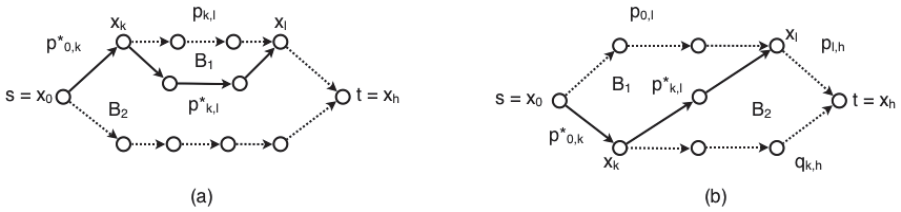


Fig. 2: Case (1) of the proof of Theorem 1. The prefix of the shortest path from s to t is shown as a solid line.

on the same leg p of B (see Fig. 2(a)). Let B_1 be the bubble with $\ell(B_1) = p^*_{k,l}$ and $\mathcal{L}(B_1) = p_{k,l}$. First, note that if either $x_k \neq s$ or $x_l \neq t$, then $p_{k,l}$ is a proper subpath of a leg of B. Hence, $|\mathcal{L}(B_1)| = |p_{k,l}| < |\mathcal{L}(B)|$, and $B_1 < B$. Otherwise, suppose $s = x_k$ and $t = x_l$. Then either $\mathcal{L}(B_1) = \ell(B) <_{lex} \mathcal{L}(B)$, or $\mathcal{L}(B_1) = \mathcal{L}(B)$ and $\ell(B_1) = p^*_{k,l} = p^* <_{lex} \ell(B)$. In both cases, $B_1 < B$. Let B_2 be the bubble which is obtained from B by replacing $p_{k,l}$ by $p^*_{k,l}$ (see Fig. 2(a)). Since $p^*_{k,l}$ is a shortest path, by subpath optimality, $p^*_{k,l} <_{lex} p_{k,l}$, thus $B_2 < B$. As a result, B can be obtained as the sum of two smaller bubbles B_1, B_2, thus contradicting the assumption that B is simple.

Consider now the case where x_k and x_l are on different legs of B (see Fig. 2(b)). Notice that this means $x_k \neq s$ and $x_l \neq t$. Let p be the leg containing x_l and q the one containing x_k. Note that $p = p_{0,l} \cdot p_{l,h}$ and $q = p^*_{0,k} \cdot q_{k,h}$. Moreover, the two legs of bubble B_1 are $p^*_{0,k} \cdot p^*_{k,l} <_{lex} q$ and $p_{0,l}$, which is a proper subpath of p. Hence, $B_1 < B$. The two legs of bubble B_2 are $q_{k,h}$ which is a proper subpath of q and $p^*_{k,l} \cdot p_{l,h} <_{lex} p$. Hence, $B_2 < B$, and $B = B_1 + B_2$ which implies again that B is not simple.

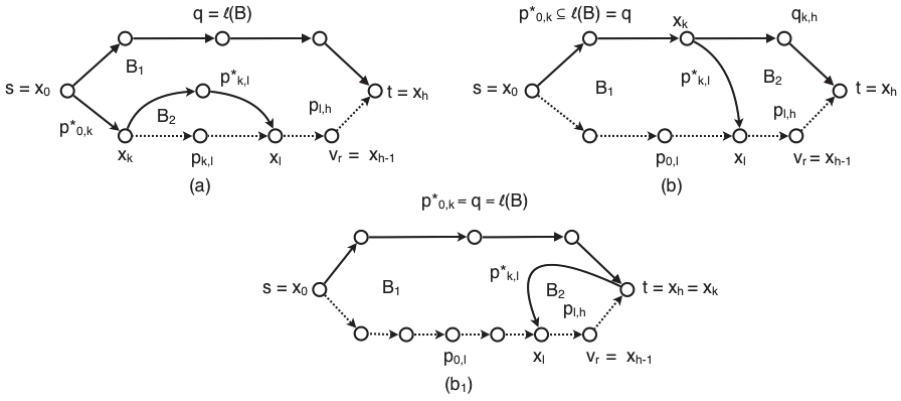


Fig. 3: Case (2) of the proof of Theorem 1. The shortest path from s to t and the prefix of the shortest path from s to v_r are shown as solid lines.

We show now that B satisfies also condition (2). Assume, by contradiction, that B satisfies condition (1) but not (2), and so $p = s, v_1, \ldots, v_r$ (note that p is equal to $\mathcal{L}(B)$ without its last edge) is not the shortest path from s to v_r in G. Let $p^* : s = x_0, \ldots, x_{h-1} = v_r$, $p^* \neq p$, be such a shortest path in G. Similarly to the previous case, let k be the smallest index, $0 \leq k < h-1$, for which the edge (x_k, x_{k+1}) does not belong to either one of the legs of B, *i.e.* x_k is the first vertex where the shortest path p^* departs from B. Such an index k must exist, as otherwise p^* would coincide with a leg of B. Let $l, k < l \leq h-1$, be the smallest index such that $x_l \in V(B)$. Namely, x_l is the first vertex after x_k where the shortest path p^* intersects again bubble B. Such a vertex x_l must always exist, since $x_{h-1} = v_r \in V(B)$. Since $k < l$, we have that $|p^*_{k,l}| \geq 1$. Furthermore, we claim that x_l must be in $\mathcal{L}(B) \setminus \{s, t\}$. If this were not the case, we would have two distinct shortest paths from s to x_l in G ($p^*_{x_0,x_l}$ and the subpath of $\ell(B)$ from $s = x_0$ to x_l), which contradicts our assumption that shortest paths are unique.

We again distinguish two cases: when both x_k, x_l belong to $\mathcal{L}(B)$, and when $x_k \in \ell(B)$ and $x_l \in \mathcal{L}(B)$. We set $p = \mathcal{L}(B), q = \ell(B)$.

In the first case (see Fig. 3(a)), let B_1 be the bubble with $\ell(B_1) = \ell(B)$ and $\mathcal{L}(B_1) = p^*_{0,k} \cdot p^*_{k,l} \cdot p_{l,h}$. Since $|p^*_{k,l}| <_{lex} |p_{k,l}|$ then $\mathcal{L}(B_1) <_{lex} \mathcal{L}(B)$, and thus $B_1 < B$. Let B_2 be the bubble with $\ell(B_2) = p^*_{k,l}$, and $\mathcal{L}(B_2) = p_{k,l}$. Since $\mathcal{L}(B_2) \subset \mathcal{L}(B)$ (as $x_k \neq t$), $B_2 < B$. As a result, B can be obtained as the sum of two smaller bubbles B_1, B_2, thus contradicting the assumption that B is simple.

In the second case (see Fig. 3(b)), let B_1 be the bubble with $\ell(B_1) = p^*_{0,k} \cdot p^*_{k,l}$ and $\mathcal{L}(B_1) = p_{0,l}$. Since $\mathcal{L}(B_1) \subset \mathcal{L}(B)$, $B_1 < B$. Let B_2 be the bubble with $\ell(B_2) = q_{k,h}$, and $\mathcal{L}(B_2) = p^*_{k,l} \cdot p_{l,h}$. Since $|\mathcal{L}(B_2)| < |\mathcal{L}(B)|$, $B_2 < B$. Again, B can be obtained as the sum of two smaller bubbles B_1, B_2, thus contradicting the assumption that B is simple. Finally, notice that this includes also the case $x_k = t$ and the argument holds identically with B_2 being a degenerate bubble. For the sake of clarity, we depicted this case separately in Fig. 3(b_1). ■

Given a directed graph G, we denote by $\mathcal{G}(G)$ the set of bubbles in G satisfying conditions (1) and (2) of Theorem 1. An example of a graph together with a generator $\mathcal{G}(G)$ is given in Fig. 1.

Theorem 2 *Let G be a directed graph. The following holds:*

(1) $\mathcal{G}(G)$ is a generator set for all the bubbles of G;
(2) $|\mathcal{G}(G)| \leq nm$.

Proof (1) Recall that $\mathcal{S}(G)$ is the set of simple bubbles. By Theorem 1, $\mathcal{S}(G) \subseteq \mathcal{G}(G)$, and thus $\mathcal{G}(G)$ is a generator set for all the bubbles of G.
(2) Since every bubble b in $\mathcal{G}(G)$, with $\ell(b) = s, u_1, \ldots, t$ and $\mathcal{L}(b) = s, v_1, \ldots, v_r, t$, can be uniquely identified by its vertex s and its edge (v_r, t), then the number of bubbles in $\mathcal{G}(G)$ is upper-bounded by nm. ■

The upper bound given in Theorem 2 is asymptotically tight, as shown by the family of simple directed graphs on vertex set $V_n = \{1, 2, \ldots, n\}$ and all possible $n * (n-1)$ edges in their edge set $E_n = \{(u, v) : u \neq v, u, v \in V\}$.

Remark 1 Conditions (1) and (2) of Theorem 1 are not sufficient to guarantee that a bubble is simple, *e.g.,* see Fig. 4. Thus, the generator $\mathcal{G}(G)$ is not necessarily minimal. Recall that a generator is *minimal* if it does not contain a proper subset that is also a generator; and a generator is *minimum* if it has the minimum cardinality.

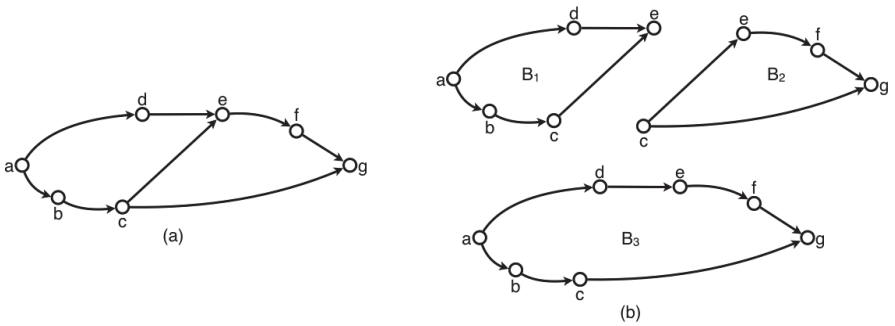


Fig. 4: An example showing that conditions (1) and (2) of Theorem 1 are not sufficient to guarantee that a bubble is simple. (a) A directed graph G. (b) The three bubbles B_1, B_2 and B_3 of G satisfying conditions (1) and (2) of Theorem 1, in which B_1 and B_2 are simple, but B_3 is composed, since $B_1 < B_3$, $B_2 < B_3$ and $B_3 = B_1 + B_2$.

4 A polynomial-time algorithm for decomposing bubbles

The main result of this section is to provide a polynomial-time algorithm for decomposing any bubble of G into bubbles of $\mathcal{G}(G)$. To do so, we make use of a tree-like decomposition. We need to take extra care in this decomposition since a naive approach could generate (several times) all the bubbles that are smaller than B, yielding an exponential number of steps.

Definition 4 A bubble B is *short* if it satisfies condition (1) of Theorem 1, but not necessarily condition (2). Namely, let $\mathcal{L}(B) = s, v_1, \ldots, v_r, t$ be such that $\ell(B)$ is a shortest path from s to t in G but $s, v_1, \ldots, v_r$ is not necessarily the shortest path from s to v_r in G.

We next introduce a measure for describing how "close" is a bubble to being short.

Definition 5 Given an (s,t)-bubble B, let p^* be the shortest path from s to t. We say that B is *k-short*, for $k \geq 0$, if there is a leg $p \in \{\ell(B), \mathcal{L}(B)\}$ for which p^* and p share a prefix of exactly k edges.

Since in our case shortest paths are unique, only one leg of a bubble B can share a prefix with the shortest path p^*. Furthermore, any bubble B is k-short for some k, $0 \leq k \leq |\ell(B)|$. In particular, a bubble is short if and only if it is k-short for $k = |\ell(B)|$.

Definition 6 Given a k-short bubble, we define the *short residual of B* as follows: $residual_s(B) = |B| - k$.

Since $0 \leq k \leq |\ell(B)|$, and $|B| = |\ell(B)| + |\mathcal{L}(B)|$, we have that $|\mathcal{L}(B)| \leq residual_s(B) \leq |B|$.

We now present our polynomial time algorithm for decomposing a bubble of the graph G into bubbles of $\mathcal{G}(G)$. In the following, we assume that we have done a preprocessing step to compute all-pairs shortest paths in G in $O(mn + n^2 \log n)$ time.

Lemma 1 *Let B be an (s,t)-bubble that is not short. Then, B can be decomposed into two bubbles B_1 and B_2 ($B = B_1 + B_2$), such that: (a) B_1 is short, and (b) $residual_s(B_2) < residual_s(B)$. Moreover, B_1 and B_2 can be found in $O(n)$ time.*

Proof Let B be a k-short (s,t)-bubble, $0 \leq k < |\ell(B)|$ and let $p^* : s = x_0, x_1, \ldots, x_h = t$ be the shortest path from s to t in G. To prove (a), we follow a similar approach to Theorem 1. Since B is k-short, there is a leg $p \in \{\ell(B), \mathcal{L}(B)\}$ such that p^* and p share a prefix of exactly k edges, $0 \leq k < h$. In other terms, leg p starts with edges (x_0, x_1), …, (x_{k-1}, x_k), the edge (x_k, x_{k+1}) is not in leg p, *i.e.*, x_k is the first vertex where the shortest path p^* departs from the leg p. Note that as a special case, $k = 0$ and $x_k = x_0 = s$. Let $l, k < l \leq h$, be the smallest index such that $x_l \in V(B)$. Namely, x_l is the first vertex after x_k where the shortest path p^* intersects again the bubble B. Such a vertex x_l must always exist, since $x_h = t \in V(B)$. Since $k < l$, we have that $|p^*_{k,l}| \geq 1$. We have two possible cases: either the vertices x_k and x_l are on the same leg of B (see Fig. 2(a)) or x_k and x_l are on different legs of B (see Fig. 2(b)). In either case, we can decompose B as $B = B_1 + B_2$, as illustrated in Fig. 2. Note that in both cases, the bubble B_1 is short since one leg of B_1 is a subpath of the shortest path p^*, and hence a shortest path itself by subpath optimality.

Consider now B_2 in Fig. 2. To prove (b), we distinguish among the following three cases: (1) $x_k \neq s$ and vertices x_k and x_l are on the same leg of B; (2) $x_k \neq s$ and vertices x_k and x_l are on different legs of B; (3) $x_k = s$. First, consider case (1) (see Fig. 2(a)) and note that $residual_s(B) = |p_{k,l}| + |p_{l,h}| + |q_{0,h}|$ where q is the other leg of B different from p. Moreover, $residual_s(B_2) = |p_{l,h}| + |q_{0,h}|$. Hence, $residual_s(B) - residual_s(B_2) = |p_{k,l}| \geq |p^*_{k,l}| \geq 1$. Consider now case (2), (see Fig. 2(b)) and note that

$residual_s(B) = |p_{0,l}| + |p_{l,h}| + |q_{k,h}|$ and $residual_s(B_2) = |p_{l,h}| + |q_{k,h}|$, and thus $residual_s(B) - residual_s(B_2) = |p_{0,l}| \geq |p^*_{0,k}| + |p^*_{k,l}| \geq 1$. The proof of case (3) is completely analogous to the one of case (1), with $x_k = s$ and $p^*_{0,k} = \emptyset$, and again $residual_s(B) - residual_s(B_2) = |p_{k,l}| \geq |p^*_{k,l}| \geq 1$. In all cases, $residual_s(B) - residual_s(B_2) > 0$, and thus the claim follows. Finally, note that in order to compute B_1 and B_2 from B, it is sufficient to trace the shortest path p^*. Since all shortest paths are pre-computed in a preprocessing step, this can be done in $O(n)$ time. ■

Lemma 2 *Any bubble B can be represented as a sum of $O(n)$ (not necessarily distinct) short bubbles. This decomposition can be found in $O(n^2)$ time in the worst case.*

Proof Each time we apply Lemma 1 to a bubble B, we produce in $O(n)$ time a short bubble B_1 and a bubble B_2 such that $residual_s(B_2) < residual_s(B)$. Since $residual_s(B) \leq |B| \leq n$, the lemma follows. ■

We next show how to further decompose short bubbles. Before doing that, we define the notion of *residual* for short bubbles, which measures how "close" is a short bubble to being a bubble of our generator set $\mathcal{G}(G)$.

Definition 7 Let B be a short (s,t)-bubble, let $\ell(B) = p^*_1$ be the shortest path from s to t in G, and let $\mathcal{L}(B) = s, v_1, \ldots, v_r, t$ be the other leg of B. Let p be the longest prefix of $\mathcal{L}(B) - (v_r, t)$ such that p is a shortest path in G. Then, the *residual of B* is defined as $residual(B) = |\mathcal{L}(B)| - 1 - |p|$.

Since p is a prefix of $\mathcal{L}(B) - (v_r, t)$, we have that $0 \leq |p| \leq |\mathcal{L}(B)| - 1$. Thus, $0 \leq residual(B) \leq |\mathcal{L}(B)| - 1$.

Lemma 3 *Let B be a short (s,t)-bubble such that $residual(B) > 0$. B can be decomposed into two bubbles B_1 and B_2 ($B = B_1 + B_2$) such that B_1 and B_2 are short and $residual(B_1) + residual(B_2) < residual(B)$. Moreover, it is possible to find the bubbles B_1 and B_2 in $O(n)$ time.*

Proof Since B is a short (s,t)-bubble, it satisfies condition (1) of Theorem 1. Furthermore, as $residual(B) > 0$, it does not satisfy condition (2). Therefore, there exists two bubbles $B_1 < B$ and $B_2 < B$ such that $B = B_1 + B_2$ (from Theorem 1). Since $\ell(B)$ is the shortest path from s to t, using arguments similar to the ones in Theorem 1, it can be shown that B can be decomposed into B_1 and B_2 and the only possible cases are the ones depicted in Fig. 3. Note that in all three cases of Fig. 3, each of the bubbles B_1 and B_2 has one leg that is a shortest path. Thus, in all three cases, B_1 and B_2 are short. Moreover, in Fig. 3(a), $residual(B_1) \leq |p_{l,h}| - 1$ and $residual(B_2) \leq |p_{k,l}| - 1$. Therefore, $residual(B_1) + residual(B_2) \leq |p_{l,h}| - 1 + |p_{k,l}| - 1 = residual(B) - 1 < residual(B)$. Similarly, in Fig. 3(b) and (b_1), $residual(B_1) \leq |p_{0,l}| - 1$, $residual(B_2) \leq |p_{l,h}| - 1$, and thus, $residual(B_1) + residual(B_2) \leq |p_{0,l}| - 1 + |p_{l,h}| - 1 = residual(B) - 1 < residual(B)$. In all three cases, B_1 and B_2 are short and $residual(B_1) + residual(B_2) < residual(B)$. The claim thus follows.

Once again, observe that in order to compute B_1 and B_2 from B, it is sufficient to trace the shortest path p^*. Since all shortest paths are pre-computed in a preprocessing step, this can be done in $O(n)$ time. ■

Lemma 4 *Any short bubble B has a tree-like decomposition into $O(n)$ (not necessarily distinct) bubbles from the generator $\mathcal{G}(G)$. This decomposition can be found in $O(n^2)$ time in the worst case.*

Proof Each time we apply Lemma 3 to a short bubble B, we produce in $O(n)$ time two short bubbles B_1 and B_2 such that $residual(B_1) + residual(B_2) < residual(B)$. Since $|\ell(B)| + residual(B) \leq n$, this implies that a short bubble can be decomposed in $O(n)$ bubbles from the generator set $\mathcal{G}(G)$ in $O(n^2)$ time. ■

Theorem 3 *Given a graph G, any bubble B in G can be represented as a sum of $O(n^2)$ bubbles that belong to $\mathcal{G}(G)$. This decomposition can be found in a total of $O(n^3)$ time.*

Proof The theorem follows by Lemma 2 and Lemma 4. ■

5 Applications of the bubble generator in analysing RNA-seq data

In this section, we describe as a proof-of-concept, two applications of the bubble generator to the analysis of RNA-seq data.

Our test dataset is a subset (coming from the same chromosome) of reads of the 58 million RNA-seq Illumina paired-end reads extracted from the mouse brain tissue (available in the ENA repository under the following study: PRJEB25574). We mapped all reads to the *Mus Musculus* reference genome and annotations (Ensembl release 94) using STAR [7]. We then selected only the reads mapping to chromosome 10 of the genome, comprising 4,932,572 reads, as our test dataset. We built the de Bruijn graph from these reads and applied standard sequencing-error-removal procedures, by using KisSplice [12, 17], a method to find alternative splicing events in a reference-free context by enumerating bubbles in a de Bruijn Graph. Finally, we extracted the bubble generator from the resulting graph, and evaluated it on two aspects: (i) how well it can preprocess the de Bruijn graph to reduce the work required by a subsequent bubble enumeration algorithm, and (ii) how it performs in terms of finding alternative splicing events. These applications are detailed in the following subsections.

5.1 Preprocessing the de Bruijn graph

Similarly to the practical application of a cycle base, the bubble generator can be used as a preprocessing step in all algorithms that find bubbles, by "cleaning" from the graph all unnecessary edges and vertices, *i.e.* those that do not belong to any bubble. In KisSplice [12,17], this cleaning is based

on a biconnected component (BCC) decomposition. A biconnected undirected graph G is a connected graph such that, for any $v \in V(G)$, $G - v$ is connected. Biconnected components (BCCs) are the maximal biconnected subgraphs of a graph G. Given a directed graph, consider its underlying undirected version by ignoring the direction of its edges. Clearly a bubble in the directed graph corresponds to a cycle in the underlying graph, and every edge that belongs to a cycle, belongs also to a BCC of the graph. The graph can then be cleaned by removing every vertex or edge that does not belong to a BCC. This cleaning partitions a potentially massive graph into smaller subgraphs, which are then processed by a bubble enumeration algorithm (*e.g.* [12,17]). However, the BCC-decomposition-based cleaning is not perfect: some vertices and edges might belong only to undirected cycles and not to bubbles.

To improve over this, we perform a more refined cleaning: we compute a bubble generator $\mathcal{G}(G)$ of the directed graph G and we remove every edge and vertex that do not belong to any bubble in $\mathcal{G}(G)$. Notice that this would be a perfect cleaning, meaning that after applying it, every edge of the graph would belong to some bubble.

We evaluated this cleaning procedure on the de Bruijn graph contructed from our test dataset. We first applied the BCC-decomposition-based cleaning on this de Bruijn graph. Then to the result obtained, which is now irreducible by this cleaning, we apply a second cleaning procedure using the bubble generator. The bubble generator cleaning led to a reduction of 40.1% on the number of vertices and of 39.8% on the number of edges. This shows that the generator can indeed yield a better procedure for cleaning the graph, although computing the generator requires more time than computing the BCCs (recall that the BCCs can be computed in linear time). In other words, as expected, a better cleaning comes at the expense of a higher computing time.

5.2 Calling alternative splicing events

As a second application, we consider the problem of finding AS events in a reference-free context. As already mentioned in the introduction, this is a challenging problem in bioinformatics. Indeed, local assemblers such as KisSplice [12] are faced with a dramatically large (and often practically unfeasible) running time due to the exponentially large number of bubbles present, most of which are not interesting as they are not related to AS events. Indeed, a significantly large number of bubbles is due to artefacts of the de Bruijn graph created by repeats longer than the reads (*i.e.,* artificial bubbles not associated with biological events). Hence, in order not to get "lost" in listing false positives, KisSplice relies on heuristics that try to avoid listing bubbles that traverse a repeat-induced subgraph. More specifically, based on the idea that subgraphs of the De Bruijn graph related to repeats have many branching vertices (*i.e.* vertices with in-degree or out-degree at least 2), KisSplice enumerates only bubbles with a number of branching vertices that is below some threshold b.

The question we tackle in this section is how many AS events we are able to find just by looking at the bubbles in the generator set. To this purpose, given our dataset we consider the set of bubbles belonging to the generator and the set of bubbles generated by KisSplice (KisSplice being run with default parameters, with a maximum number of branching vertices set to 5). In both cases some simple filters are applied to filter out bubbles that probably do not correspond to AS events (*e.g.* the shorter leg of AS events usually has a length between $2k-8$ and $2k-2$, with k being the size of the k-mer in the De Bruijn graph [12,17]). We obtained, as putative AS events, 1403 bubbles for the generator set and 1293 bubbles for KisSplice. In order to assess the precision of our method, we mapped the bubbles output by both methods to the *Mus Musculus* reference genome and annotations (Ensembl release 94) using STAR [7], which were then analysed by KisSplice2RefGenome [1]. KisSplice2RefGenome provides, for each bubble, the gene name, the AS event type (exon skipping, alternative acceptor/donor splice site, intron retention, etc), the genomic coordinates and the list of splice sites used (novel or annotated). We retrieved only those that corresponded to AS events.

Among the generator bubbles classified as putative AS events, 1085 bubbles correspond to true AS events, according to KisSplice2RefGenome, yielding a precision (AS events / putative AS events) of 77.3%. Note that the precision of KisSplice is 90.3% for this dataset. However, what is interesting to see is that 18.5% of the putative AS events from our bubble generator will never be found by KisSplice using the default parameters, as they have more than 5 branching vertices. Moreover, 10% of these bubbles correspond to true AS events that are missed by KisSplice. Increasing the maximum number of allowed branching vertices will increase the running time of KisSplice's algorithm exponentially. A large threshold of b is in practice unfeasible. Since we have bubbles corresponding to putative AS events in the generator that have more than 20 branching vertices, these will be missed by KisSplice.

This analysis shows the practical interest of the bubble generator. Even this simple application led to results that were comparable with the state-of-art algorithm KisSplice and sometimes complementary.

6 Conclusions and open problems

Bubbles in De Bruijn graphs represent interesting biological events, like alternative splicing and allelic differences (SNPs and indels). However, the set of all bubbles in a De Bruijn graph built from real data is usually too large to be efficiently enumerated and analysed. To tackle this issue, in this paper we have proposed a bubble generator, which is a polynomial-sized subset of the bubble space that can be used to generate all and only the bubbles in a directed graph. In particular, we have presented efficient algorithms to identify, for any given directed graph G, a generator set of bubbles $\mathcal{G}(G)$, and to decompose any bubble B in G into bubbles from $\mathcal{G}(G)$. Concerning the applications of the bubble generator, we showed its usefulness in analysing RNA data. In

particular, we indicated that our bubble generator can be used in addition to KisSplice to find AS events corresponding to bubbles with a high branching number.

Our work raises several open theoretical questions. First, our generator $\mathcal{G}(G)$ is not necessarily minimal, *i.e.* it might happen that there exists three bubbles $B_1, B_2, B_3 \in \mathcal{G}(G)$ such that $B_1 < B_3$, $B_2 < B_3$, and $B_3 = B_1 + B_2$. Is it possible to find in polynomial time a generator $\mathcal{G}'(G)$ that is minimal? Second, it seems natural to ask whether all minimal generators for bubbles in directed graphs have the same cardinality. Third, it would be interesting to find a generator $\mathcal{G}(G)$ with some additional biologically motivated constraints, as for example the maximum length of the legs of a bubble [18]. Given an integer k and a graph G, is it possible to find a generator $\mathcal{G}(G)$ that generates all and only the bubbles of G which have both legs of length at most k? Fourth, are there faster algorithms to find a bubble generator? Fifth, this work is related to the research done in the direction of cycle bases. However, as we already mentioned, our problem displays characteristics that make it very different from the ones related to cycle bases. Thus, it may be of independent interest to further investigate the connections between those two problems.

There are also some practical questions that need to be addressed in future work, and which might be interesting on their own. We see three possible directions: (i) reduce the false positive AS events by adding more biologically motivated constraints (*e.g.* the ones mentioned in the previous paragraph) to the bubbles in the generator, (ii) find "complex" AS events by listing also the bubbles that result from a combination of two or more bubbles from the generator.

Finally, our polynomial-time decomposition algorithm could be useful in the case where we want to identify and decompose complex alternative splicing events [19] into their elementary parts. We defer all those problems to further investigations.

Acknowledgments

V. Acuña is supported by Fondecyt 1140631, Center for Genome Regulation FONDAP 15090007, Basal Grant of the Center for Mathematical Modeling UMI2807 UCHILE-CNRS N PFB03 project. R. Grossi and G. F. Italiano are partially supported by MIUR, the Italian Ministry of Education, University and Research, under the Project AMANDA (Algorithmics for MAssive and Networked DAta). Part of this work was done while G. F. Italiano was visiting Université de Lyon. L. Lima is supported by the Brazilian Ministry of Science, Technology and Innovation (in portuguese, Ministério da Ciência, Tecnologia e Inovação - MCTI) through the National Counsel of Technological and Scientific Development (in portuguese, Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq), under the Science Without Borders (in portuguese, Ciências Sem Fronteiras) scholarship grant process number 203362/2014-4. B. Sinaimeri, L. Lima and M.-F. Sagot are partially funded by the French

ANR project Aster (2016-2020), and together with V. Acuña, also by the Stic AmSud project MAIA (2016-2017). This work was performed using the computing facilities of the CC LBBE/PRABI.

References

1. Benoit-Pilven, C., Marchet, C., Chautard, E., Lima, L., Lambert, M.P., Sacomoto, G., Rey, A., Cologne, A., Terrone, S., Dulaurier, L., Claude, J.B., Bourgeois, C., Auboeuf, D., Lacroix, V.: Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data. Scientific Reports **8**(1) (2018)
2. Birmelé, E., Crescenzi, P., Ferreira, R., Grossi, R., Lacroix, V., Marino, A., Pisanti, N., Sacomoto, G., Sagot, M.F.: Efficient Bubble Enumeration in Directed Graphs. In: SPIRE, pp. 118–129 (2012)
3. Bollobás, B.: Modern graph theory, *Graduate Texts in Mathematics*, vol. 184. Springer-Verlag, Berlin (1998)
4. Brankovic, L., Iliopoulos, C.S., Kundu, R., Mohamed, M., Pissis, S.P., Vayani, F.: Linear-time superbubble identification algorithm for genome assembly. Theoretical Computer Science **609**, 374–383 (2016)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Electrical Engineering and Computer Science Series. MIT Press, Cambridge, MA (1991)
6. Deo, N.: Graph theory with applications to engineering and computer science. Prentice-Hall series in automatic computation. Englewood Cliffs, N.J. Prentice-Hall (1974)
7. Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., Gingeras, T.R.: Star: ultrafast universal rna-seq aligner. Bioinformatics **29**(1), 15–21 (2013). DOI 10.1093/bioinformatics/bts635
8. Gleiss, P.M., Leydold, J., Stadler, P.F.: Circuit bases of strongly connected digraphs. Discussiones Mathematicae Graph Theory **23**(2), 241–260 (2003)
9. Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., McVean, G.: De novo assembly and genotyping of variants using colored de bruijn graphs. Nat Genet **44**(2), 226–232 (2012)
10. Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., Zweig, K.A.: Cycle bases in graphs characterization, algorithms, complexity, and applications. Computer Science Review **3**(4), 199 – 243 (2009). DOI http://dx.doi.org/10.1016/j.cosrev.2009.08.001
11. Kavitha, T., Mehlhorn, K.: Algorithms to compute minimum cycle bases in directed graphs. Theory of Computing Systems **40**(4), 485 – 505 (2007)
12. Lima, L., Sinaimeri, B., Sacomoto, G., Lopez-Maestre, H., Marchet, C., Miele, V., Sagot, M.F., Lacroix, V.: Playing hide and seek with repeats in local and global de novo transcriptome assembly of short rna-seq reads. Algorithms Mol Biol **12**, 2–2 (2017). DOI 10.1186/s13015-017-0091-2
13. MacLane, S.: A combinatorial condition for planar graphs. Fundamenta Mathematicae **28**, 22–32 (1937)
14. Miller, J.R., Koren, S., Sutton, G.: Assembly algorithms for next-generation sequencing data. Genomics **95**(6), 315–327 (2010)
15. Onodera, T., Sadakane, K., Shibuya, T.: Detecting Superbubbles in Assembly Graphs. In: Algorithms in Bioinformatics, *Lecture Notes in Computer Science*, vol. 8126, pp. 338–348. Springer Berlin Heidelberg (2013)
16. Pevzner, P.A., Tang, H., Tesler, G.: De Novo Repeat Classification and Fragment Assembly. Genome Research **14**(9), 1786–1796 (2004)
17. Sacomoto, G., Kielbassa, J., Chikhi, R., Uricaru, R., Antoniou, P., Sagot, M.F., Peterlongo, P., Lacroix, V.: Kissplice: de-novo calling alternative splicing events from rna-seq data. BMC Bioinformatics **13**(S-6), S5 (2012)
18. Sacomoto, G., Lacroix, V., Sagot, M.F.: A polynomial delay algorithm for the enumeration of bubbles with length constraints in directed graphs and its application to the detection of alternative splicing in RNA-seq data. In: WABI, pp. 99–111 (2013)

19. Sammeth, M.: Complete alternative splicing events are bubbles in splicing graphs. Journal of Computational Biology **16**(8), 1117–1140 (2009)
20. Shilov, G.E.: Linear Algebra. Dover Publications, New York (1977). (Trans. R. A. Silverman)
21. Simpson, J.T., Wong, K., Jackman, S.D., Schein, J.E., Jones, S.J.M., Birol, I.: ABySS: A parallel assembler for short read sequence data. Genome Research **19**(6), 1117–1123 (2009)
22. Sung, W.K., Sadakane, K., Shibuya, T., Belorkar, A., Pyrogova, I.: An o(m log m)-time algorithm for detecting superbubbles. IEEE/ACM Trans. Comput. Biol. Bioinformatics **12**(4), 770–777 (2015)
23. Uricaru, R., Rizk, G., Lacroix, V., Quillery, E., Plantard, O., Chikhi, R., Lemaitre, C., Peterlongo, P.: Reference-free detection of isolated SNPs. Nucleic Acids Research **43**(2), e11 (2015)
24. Younsi, R., MacLean, D.: Using 2k+2 bubble searches to find single nucleotide polymorphisms in k-mer graphs. Bioinformatics **31**(5), 642–646 (2015)
25. Zerbino, D., Birney, E.: Velvet: Algorithms for De Novo Short Read Assembly Using De Bruijn Graphs. Genome Res. (2008)

Chapter 6

A fast and agnostic method for bacterial genome-wide association studies: bridging the gap between k-mers and genetic events

Preamble

Key points

- The most common approaches for GWAS are unsuitable when working on bacterial species with a large accessory genome. They might be unable to cover variants in noncoding regions, and the analysis can be compromised on species without a good annotation;

- Recent studies have relied on k-mers, which can reflect most genomic variations in a panel. A k-mer representation often loses in interpretability what it gains in flexibility;

- This work bridges the gap between, on the one hand, SNP- and gene-based representations lacking the right level of flexibility to cover complete genomic variations, and, on the other hand, k-mer-based representations which are flexible but not readily interpretable;

- We rely on compacted DBGs to eliminate local redundancy, and reflect genomic variations. Each variant is then tested for association with the phenotype using a linear mixed model, adjusting for the population structure. The unitigs found to be phenotype-associated are then localised in the cDBG, and their neighbourhood is used as a proxy for their genomic environment at the population level. Subgraphs induced by their genomic environment are extracted, which often provide a direct

interpretation in terms of genetic events, aided by their topology, metadata and optional annotation;

- This approach makes GWAS more amenable to bacterial panels and it was effective in catching the dynamics of mobile genetic elements in *Staphylococcus aureus* and *Pseudomonas aeruginosa* genomes, and retrieved known local polymorphisms in *Mycobacterium tuberculosis* genomes.

Status

Published in journal *PLOS Genetics* [48].

Author contributions

M. Jaillard and *L.* share first authorship in this paper.

PLOS GENETICS

RESEARCH ARTICLE

A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events

Magali Jaillard[1,2☯]*, Leandro Lima[2,3☯], Maud Tournoud[1], Pierre Mahé[1], Alex van Belkum[1], Vincent Lacroix[2,3], Laurent Jacob[2]

1 bioMérieux, Marcy l'Étoile, France, **2** Univ Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Evolutive UMR5558 F-69622 Villeurbanne, France, **3** EPI ERABLE - Inria Grenoble, Rhône-Alpes, France

☯ These authors contributed equally to this work.
* magali.dancette@biomerieux.com

Citation: Jaillard M, Lima L, Tournoud M, Mahé P, van Belkum A, Lacroix V, et al. (2018) A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events. PLoS Genet 14(11): e1007758. https://doi.org/10.1371/journal.pgen.1007758

Editor: Xavier Didelot, Imperial College London, UNITED KINGDOM

Received: June 4, 2018

Accepted: October 12, 2018

Published: November 12, 2018

Data Availability Statement: We put online all the GWAS results generated by our method which are discussed in the manuscript (http://pbil.univ-lyon1.fr/datasets/DBGWAS_support/). The proposed method is available on gitlab: https://gitlab.com/leoisl/dbgwas/.

Funding: MJ, MT, PM and AvB are employees of bioMérieux. LL is funded by the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico – CNPq, Brazil, under the Science Without Borders

Abstract

Genome-wide association study (GWAS) methods applied to bacterial genomes have shown promising results for genetic marker discovery or detailed assessment of marker effect. Recently, alignment-free methods based on k-mer composition have proven their ability to explore the accessory genome. However, they lead to redundant descriptions and results which are sometimes hard to interpret. Here we introduce DBGWAS, an extended k-mer-based GWAS method producing interpretable genetic variants associated with distinct phenotypes. Relying on compacted De Bruijn graphs (cDBG), our method gathers cDBG nodes, identified by the association model, into subgraphs defined from their neighbourhood in the initial cDBG. DBGWAS is alignment-free and only requires a set of contigs and phenotypes. In particular, it does not require prior annotation or reference genomes. It produces subgraphs representing phenotype-associated genetic variants such as local polymorphisms and mobile genetic elements (MGE). It offers a graphical framework which helps interpret GWAS results. Importantly it is also computationally efficient—experiments took one hour and a half on average. We validated our method using antibiotic resistance phenotypes for three bacterial species. DBGWAS recovered known resistance determinants such as mutations in core genes in *Mycobacterium tuberculosis*, and genes acquired by horizontal transfer in *Staphylococcus aureus* and *Pseudomonas aeruginosa*—along with their MGE context. It also enabled us to formulate new hypotheses involving genetic variants not yet described in the antibiotic resistance literature. An open-source tool implementing DBGWAS is available at https://gitlab.com/leoisl/dbgwas.

Author summary

Genome-wide association studies (GWAS) help explore the genetic bases of phenotype variation in a population. Our objective is to make GWAS amenable to bacterial genomes. These genomes can be too different to be aligned against a reference, even within a single

scholarship grant process number 203362/2014-4. VL is funded by the Agence Nationale de la Recherche ANR-12-BS02-0008 (Colib'read) and ANR-16-CE23-0001 (ASTER). LJ is funded by the Agence Nationale de la Recherche ANR-14-CE23-0003-01 (MACARON) and ANR-17-CE23-0011-01 (FAST-BIG). This work was performed using the computing facilities of the CC LBBE/PRABI. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: I have read the journal's policy and the authors of this manuscript have the following competing interests: MJ, MT, PM and AvB are employees of bioMérieux, a company that develops and sells diagnostic tests in the field of infectious diseases. However, the study was designed and executed in an open manner and the presented method as well as all data generated have been deposited in the public domain, also resulting in the current publication.

species, making the description of their genetic variation challenging. We test the association between the phenotype and the presence in the genomes of DNA subsequences of length k – the so-called k-mers. These k-mers provide a versatile descriptor, allowing to capture genetic variants ranging from local polymorphisms to insertions of large mobile genetic elements. Unfortunately, they are also redundant and difficult to interpret. We rely on the compacted De Bruijn graph (cDBG), which represents the overlaps between k-mers. A single cDBG is built across all genomes, automatically removing the redundancy among consecutive k-mers, and allowing for a visualisation of the genomic context of the significant ones. We provide a computationally efficient and user-friendly implementation, enabling non-bioinformaticians to carry out GWAS on thousands of isolates in a few hours. This approach was effective in catching the dynamics of mobile genetic elements in *Staphylococcus aureus* and *Pseudomonas aeruginosa* genomes, and retrieved known local polymorphisms in *Mycobacterium tuberculosis* genomes.

Introduction

The aim of Genome-Wide Association Studies (GWAS) is to identify associations between genetic variants and a phenotype observed in a population. They have recently emerged as an important tool in the study of bacteria, given the availability of large panels of bacterial genomes combined with phenotypic data [1–7].

GWAS rely on a representation of the genomic variation as numerical factors. The most common approaches are based on single nucleotide polymorphisms (SNPs), defined by aligning all genomes of the studied panel against a reference genome [1, 3, 4] or against a pangenome built from all the genes identified by annotating the genomes [8], and on gene presence/absence, using a pre-defined collection of genes [5, 7]. The use of a reference genome becomes unsuitable when working on bacterial species with a large accessory genome—the part of the genome which is not present in all strains. On the other hand, methods focusing on genes are unable to cover variants in noncoding regions, including those related to transcriptional and translational regulation [9, 10]. Moreover, some poorly studied species still lack a representative annotation [11].

To circumvent these issues and make bacterial genomes amenable to GWAS, recent studies have relied on k-mers: all nucleotide substrings of length k found in the genomes [2, 5, 6]. The presence of k-mers in genomes can account for diverse genetic events such as the acquisition of SNPs, (long) insertions/deletions and recombinations. Unlike SNP- or gene-based approaches, k-mer analyses do not require a reference genome or any assumption on the nature of the causal variants and can even be performed without assembling the genome sequences [12].

While k-mers can reflect any genomic variation in a panel, they do not themselves represent biological entities. Translating the result of a k-mer-based GWAS into meaningful genetic variants typically requires mapping a large and redundant set of short sequences [2, 5, 6, 13]. Recent studies have suggested reassembling the significantly associated k-mers to reduce redundancy and retrieve longer marker sequences [6, 13]. Nonetheless, k-mer representation often loses in interpretability what it gains in flexibility, and the best way to encode the genomic variation in bacterial GWAS is not yet clearly defined [14, 15].

Our approach, coined DBGWAS, for *De Bruijn Graph GWAS*, bridges the gap between, on the one hand, SNP- and gene-based representations lacking the right level of flexibility to cover complete genomic variation, and, on the other hand, k-mer-based representations

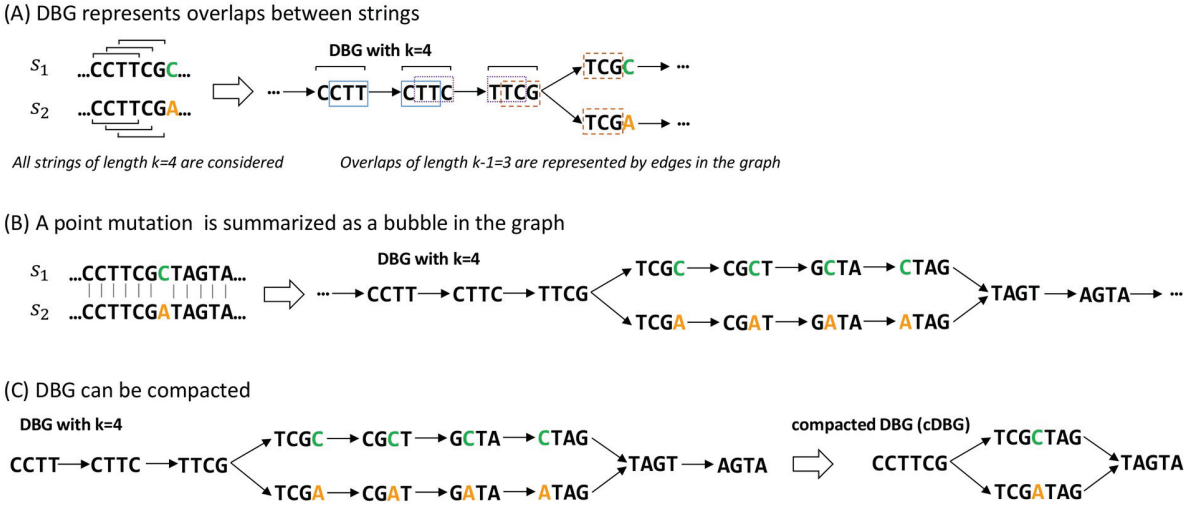


Fig 1. Compacted DBG construction over a set of sequences differing by a single point mutation. In this example two sequences s_1 and s_2 of length 12 differ by a single letter. (A) All k-mers ($k = 4$) present in these sequences are listed. A link is drawn between two k-mers when the $k - 1 = 3$ last nucleotides of the first k-mer equal the 3 first nucleotides of the second k-mer. (B) The bubble pattern represents the SNP C to A; each branch of the bubble represents an allele. (C) Linear paths of the graph are compacted; the compacted DBG of the example only contains four nodes (unitigs) and represents the same variation as the original DBG, which contained 13 nodes (k-mers).

https://doi.org/10.1371/journal.pgen.1007758.g001

which are flexible but not readily interpretable. We rely on De Bruijn graphs [16] (DBGs), which are widely used for *de novo* genome assembly [17, 18] and variant calling [12, 19]. These graphs connect overlapping k-mers (here DNA fragments), yielding a compact summary of all variations across a set of genomes. Fig 1 illustrates the construction of such a graph for a simple example, where the only variation among the aligned genomes is a point mutation. DBGs also accommodate more complex disparities including rearrangements and insertions/deletions (S1 Fig).

DBGWAS relies on the ability of compacted DBGs (cDBGs) to eliminate local redundancy, reflect genomic variations, and characterise the genomic environment of a k-mer at the population level. More precisely, we build a single cDBG from all the genomes included in the association study (in practice, up to thousands). The graph nodes—called unitigs—represent, by construction, sequences of variable length and are at the right level of resolution for the set of genomes considered, taking into account adaptively the genomic variation. The unitigs are individually tested for association with the phenotype, while controlling for population structure. The unitigs found to be phenotype-associated are then localised in the cDBG. Subgraphs induced by their genomic environment are extracted. They often provide a direct interpretation in terms of genetic events which results from the integration of three types of information: 1) the *topology* of the subgraph, reflecting the nature of the genetic variant, 2) the *metadata* represented by node size and colour, allowing us to identify which unitigs in the subgraph are associated to a particular phenotype status, and 3) an optional sequence *annotation* helping to detect unitig mapping to—or near—a known gene.

We benchmarked our novel method using several antibiotic resistance phenotypes within three bacterial species of various degrees of genome plasticity: *Mycobacterium tuberculosis*, *Staphylococcus aureus* and *Pseudomonas aeruginosa*. The subgraphs built from significant unitigs described SNPs or insertions/deletions in both core and accessory regions, and were consistent with results obtained with a resistome-based association study. In addition, novel genotype-to-phenotype associations were also suggested.

Results

We developed DBGWAS, available at https://gitlab.com/leoisl/dbgwas, and validated it on panels for several bacterial species for which genome sequences and antibiotic resistance phenotypes were available. DBGWAS comprises three main steps: it first builds a variant matrix, where each variant is a pattern of presence/absence of unitigs in each genome. Each variant is then tested for association with the phenotype using a linear mixed model, adjusting for the population structure. Finally, it uses the cDBG neighbourhood of significantly associated unitigs as a proxy for their genomic environment. DBGWAS outputs a set of such subgraphs ordered by $\min_q$, which is the smallest q-value observed over unitigs in each subgraph. The top subgraphs therefore represent the genomic environment of the unitigs most significantly associated with the tested phenotype. Fig 2 summarises the main steps of the process. A detailed description of the pipeline is presented in the Methods section.

Here we rely on a few experiments to illustrate how the subgraphs output by DBGWAS can be read as genetic events. We then benchmark DBGWAS against two other k-mer-based approaches and one resistome-based approach. DBGWAS recovers known variants, while suggesting novel candidates out of the range of the resistome-based approach. We also find it to be more computationally efficient and to provide more interpretable outputs than the other k-mer-based methods.

A synthetic description of the discussed subgraphs is provided in Table 1, while a description of the top subgraphs obtained for all tested antibiotics is provided in S3, S4, and S5 Tables.

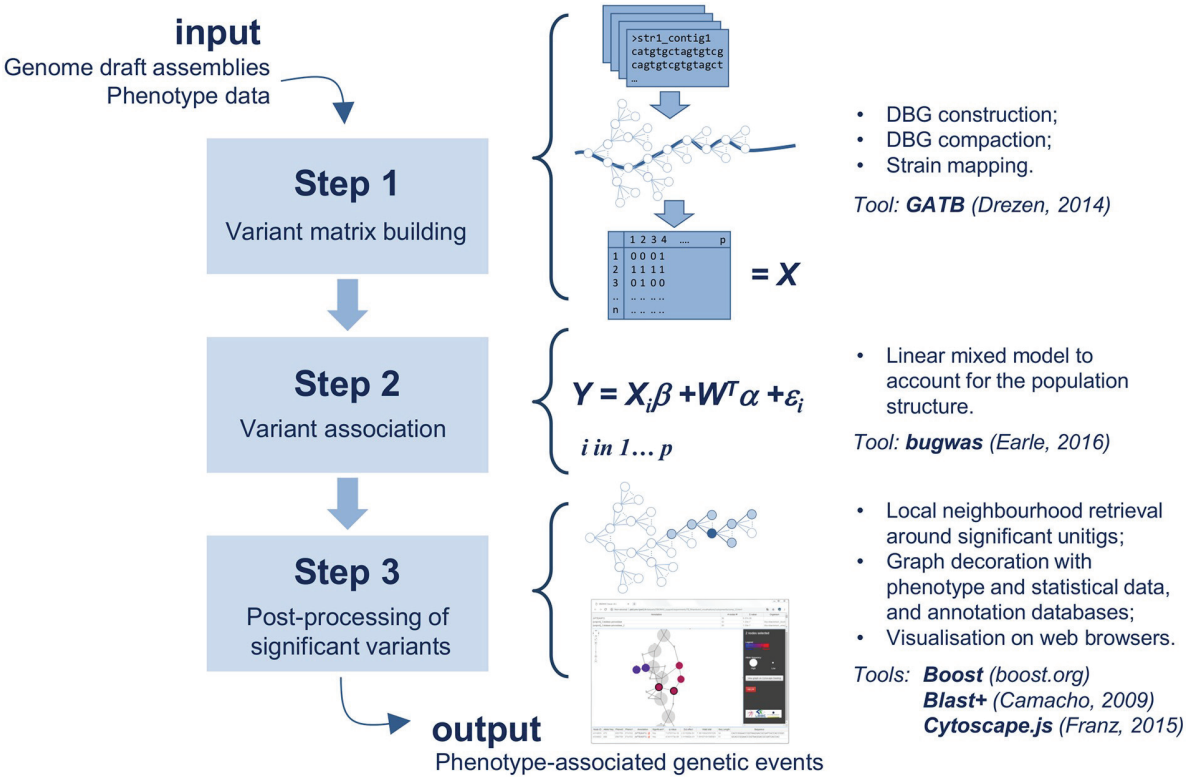


Fig 2. DBGWAS pipeline. DBGWAS takes as input draft assemblies and phenotype data for a panel of bacterial strains. A variant matrix X is built in *step* 1 using cDBG nodes (called unitigs). Variants are tested in *step* 2 using a linear mixed model taking into account the population structure. Significant variants are post-processed in *step* 3 to provide an interactive interface assisting their interpretation.

https://doi.org/10.1371/journal.pgen.1007758.g002

Table 1. Resistance determinants identified by DBGWAS for *S. aureus* (SA), *M. tuberculosis* (TB) and *P. aeruginosa* (PA) panels.

Panel	Phenotype	Rank	Sign. unitigs	min_q	Est. effect	Annotation	Type	Knowledge on markers
SA	Methicillin	1	71/565	7.68×10^{-188}	0.949	*mecA* + 7000 bp of SC *Cmec*	MGE	Pos
		2	99/735	3.39×10^{-72}	0.865	6000 bp of SCC*mec*	MGE	$r^2 = 0.96$
		3	11/190	2.14×10^{-61}	0.813	2000 bp of SCC*mec*	MGE	$r^2 = 0.94$
		4	13/117	2.29×10^{-37}	0.957	1500 bp of SCC*mec*	MGE	$r^2 = 0.93$
	Ciprofloxacin	1	7/57	8.67×10^{-104}	-0.893	*par*CQRDR	LPG	Pos
		2	7/31	2.21×10^{-76}	0.955	*gyrA*QRDR	LPG	Pos
	Erythromycin	1	110/510	2.69×10^{-100}	0.823	*ermC* + circular plasmid	MGE	Pos
	Fusidic acid	1	7/50	2.75×10^{-136}	-0.910	*fusA*	LPG	Pos
		2	214/882	7.94×10^{-49}	0.924	*fusC* + SCC *fusC*cassette	MGE	Pos
		3	22/260	5.35×10^{-43}	0.924	1,500 bp of SCCfusC	MGE	$r^2 = 0.98$
		3	1/72	5.35×10^{-43}	0.924	200 bp of SCC*fusC*	MGE	$r^2 = 0.98$
		5	5/64	2.02×10^{-22}	-0.888	*purN*	LPG	$r^2 = 2 \times 10^{-3}$
	Trimethoprim	1	7/54	8.38×10^{-24}	0.969	*folA*	LPG	Pos
		2	3/41	9.30×10^{-18}	-0.966	btw. hyp. prot. & VOC prot.	LPN	$r^2 = 0.19$
		3	11/70	9.30×10^{-18}	-0.966	*ybaK*	LPG	$r^2 = 0.44$
		4	2/30	6.82×10^{-10}	-0.632	*mqo1*	LPG	$r^2 = 0.29$
	Gentamicin	1	173/1193	1.30×10^{-205}	0.873	*aac(6')*gene within a plasmid	MGE	Pos
		2	127/367	9.02×10^{-75}	0.751	seq. of plasmid carrying *aac(6')*	MGE	$r^2 = 0.38$
		3	2/23	9.01×10^{-53}	0.634	seq. of plasmid carrying *aac(6')*	MGE	$r^2 = 0.40$
		4	1/29	1.04×10^{-40}	0.579	seq. of plasmid carrying *aac(6')*	MGE	$r^2 = 0.48$
		5	2/56	1.49×10^{-33}	-0.831	*odhB*	LPG	$r^2 = 8 \times 10^{-5}$
TB	Rifampicin	1	36/115	4.84×10^{-70}	-0.577	*rpoB*RRDR	LPG	Pos
		2	6/37	4.35×10^{-20}	-0.355	*katG*	LPG	CR
		3	5/41	4.02×10^{-8}	-0.224	*embB*M306V	LPG	Pos
	Streptomycin	1	5/30	3.70×10^{-31}	0.544	*rpsL*(30S ribos.protein S12)	LPG	Pos
		2	6/37	1.06×10^{-28}	-0.428	*katG*	LPG	CR
		3	25/113	2.87×10^{-16}	-0.339	*rpoB*RRDR	LPG	CR
		4	6/45	1.40×10^{-9}	-0.271	*embB*M306V	LPG	CR
		5	8/31	2.86×10^{-9}	-0.535	*rrs*, 16S rRNA C517T	LPG	Pos
		6	13/69	9.18×10^{-5}	-0.216	*gyrA*QRDR	LPG	CR
		7	2/20	1.20×10^{-3}	0.739	*espG1*	LPG	$r^2 = 3 \times 10^{-3}$
	Ofloxacin	1	31/85	9.66×10^{-144}	-0.888	*gyrA*QRDR	LPG	Pos
		2	9/68	1.59×10^{-4}	0.507	*ubiA*(Rv3806c)	LPG	CR
		3	3/32	3.86×10^{-2}	-0.746	Rv3909	LPG	$r^2 = 9 \times 10^{-3}$
	Ethionamide	1	9/39	7.86×10^{-11}	-0.462	*fabG1*promoter	LPN	Pos
		2	15/47	5.16×10^{-10}	-0.406	*gyrA*QRDR	LPG	CR
		3	4/26	5.55×10^{-4}	0.319	*rrs*, 16S rRNA A1401G	LPG	CR
	XDR	1	6/68	3.66×10^{-39}	0.905	*rpoB*I1187T (out. RRDR)	LPG	Ukn
		1	3/27	3.66×10^{-39}	0.905	Rv2000	LPG	$r^2 = 1$
		3	3/24	9.58×10^{-36}	0.883	*espA*promoter	LPN	$r^2 = 0.98$
PA	Amikacin	1	4/83	5.86×10^{-9}	0.621	SNP in *aac(6')*	LPG	Pos
		2	3/82	1.37×10^{-6}	0.662	DEAD/DEAH box helicase	LPG	$r^2 = 0.55$
		3	38/315	2.21×10^{-6}	0.523	plasmid mapping on pHS87b	MGE	$r^2 = 0.17$
	Levofloxacin	1	5/27	7.21×10^{-29}	-0.884	*gyrA*QRDR	LPG	Pos
		2	5/29	5.68×10^{-6}	-0.737	*par*CQRDR	LPG	Pos
		3	5/38	1.87×10^{-2}	0.688	Histidine kinase/response regulator	LPG	$r^2 = 0.17$

For each antibiotic, we report subgraphs with their rank, number of significant unitigs over all unitigs in the subgraph (Sign. unitigs), q-value of the unitig with the lowest q-value (min_q), the corresponding estimated effect ($\hat{\beta}$ coefficient of the linear mixed model) and annotation of the subgraph. The type of event represented by the subgraph is colour-coded as: yellow for MGE, light blue for local polymorphism in gene (LPG), and dark blue for local polymorphism in noncoding region (LPN). Known resistance markers are indicated in dark green (Pos), determinants whose presence was described to be caused by co-resistance in orange (CR), unknown variants arriving at the first rank in grey (Ukn). For other subgraphs, an r^2 value relative to the first subgraph is provided as an estimation of linkage disequilibrium with the first subgraph. It was computed between the most significant patterns of the first and the considered subgraphs.

https://doi.org/10.1371/journal.pgen.1007758.t001

The subgraphs themselves are available at http://pbil.univ-lyon1.fr/datasets/DBGWAS_support/experiments/#DBGWAS_all_results.

Coloured bubbles highlight local polymorphism in core genes, accessory genes and noncoding regions

For *P. aeruginosa* levofloxacin resistance, the subgraph obtained with the lowest $\min_q$ highlighted a polymorphic region in a core gene (Fig 3A). Indeed, it showed a linear structure containing a complex bubble, with a fork separating susceptible (blue) and resistant (red) strains. The annotation revealed that all unitigs in this subgraph mapped to the quinolone resistance-determining region (QRDR) of the *gyrA* gene. *gyrA* codes for a subunit of the DNA gyrase targeted by quinolone antibiotics such as levofloxacin and its alteration is therefore a prevalent and efficient mechanism of resistance [20, 21]. In all our experiments related to quinolone resistance, DBGWAS identified QRDR mutations in either *gyrA* or *parC*, which codes for another well-known quinolone target: *P. aeruginosa* levofloxacin (first subgraph, *gyrA*: $\min_q = 7.21 \times 10^{-29}$ and second, *parC*: 5.68×10^{-06}), *S. aureus* ciprofloxacin (first, *parC*: $\min_q = 8.67 \times 10^{-104}$ and second, *gyrA*: 2.21×10^{-76}), and ofloxacin resistance in *M. tuberculosis*, whose genome does not contain the *parC* gene [22] (first, *gyrA*: $\min_q = 9.66 \times 10^{-144}$).

For *P. aeruginosa* amikacin resistance, the top subgraph ($\min_q = 5.86 \times 10^{-9}$) highlighted a SNP in an accessory gene (Fig 3B). As in Fig 3A, it contained a fork separating a blue and a red node. However, other remaining nodes were not grey: they represented an accessory sequence because they were not present in all the strains. Most of these nodes were pale-red, showing that the accessory sequence was more frequent in resistant samples. The annotation revealed that this subgraph corresponded to *aac(6')*, a gene coding for an aminoglycoside 6-acetyltransferase, an enzyme capable of inactivating aminoglycosides, such as amikacin, by acetylation [23]. Most unitigs in this gene had a low association with resistance, except for the ones describing this particular SNP. Mapping the sequence of these unitigs on the UniProt database [24] revealed an amino-acid change at L83S, right in the enzyme binding site. This SNP was previously shown to be responsible for substrate specificity alteration in a strain of *Pseudomonas fluorescens* [25]. It appears to increase the amikacin acetylation ability of *aac(6')*, making its association to amikacin resistance more significant than the gene presence itself.

Finally, for *M. tuberculosis* ethionamide resistance, the top subgraph ($\min_q = 7.86 \times 10^{-11}$, Fig 3C) represented a polymorphic region in a core gene promoter. The subgraph was mostly grey and linear with a localised blue and red fork. The most reliable annotation for this subgraph was *fabG1* (also known as *mabA*), a core gene previously shown to be involved in ethionamide and isoniazid resistance [26, 27]. None of the significantly associated unitigs mapped to the *fabG1* gene, but their close neighbours did (highlighted in Fig 3C by black circles), suggesting that the detected variant was located in the promoter region of the gene. This was confirmed by mapping the significant unitig sequences using the Tuberculosis Mutation database of the *mubii* resource [28].

Long single-coloured paths denote mobile genetic element insertions

For *S. aureus* resistance to methicillin, the top subgraph ($\min_q = 7.68 \times 10^{-188}$), shown in Fig 3D, revealed a gene cassette insertion. It contained a long path of red nodes, and a branching region including another red node path. The first path mapped to the *mecA* gene, extensively described in this context and known to be carried by the Staphylococcal Cassette Chromosome *mec* (SCC*mec*) [21, 29, 30]. The other part of the subgraph represented a >5,000 bp fragment of the cassette. It was less linear because it summarised several types of the cassette differing by their structure and gene content [29]. The next subgraphs represented other regions of the

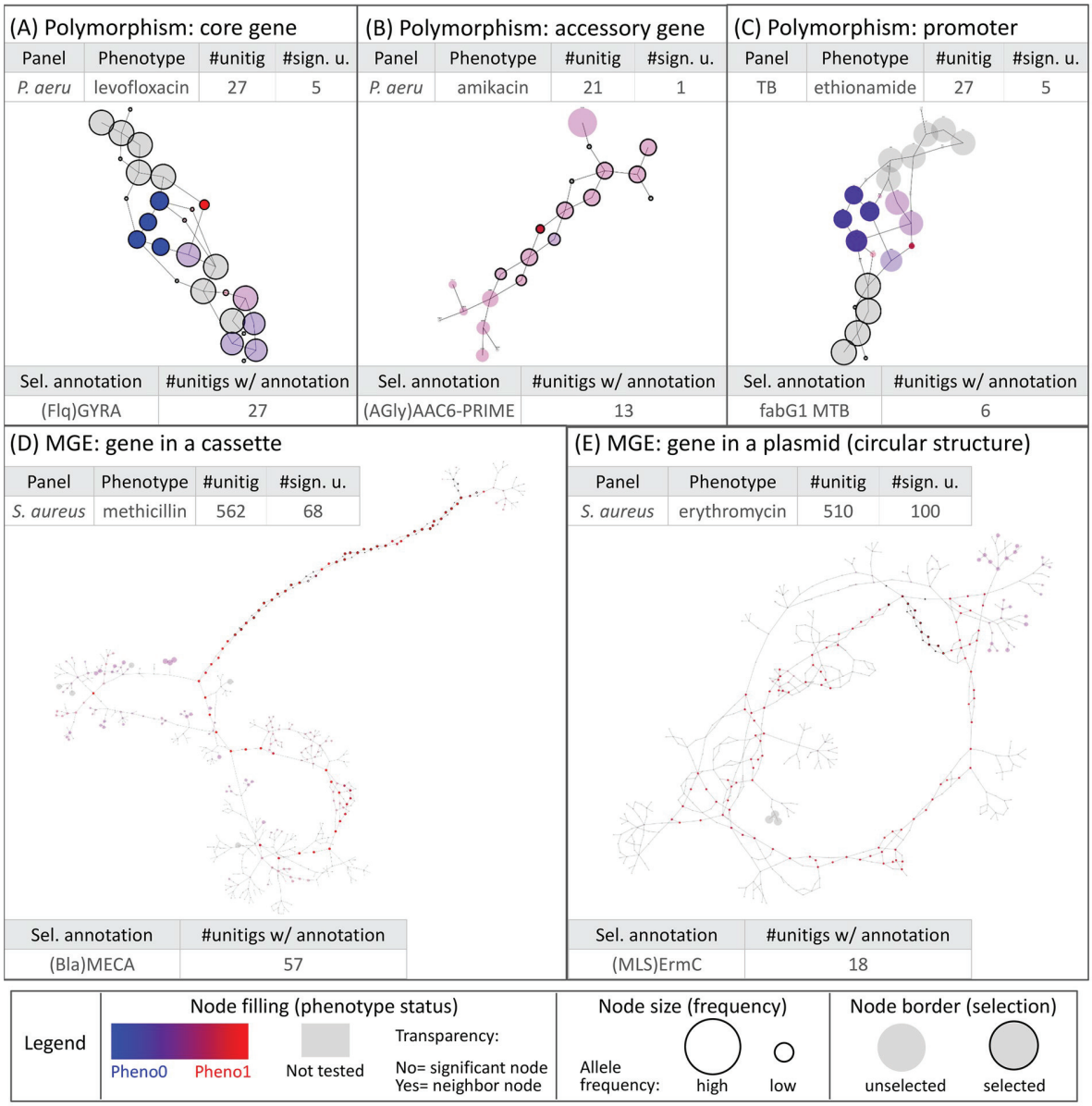


Fig 3. Different types of genetic events identified by DBGWAS. Each subgraph represents a distinct genetic event. Colours are continuously interpolated between blue for susceptible unitigs and red for resistant ones. Untested unitigs, present in $> 99\%$ or $< 1\%$ of the strains, are shown in grey. Nodes found to be not significative are shown with a transparency degree. The node size relates to its allele frequency: the larger the node, the higher the allele frequency. Circled black nodes map to annotated genes. The two tables in each panel provide information on the sugraph nodes. As an example, the subgraph in panel (A) is composed of 27 unitigs, 5 of which were significantly associated with resistance. All unitigs of this subgraph mapped to the *gyrA* gene. The subgraphs presented in the four other panels correspond to the top subgraphs (with lowest $\min_q$) obtained for different panels/phenotypes. All subgraphs are snapshots taken from DBGWAS interactive visualisation and are available online.

https://doi.org/10.1371/journal.pgen.1007758.g003

same cassette. Interestingly, retaining a greater number of unitigs to build the subgraphs leads to merging these individual subgraphs, representing related genomic regions, into a single one. This can be done by increasing the Significant Features Filter (*SFF*) parameter value, which defines the unitigs used to build the subgraphs. By default, the unitigs corresponding to

the 100 lowest q-values are retained (*SFF* = 100). Increasing the *SFF* value to 150 (150th q-value = 1.60×10^{-27}) allowed us to reconstruct the entire SCC*mec* cassette, as shown in S3 Fig.

For *S. aureus* erythromycin resistance, a unique subgraph was generated ($\min_q = 2.69 \times 10^{-100}$). As shown in Fig 3E, the subgraph described the circular structure of a 2,500 bp-long plasmid known to carry the causal *ermC* gene together with a replication and maintenance protein in strong linkage disequilibrium with *ermC* [30, 31].

For *P. aeruginosa* amikacin resistance, the third subgraph ($\min_q = 2.21 \times 10^{-6}$) represented a 10,000 bp plasmid acquisition. Using the NCBI nucleotide database [32], most of the unitigs in this subgraph mapped to the predicted prophage regions of an integrative and conjugative plasmid, whose structure corresponds to a plasmid, pHS87b, recently described in the amikacin resistant *P. aeruginosa* HS87 strain [33]. S4 and S5 Figs provide more examples of MGEs recovered by DBGWAS, and the Interpretation of significant unitigs (step 3) subsection of the Methods section discusses *SFF* default value and tuning.

DBGWAS reports expected variants without prior knowledge

Although resistance determinants are not perfectly or exhaustively known for all species, some resistance mechanisms are well described. This is the case of *gyrA* and *parC* alteration in fluoroquinolone resistance in *P. aeruginosa* [20], and of the alteration of two streptomycin targets: the ribosomal protein S12 (coded by *rpsL*) and the 16S rRNA (coded by *rrs*) in *M. tuberculosis* [34]. Here we verify the ability of bacterial GWAS methods to recover these known mechanisms. We compared DBGWAS results to those obtained by applying the same association model to a collection of known resistance genes and SNPs [7, 35] (see the Resistome-based association studies subsection of the Methods section), and to two other recent k-mer-based methods: pyseer [6, 36], and HAWK [13].

For *P. aeruginosa* levofloxacin resistance (Table 2), both DBGWAS and pyseer identified the two expected known causal determinants reported by the prior resistome-based study: *gyrA* and *parC*, while HAWK only reported *gyrA*. pyseer reported 224 k-mers, all mapping to *gyrA* and *parC*, while the other methods reported less than 10 features (subgraphs or reassembled k-mers), among which were several unknown, potentially new candidate markers.

For *M. tuberculosis* streptomycin resistance (Table 3), the four methods reported the two expected known causal determinants *rpsL* and *rrs*. However, while the resistome-based study

Table 2. Resistance determinants found by the four methods for *P. aeruginosa* levofloxacin resistance.

Legend	resistome-based	DBGWAS	pyseer	HAWK
Time (mem)	37m (7.2 GB)	21m (3.2 GB)	24h22m (14.5 GB)	39m (4.2 GB)
Nb reported	2 variants	5 subgraphs	224 k-mers	8 reassembled k-mers
Known positive	*gyrA* (2.11×10^{-22})	*gyrA* (7.21×10^{-29})	*gyrA* (1.97×10^{-17})	*gyrA* (2.82×10^{-14})
	parC (1.83×10^{-5})	*parC* (5.68×10^{-6})	*parC* (5.68×10^{-9})	
Unknown		HK/RR (1.87×10^{-2})		tnp (1.66×10^{-14})
		tnp		NC near tnp
		topA		

This table presents the annotation of the features identified by the tested methods with default parameters. The total number of reported features, as well as the execution time and memory load (in Gigabytes) are given in the header. For k-mer-based methods, annotations were retrieved by mapping unitig/k-mer sequences to the resistance and Uniprot databases (see Interpretation of significant unitigs (step 3) subsection of the Methods section), and completed when needed by Blast on NCBI Nucleotide database. Green cells correspond to resistance determinants already described in the literature. Grey cells represent unknown determinants. Within each category, annotations are ordered by increasing minimum p/q-values. p/q-values are reported only for the most significant annotations. For each method, the annotation with the lowest p/q-values is underlined. 'NC' means noncoding region and 'tnp' transposase.

https://doi.org/10.1371/journal.pgen.1007758.t002

Table 3. Resistance determinants found by the four methods for *M. tuberculosis* streptomycin resistance.

Legend	resistome-based	DBGWAS	pyseer	HAWK
Time (mem)	1h31m (2.1 GB)	42m (4.3 GB)	14h14m (102.4 GB)	3h01m (3.7 GB)
Nb reported	28 variants	24 subgraphs	85,011 k-mers	2,038 reassembled k-mers
Known positive	*rpsL* (1.96×10^{-33})	*rpsL* (3.70×10^{-31})	*rpsL* (4.85×10^{-55})	*rpsL* (5.72×10^{-47})
	rrs (5.40×10^{-8})	*rrs* (2.86×10^{-9})	*rrs* (1.63×10^{-14})	*rrs* (3.45×10^{-20})
Determinant described for other antibiotics	*katG* (2.61×10^{-30})	*katG* (1.06×10^{-28})	*katG* (2.12×10^{-71})	*katG* (1.44×10^{-57})
	rpoB	*rpoB*	*rpoB*	*embB*
	gidB	*embB*	*embB*	***kasA***
	gyrA	*gyrA*	***ubiA***	***embC***
	embB	*gidB*	*pncA*	*gyrA*
	fabG1 promoter	*rpoC*	*fabG1* promoter	***iniA***
	pncA	*fabG1* promoter	*gyrA*	***embA***
	rpoC	***ubiA***	*gidB*	***embR***
	inhA		***ethA***	*gidB*
			embA	***tsnR***
			embC	*rpoB*
				pncA
				ethA
Unknown (top list)		*espG1* (1.20×10^{-3})	NC near tnp/PE (1.13×10^{-19})	NC near tnp/PPE (2.93×10^{-57})
		rpsN	Rv0270	tnp
		NC near tnp/PPE	Rv2665	Rv2825c/Rv2828c
		rnj	Rv2743c	13E12 repeat family protein
		Rv2672	Rv2522c	PPE
		espA promoter	NC near tnp/PPE	CRISPR repeats, down *Cas* genes
		Rv2456c promoter	*guaA*	*mmpL14*
		whiB6	*kdpD*	*esxM*
		...	...	...

This table presents the annotation of the features identified by the tested methods with default parameters. The total number of reported features, as well as the execution time and memory load (in Gigabytes) are given in the header. For k-mer-based methods, annotations were retrieved by mapping unitig/k-mer sequences to the resistance and Uniprot databases (see Interpretation of significant unitigs (step 3) subsection of the Methods section), and completed when needed by Blast on NCBI Nucleotide database. Green cells correspond to resistance determinants already described in the literature, orange cells to resistance determinants described for association with other antibiotics. The annotations not found by the resistome-based strategy are written in bold. Grey cells represent unknown determinants. Within each category, annotations are ordered by increasing minimum p/q-values. p/q-values are reported only for the most significant annotations. For each method, the annotation with the lowest p/q-values is underlined. 'NC' means noncoding region, 'tnp' transposase, 'PE' stands for PE-family protein and 'PPE' for PPE-family protein.

https://doi.org/10.1371/journal.pgen.1007758.t003

and DBGWAS methods ranked the causal *rpsL* determinant first, pyseer and HAWK reported their lowest p/q-values for the false positive *katG* determinant. *katG* and other false positives caused by co-resistance were among the top-ranked features for all methods and this is a well described phenomenon in *M. tuberculosis* species [34, 37].

Additional results for all antibiotics can be found in S6 and S7 Tables for resistome-based association studies, and in S3 and S5 Tables for DBGWAS.

DBGWAS provides novel hypotheses

In addition to resistance markers, all three k-mer-based approaches reported several unknown variants, not described in the context of resistance. Among them, in the context of streptomycin resistance, a noncoding region between a transposase and a PPE-family protein was

reported by the three methods but, as expected, not by the resistome-based approach, as only resistance genes were included in this analysis. More generally, knowledge-based approaches such as SNP-, gene- or resistome-based GWAS can be limited in the context of new marker discovery, since any causal variant absent from the chosen reference would remain untested. Besides being time-consuming, preparing such a list of genetic variants can be problematic for bacterial species without extensive annotation or reference availability. Here we describe associations identified by DBGWAS and which were never described in the antibiotic resistance literature.

In our *P. aeruginosa* panel, the second subgraph obtained for amikacin resistance ($\min_q = 1.37 \times 10^{-6}$) gathered unitigs mapping to the 3' region of a DEAD/DEAH box helicase, known to be involved in stress tolerance in *P. aeruginosa* [38]. The unitig with the lowest q-value was present in 13 of 47 resistant strains and in only 1 of 233 susceptible strains and represented a C-C haplotype summarising two mutated positions: 2097 and 2103. This annotation was not an artefact of the population structure, properly taken into account by the linear mixed model. Indeed the 13 resistant strains corresponded to distinct clones belonging to two phylogroups, one of them containing the susceptible strain. In *P. aeruginosa* levofloxacin resistance, the third subgraph ($\min_q = 1.87 \times 10^{-2}$) represented a L650M amino-acid change in a hybrid sensor histidine kinase/response regulator. Such two-components regulatory systems play important roles in the adaptation of organisms to their environment, for instance in the regulation of biofilm formation in *P. aeruginosa* [39], and as such may play a role in antibiotic resistance.

In *S. aureus*, polymorphisms within genes not known to be related to resistance were identified for several antibiotics: *purN* ($\min_q = 2.02 \times 10^{-22}$) for fusidic acid, *odhB* ($\min_q = 1.49 \times 10^{-33}$) for gentamicin, *ybaK* and *mqo1* ($\min_q = 9.30 \times 10^{-18}$, resp. 6.82×10^{-10}) for trimethoprim. None of these genes have been associated with antibiotic resistance before, to the best of our knowledge.

In *M. tuberculosis*, polymorphisms in two genes encoding proteins involved in *cell wall and cell processes*, *espG1* and *espA*, were found associated with streptomycin (seventh subgraph, $\min_q = 9.43 \times 10^{-4}$) and XDR phenotype (third subgraph, $\min_q = 9.58 \times 10^{-36}$), respectively. Again, these genes have never been reported in association with antibiotic resistance before.

Although experimental validation would be required to tell whether these hypotheses are false positive (e.g., in linkage with causal variants) or actual resistance mechanisms not yet documented, DBGWAS is a valuable tool to screen for novel candidate markers. Moreover it provides a first level of variant description (SNPs in gene or promoter, MGE, etc) which can directly drive the biological validation.

DBGWAS facilitates the interpretation of k-mer-based GWAS

Other k-mer-based approaches are as agnostic as DBGWAS and were also able to provide novel hypotheses, but interpreting their output can prove more challenging than a SNP/gene-based GWAS. In the *M. tuberculosis* streptomycin resistance experiment for example, they reported several thousands of features, while DBGWAS reported only 24 annotated subgraphs without missing any expected determinant (see Table 3). The thousands of k-mers generated by HAWK and pyseer are of course also amenable to interpretation: to build our Table 3, we mapped these k-mers to references and extracted annotated variants which showed at least one hit. However, doing so required additional efforts and a working knowledge of the most appropriate annotated references. In addition, k-mers which do not map to the chosen reference cannot be interpreted. By contrast, DBGWAS always returns a subgraph containing these k-mers. Even when no annotation exists, the topology and colours of the subgraphs may hint towards the nature of the causal variant.

In addition to providing context for significant k-mers and guiding their interpretation as SNPs or MGEs, DBGWAS clustering of close variants into a subgraph can describe hypervariable regions as single entities, and highlight highly associated haplotypes. As an example, the top subgraph for rifampicin resistance ($\min_q = 4.84 \times 10^{-70}$) contained 36 significant unitigs, distinguishing between susceptible (blue) and resistant (red) strains. Instead of a single point mutation, this subgraph represented a polymorphic region known as the rifampicin resistance-determining region (RRDR) of the *rpoB* gene. The unitig with the lowest q-value covered several mutant positions, defining a particular haplotype strongly associated with rifampicin susceptibility. Where DBGWAS reported in this case only one subgraph, pyseer, for instance, reported 470 k-mers with the *rpoB* annotation, and the resistome-based association study reported in this case 4 distinct SNPs in *rpoB* (S6 Table). In another user-submitted example, DBGWAS identified mosaic alleles of three *pbp* genes involved in beta-lactam resistance of *Streptococcus pneumoniae*. Like in the RRDR example, it returned five subgraphs corresponding to the three genes—three subgraphs were annotated *pbp2x* and represented three distinct polymorphic regions of the gene. Each subgraph summarised the polymorphism of the gene, as opposed to one separate feature for each SNP.

Admittedly, some subgraphs output by DBGWAS are not readily interpretable: they are neither coloured bubbles highlighting SNPs, nor long single-coloured paths denoting MGE insertions. This was the case of several subgraphs produced for *P. aeruginosa* amikacin resistance, and presented in S6 Fig. Genetic variants inserted in variable regions, for example, lead to subgraphs with a high average degree, or to very large subgraphs. The fourth subgraph for instance ($\min_q = 2.21 \times 10^{-6}$) contains a path of three red (positively-associated) nodes lying in a noncoding region between variable accessory genes. Consequently, their neighbour unitigs branch to various other unitigs, making the structure complex and hard to interpret. Complex subgraphs also arise when several associated variants have overlapping neighbourhoods (as defined in the Graph neighbourhoods subsection in the Methods section, and tuned with the *nh* parameter) in at least one strain. This is the case for the subgraph with the smallest $\min_q$ which aggregates *aac*(6′) acetyltransferase and the CML efflux pump.

The interpretation of such subgraphs is not straightforward. We often found it helpful to tune the *nh* and *SFF* parameters to break large subgraphs into a set of smaller ones, as discussed in the discussed in the Methods section. For the *aac*(6′) subgraph, where nearby variants are aggregated into a large subgraph, reducing the *SFF* value to 15 provided a much smaller and easier-to-interpret subgraph focusing on the *aac*(6′) mutation (Fig 3B). Otherwise, we recommend to focus on the topology of the most significant unitigs and their close neighbours.

DBGWAS is fast, memory-efficient, and scales to very large panels

To assess the scalability of DBGWAS to large datasets, we retrieved 5,000 genomes from *M. tuberculosis*, 9,000 genomes from *S. aureus* and 2,500 genomes from *P. aeruginosa*, as described in the Large panels subsection of the Methods section. We present in S9 Fig the runtime and memory usage performances for these panels. All 180 runs took less than 5 days and 250 GB of RAM on 8 cores. Both the computational time and memory usage increase log-linearly with the panel size. Moreover, at equal panel size, DBGWAS performance also depends on the genome complexity, requiring less computational resource for more clonal genomes such as *M. tuberculosis*.

We also compared the computational performance of DBGWAS with pyseer and HAWK. The benchmark was performed on 13 datasets, including one large dataset of 2,500 genomes for each of the 3 species (see the Datasets subsection in the Methods section for details).

Detailed results are presented in S2 Table. DBGWAS was the fastest tool in 11 out of 13 experiments, always taking less than 2 hours. HAWK ran in less than 10 hours in 12 out of 13 experiments, and was a little faster than DBGWAS on two of the large-scale datasets. pyseer took from 13 to 53 hours on 9 experiments, and failed on the 4 others: one exceeded the disk space limit of 1TB, three exceeded the runtime limit of five days. It was brought to our attention during the reviewing process that piping the output of fsm-lite through gzip would decrease the disk space usage. HAWK was more parsimonious in memory usage than DBGWAS on the large scale panels. This can be explained by the fact that the 0.8.3-beta version of HAWK which we are using does not take into account the population structure, and as such does not have to compute an $n \times n$ covariance matrix, providing it a large gain in memory usage—and, to a lesser extent, runtime—for large panels. On the other hand, disregarding the population structure could also lead to spurious discoveries. HAWK v0.9.8-beta offers an adjustment but failed to recover the known true positives, which is why we chose to present the results of the 0.8.3-beta version. DBGWAS and HAWK typically used one order of magnitude less memory than pyseer. The most memory-consuming step for pyseer was the k-mer counting step relying on fsm-lite.

Discussion

In this article we introduce an efficient method for bacterial GWAS. Our method is agnostic: it considers all regions of the genomes and is able to identify potentially new causal variants as different as SNPs in noncoding regions and MGE insertions/deletions. It performs as well as the current SNP- and gene-based gold standard approaches for retrieving known determinants, from genome pre-assemblies and without relying on annotations or reference genomes.

DBGWAS exploits the genetic environment of the significant k-mers through their neighbourhood in the cDBG, providing a valuable interpretation framework. Because it uses only contig sequences as input, it allows GWAS on bacterial species for which the genomes are still poorly annotated or lack a suitable reference genome. DBGWAS makes bacterial GWAS possible in two hours using a single-core computer (see S1 Table), outperforming other state-of-the-art k-mer-based approaches.

Underlying our method, graph-based genome sequence representations such as DBGs, extend the notion of the reference genome to cases where a single sequence stops being an appropriate approximation [40, 41]. As demonstrated in this paper, they pave the way to GWAS on highly plastic bacterial genomes and could also be useful for microbiomes [42] or human tumours [13].

DBGWAS currently relies on the Benjamini-Hochberg procedure to control the FDR and offers no advance exploiting the dependence among presence/absence patterns. An important improvement would be to control the false discovery rate at the subgraph level instead of the unitig level. DBGWAS could be extended to different statistical tasks by adapting its underlying association model, to allow for continuous phenotypes or identify epistatic effects, for instance. The interpretability of the extracted subgraphs could also be improved by training a machine learning model to predict which types of event they represent [43]. This automated labelling could guide users in their interpretation and allow them to search for specific events, such as SNPs in core genes or rearrangements.

Several recent studies describe *in silico* models for defining a genomic antibiogram and hopes are high that such technologies will complement the classic phenotypic methods [44]. Several studies have already demonstrated that in some cases, genomic antibiograms can be at least as good as phenotypic ones [30, 45–47]. Contrary to our approach, these studies require

extensive resistance marker databases. DBGWAS will surely contribute to the extension of such databases or to the development of agnostic genomic antibiograms.

In conclusion, we demonstrate for three medically important bacterial species that resistance markers can be detected rapidly with relative ease, using simple computer equipment. Our integrated software and visualisation tools offer an intuitive variant representation, hence will provide future users with an enhanced insight into genotype to phenotype correlations, in all domains of microbiology, beyond that of antibiotic resistance. This will include complex traits such as biofilm formation, epidemicity and virulence.

Methods

Encoding genomic variation with compacted DBGs

DBGs are directed graphs that efficiently represent all the information contained in a set of sequences. Nodes represent all the unique k-mers (genome sequence substrings of length k) extracted from the input sequences. Edges represent $(k-1)$-exact-overlaps between k-mers: an edge connects a node n_1 to a node n_2 if and only if the $(k-1)$-length-suffix of n_1 equals the $(k-1)$-length-prefix of n_2 (Fig 1A).

These graphs can be compacted into cDBGs by merging linear paths (sequences of nodes not linked to more than two other nodes) into a single node referred to as a *unitig* [48–50] (Fig 1C). Compaction yields a graph with locally optimal resolution: regions of the genome which are conserved across individuals are represented by long unitigs, while regions which are highly variable are fractioned into shorter unitigs (S1 Fig).

Representing strains by their unitig content (step 1)

cDBG construction. We build a single DBG from all genomes given as input using the GATB C++ library [51]. We start from contigs rather than reads and, consequently, we do not need to filter out low abundance k-mers, allowing for the exploration of any variation present in the set of input genomes. We then compact the DBG using a graph traversal algorithm, which identifies all linear paths in the DBG—each forming a unitig in the cDBG. During this step, we also associate each k-mer index to its corresponding unitig index in the cDBG.

There is no general rule for choosing the ideal k-mer length as it depends on many factors, including the assembly quality, complexity of the input genomes, or presence of repeats. High values of k lead to haplotypes containing multiple SNPs instead of distinct single SNPs, if these SNPs are separated by less than k bases. As k increases, the k-mer-defined haplotypes also become more specific to a genome sub-population, leading to a loss of power to detect genotype to phenotype associations. Low values of k, on the other hand, produce highly connected sets of non-specific k-mers. In particular, any repeated region with at least k bases may create a cycle in the DBG (Fig 4). We use $k = 31$ by default, as it produced the best performance to retrieve known markers of *P. aeruginosa* resistance to amikacin and levofloxacin (Fig 5). We found DBGWAS results to be robust to small variations of k between 21 and 41. Similar graph structures were generated whatever the tested value of k for the clonal *M. tuberculosis* species (S7 Fig). More variability was observed for *P. aeruginosa* resistance to amikacin, which involves more complex resistance mechanisms (S8 Fig).

Unitig presence across genomes. Each genome is represented by a vector of presence/absence of each unitig in the cDBG. To do so, we query the unitig associated to each k-mer in a given genome. This procedure is efficient because it relies on constant time operations. Firstly, we use GATB's Minimal Perfect Hash Function (MPHF) [52] to retrieve the index of a given k-mer, and then we use the previously computed association between k-mer and unitig indices to know which unitigs the given genome contains. Since these two operations take

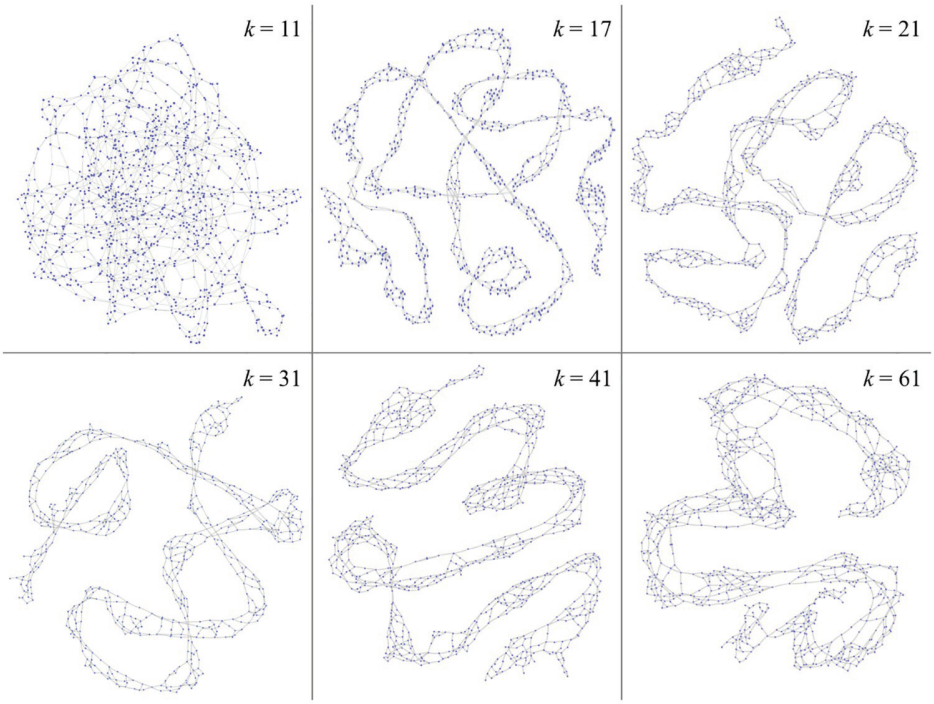


Fig 4. Effect of *k* on the graph topology. A cDBG was built from the *P. aeruginosa gyrA* gene sequences from several strains. When *k* is small, k-mers are highly repeated, which generate numerous loops. As *k* increases, k-mer sequences become more specific and the graph gets more linear. For large values of *k*, few k-mers are shared by all the strains, and the linear path thickens into parallel paths belonging to variable strain populations.

https://doi.org/10.1371/journal.pgen.1007758.g004

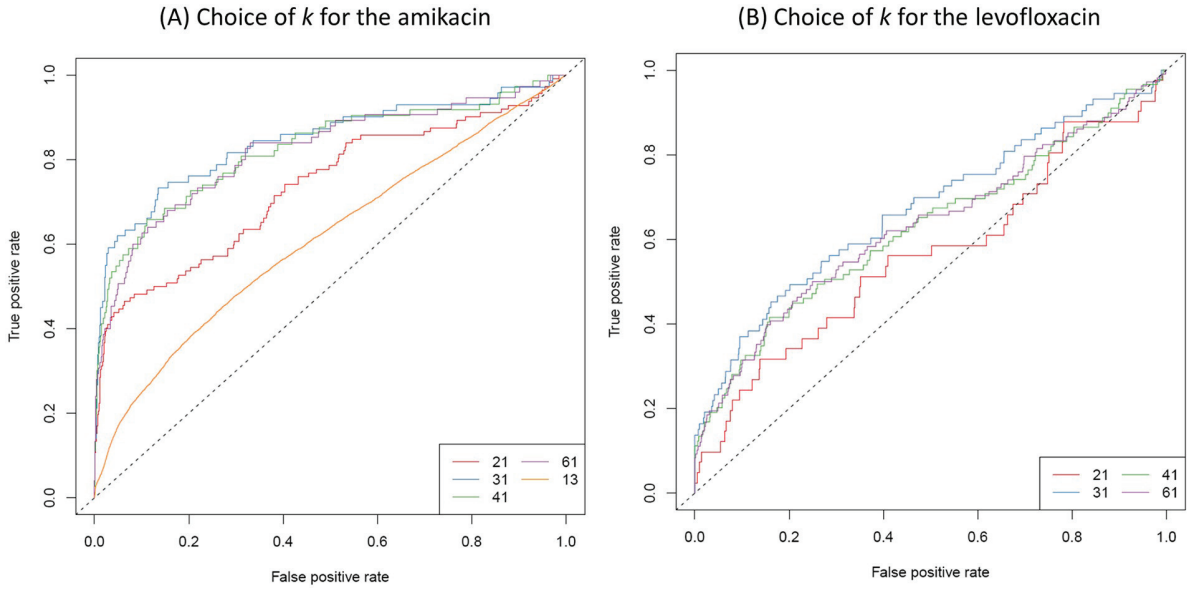


Fig 5. Choice of *k*. True positive *versus* false positive curves for several values of *k* for both amikacin and levofloxacin resistance phenotypes. True positives are unitigs mapping to genuine variants described in resistance databases for the studied drugs [7]. In both cases, the value of *k* leading to the best AUC is $k = 31$.

https://doi.org/10.1371/journal.pgen.1007758.g005

constant time, producing this vector representation for a genome takes linear time on the size of the genome. It is important to note that the GATB's MPHF can be successfully applied here because we always use the same list of k-mers, i.e., after building the DBG, the set of k-mers is fixed and not updated, and because we always query k-mers that are guaranteed to be in the DBG (since we do not filter out any k-mer).

The unitig description on all the input genomes is stored into a matrix U:

$$U_{i,j} = \begin{cases} 1, \text{ if the } j\text{-th unitig is present in the } i\text{-th input genome;} \\ 0, \text{ otherwise.} \end{cases}$$

We then transform the matrix U into Z, which represents the minor allele description, in terms of presence [5]: Z is identical to U except for columns with a mean larger than 0.5, which are complemented: $Z_j = 1 - U_j$ for these columns.

We then restrict Z to its set of unique columns. If several unitigs have the same minor allele presence pattern, then they will be represented by a single column. Keeping duplicates would lead to performing the same statistical test several times. Finally, we filter out columns whose average is below 0.01—the user can specify this threshold using the `-maf` option. We denote the de-duplicated, filtered matrix of patterns by X.

Importantly, both k-mers and unitigs lead to the same set of distinct patterns across the genomes. Indeed, every unitig represents (at least) one k-mer, and conversely every k-mer is represented by one (single) unitig. When de-duplicated, the two representations therefore lead to the same set of patterns to be tested for association with the phenotype.

Testing unitigs for association with the phenotype (step 2)

Human GWAS literature extensively discusses how testing procedures can result in spurious associations if the effect of the population structure is not taken into account [53–55]. Population structures can be strong in bacteria because of their clonality [5, 6, 56, 57]. An additional performance analysis comparing several models for population structure, on both simulated and real data, showed that correcting for population structure using LMMs is often preferable to using a fixed effect correction or not correcting at all (S1 Appendix).

We thus rely on the bugwas method [5], which uses the linear mixed model (LMM) implemented in the GEMMA library [58], to test for association with phenotypes while correcting for the population structure. This method also offers the possibility to test for lineage effects, by calculating p-values for association between the columns of the matrix representing the population structure, and the phenotype [5]. DBGWAS optionally provides bugwas lineage effect plots when the user specifies a phylogenetic tree using the `-newick` option. An example of the generated figures is available at http://pbil.univ-lyon1.fr/datasets/DBGWAS_support/full_dataset_visualization/.

Formally, the LMM represents the distribution of the binarized phenotype Y_i, given the j-th minor allele pattern X_{ij} and the population structure represented by a set of factors $W \in \mathbb{R}^{n \leq p}$, by:

$$Y_i = X_{ij}\beta + W_i^T\alpha + \varepsilon_{ij}, \quad j = 1, \ldots, p. \tag{1}$$

β is the fixed effect of the tested candidate on the phenotype, $\alpha \sim \mathcal{N}(0, \sigma_a^2)$, $\sigma_a^2 > 0$ is the random effect of the population structure, and $\varepsilon_{ij} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ are the residuals with variance $\sigma^2 > 0$. W is estimated from the Z matrix, which includes duplicate columns representing both core and accessory genome. More precisely, denoting $Z = USV^\top$ the singular value decomposition of Z, we use $W = US$.

We test H_0: $\beta = 0$ versus H_1: $\beta \neq 0$ in Eq 1 for each pattern using a likelihood ratio procedure producing p-values and maximum likelihood estimates $\hat{\beta}$. To tackle the situation of multiple testing caused by the high number of tested patterns, we compute q-values, which are the Benjamini-Hochberg transformed p-values controlling for false discovery rate (FDR) [59].

Interpretation of significant unitigs (step 3)

The LMM is used to identify de-duplicated minor allele presence patterns significantly associated with the phenotype at a chosen FDR level. While the testing step is done at the pattern level, the interpretation of the selected features is done at the unitig level. As a result of the de-duplication procedure, a given pattern may correspond to several distinct unitigs. To faithfully interpret the results, all the unitigs corresponding to the significant patterns are retrieved and are assigned the q-value of their pattern. We now show how the initial cDBG can be used in the interpretation step.

Significance threshold. The interpretation step focuses on the unitigs with the lowest q-values. These unitigs are indeed used to build the resulting annotated subgraphs. The unitig selection can be either based on the FDR (q-value threshold) or on a number of presence/absence patterns ordered by increasing q-values. Practically, this is done in DBGWAS using a Significant Features Filter (SFF). For a selection based on a FDR threshold, the SFF value is set between 0 and 1, while any integer value > 1 defines the number of patterns to consider.

In our experiments, we choose not to apply a fixed FDR threshold, even though DBGWAS offers this option. Different datasets lead to different q-values, even by several orders of magnitude, and a single FDR threshold would lead to selecting a large number of unitigs generating more than 1,000 subgraphs on some of them (e.g. *S. aureus* ciprofloxacin) as shown in S8 Table. Instead, we retain the 100 patterns with lowest q-values. Although arbitrary, this choice is tractable for all datasets and provides satisfactory results in our experiments. It does not provide and explicit control of the FDR: only the q-value provides an estimation of the proportion of false discoveries incurred when considering patterns below this value. Checking the q-values of the selected unitigs is therefore essential to assess their significance. If the default SFF = 100 is not satisfactory, it is also possible to re-run the third step only, with a more suitable SFF value.

Graph neighbourhoods. We define the neighbourhood of each significant unitig u (defined by the *SFF*) as the set of unitigs whose shortest path to u has at most $ne = 5$ edges. Users can modify the *ne* value using the `-nh` option. The objects returned by DBGWAS are the connected components of the graph induced by the neighbourhoods of all significant unitigs in the cDBG. As illustrated in Fig 6, nearby significant unitigs might belong to the same connected component, so this process groups unitigs which are likely to be located closely in the genomes. We refer to the connected components as *subgraphs* in the Results section.

The *SFF* value can be tuned to optimise the number and size of the output subgraphs. It has no impact on subgraphs describing SNPs in core sequences (S2 Fig). On the other hand, when significant unitigs map to different regions of a single MGE, such as a plasmid, several subgraphs are generated but can be gathered into a single subgraph by increasing the *SFF* threshold (S4 Fig). When significant unitigs map to several distinct mobile regions, which can be found in different contexts (transposon, integron, etc.) at the population level, the resulting subgraph can become very large and highly branching: decreasing the *SFF* threshold allows to select the few most significant unitigs, generating a subgraph focusing on the most relevant region (S6 Fig). Reducing the graph complexity can also be done by decreasing the *ne* value, using the `-nh` option.

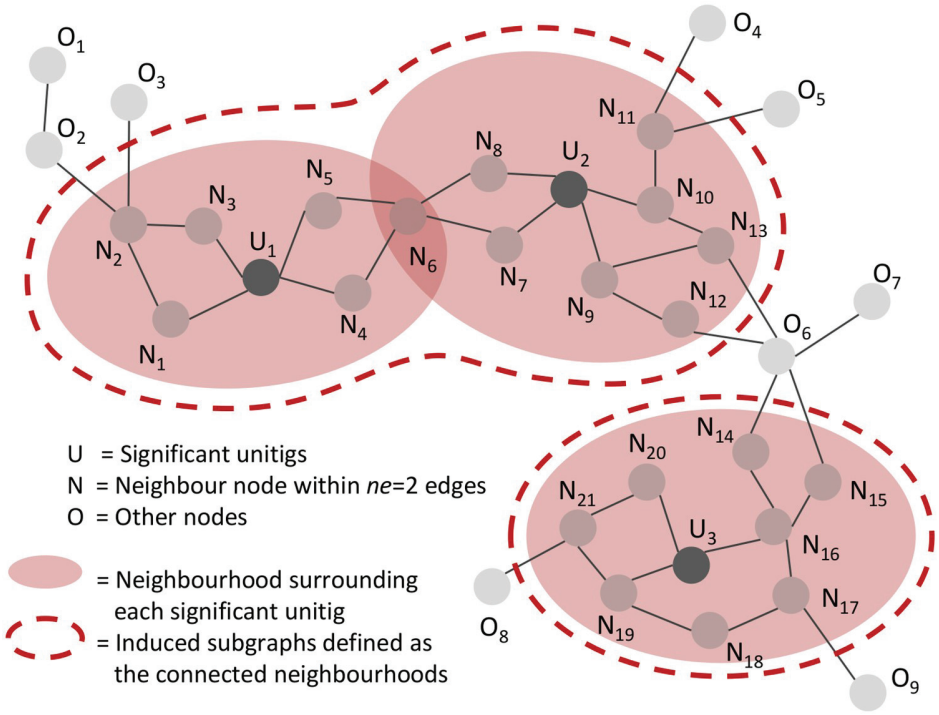


Fig 6. Subgraphs induced by the neighbourhood of significantly associated unitigs. In this example, a neighbourhood of size *ne* = 2 was used: any unitig distant up to 2 edges from a significant unitig is retrieved to define its neighbourhood. Neighbourhoods are merged if they share at least one node, e.g. the neighbourhoods of U_1 and U_2 are merged because they share N_6, and will be represented in a single subgraph.

https://doi.org/10.1371/journal.pgen.1007758.g006

Representing metadata with coloured DBGs. The subgraphs are enriched with metadata to make their interpretation easier. We use the node size to represent allele frequencies, *i.e.*, the proportion of genomes containing the unitig sequence. We describe the effect β of each unitig as estimated by the LMM using colours, in the spirit of the coloured DBGs [19]. Colours are continuously interpolated between red for unitigs with a strong positive effect and blue for those with a strong negative effect.

Annotating the subgraphs. DBGWAS can optionally integrate an automated annotation step using the Blast suite [60] (version 2.6.0+) on local user-defined protein (`-pt-db` option) or nucleic acid (`-nt-db` option) sequence databases. We annotate the subgraphs of interest by blasting each unitig sequence to the available databases. Users can then easily retrieve the annotations which are the most supported by the nodes in the subgraph, or with the lowest E-value. Importantly, DBGWAS works with any nucleotide or protein Fasta files as annotation databases straight away. However, users can customize the annotation databases by changing the Fasta sequences headers to make DBGWAS results more interpretable. A common example is compacting the annotation in the summary page by using abbreviations or gene class names, and expanding them to full names in the subgraph page. Other custom fields can also be included in the annotation table by adding specific tags to the headers. A detailed explanation on how to customize annotation databases for DBGWAS can be found in https://gitlab.com/leoisl/dbgwas/wikis/Customizing-annotation-databases. We also provide on the DBGWAS website a resistance determinant database built by merging the ResFinder, MEGARes, and ARG-ANNOT databases [61–63], and a subset of UniProt restricted to bacterial proteins [24]. Subgraphs discussed in the Results section were annotated using these databases.

Interactive visualisation. DBGWAS produces an interactive view of the enriched and annotated subgraphs, allowing the user to explore the graph topology together with information on each node: allele and phenotype frequencies, q-value, estimated effect, and annotation. The view is built using HTML, CSS, and several Javascript libraries, the main one being Cytoscape.js [64]. Results can be shared and visualised in a web browser. As a large number of components can be produced in one run of DBGWAS, we provide a summary page allowing users to preview and filter the subgraphs. Filtering can be based upon the minimum q-value of all unitigs in the component ($\min_q$), or based on the annotations. A complete description of the DBGWAS interactive interface is available in https://gitlab.com/leoisl/dbgwas/wikis/DBGWAS-web-based-interactive-visualization.

Re-running from *step 2* or *step 3*. It is possible to re-run a part of the analysis if a first run with the default values was unsatisfactory. The `-skip1` option allows to re-run from the second step, for instance to compute the lineage effects (adding the `-newick` option). It is also possible to re-run only the third step by using the `-skip2` option, for instance when the default *SFF* and *nh* values generated highly connected graphs, or if the annotation was incomplete.

Datasets

We used in our experiments genome sequences from three bacterial species with various degrees of genome plasticity, from more clonal to more plastic: *M. tuberculosis*, *S. aureus*, and *P. aeruginosa*. We also built large datasets with random phenotypes for these 3 species, and used them only for time performance and memory usage assessment. All panels are summarised in Table 4.

TB panel. *M. tuberculosis* (TB) is a human pathogen causing 1.7 million deaths each year [66]. This species is known for its apparent absence of horizontal gene transfer (HGT) and, accordingly, most of the reported resistance determinants are chromosomal mutations [67] in core genes or gene promoters. Intergenic regions are also described to be instrumental in multidrug-resistance (MDR) and extensively drug-resistant (XDR) phenotypes [9]. We use the PATRIC AMR phenotype data, as well as genome assemblies from their resource [35, 68]. We thus gather a total of 1302 genomes after filtering based on genome length. Phenotype data include isoniazid, rifampicin, streptomycin, ethambutol, ofloxacin, kanamycin and ethionamide resistance status. Except for the last three drugs, phenotype data are available for more than a thousand genomes. We reconstruct MDR and XDR phenotypes based on the WHO definition [66]. XDR phenotype could only be defined for 689/1302 strains as it required data for at least 4 drugs. Information on how phenotype data and genome assemblies were obtained is available on the PATRIC website.

SA panel. *S. aureus* is a human pathogen causing life-threatening infections. It is subject to HGT and many plasmids, mobile elements, and phage sequences have been described in its genome. However, this does not affect the species' genome size, which is always close to 3 Mbp [69]. Most antibiotic resistance mechanisms are well determined by known variants, as shown in a previous study [30]. This study obtained an overall sensitivity of 97% for predicting 12 phenotypes from rules based on antibiotic marker mapping. We use this study panel of 992 strains obtained by merging their derivation and validation sets.

PA panel. *P. aeruginosa* is a ubiquitous bacterial species responsible for various types of infections. It is highly adaptable thanks to its ability to exchange genetic material within and between species [70]. The species accessory genome is particularly important both in terms of size and diversity, and carries more than half of the genetic determinants already described to confer resistance to antimicrobial drugs [7, 65, 71]. We use a panel of 282 strains, gathered

Table 4. Microbial panels.

Species	Genome plasticity	Range of genome length	Panel name	Source	Phenotype	Number of available genomes
M. tuberculosis	very low	4.4 Mbp	TB	[35]	rifampicin	1,197
					isoniazid	1,287
					ethambutol	1,041
					streptomycin	1,166
					kanamycin	671
					ofloxacin	696
					ethionamide	420
					MDR	1,211
					XDR	689
			Large TB	[11]	random	5,000
S. aureus	low	2.7-3.1 Mbp	SA	[30]	methicillin	501
					ciprofloxacin	991
					erythromycin	991
					penicillin	991
					tetracycline	991
					fusidic acid	991
					trimethoprim	323
					gentamicin	991
					rifampin	991
					mupirocin	490
					vancomycin	501
			Large SA	[11]	random	9,000
P. aeruginosa	high	5.8-7.6 Mbp	PA	[65]	amikacin	280
					levofloxacin	117
					meropenem	280
					piperacillin	280
					colistin	164
					polymyxin B	117
					chloramphenicol	103
					cefepime	280
					fosfomycin	113
			Large PA	[11]	random	2,500

We selected 3 bacterial species with distinct levels of genome plasticity, and with antibiotic resistance phenotypes available for several drugs. For each species, we also created large datasets by computing random phenotypes for all available genome assemblies from NCBI RefSeq.

https://doi.org/10.1371/journal.pgen.1007758.t004

from two collections which mostly include clinical strains: the bioMérieux collection [65] (n = 219) and the Pirnay collection [72] (n = 63). Genome assemblies and categorical phenotypes for 9 antibiotics are available [7]. Binarised phenotypes of amikacin resistance are available on the DBGWAS project page as an example for users.

Phenotype binarisation. Most available phenotypes are categorical, with S, I and R levels, respectively, for susceptible, intermediary, and resistant. We binarise them by assigning a zero value to susceptible strains (S) and one to others (I and R).

Large panels. We built large panels for the three species, in order to analyse the computational performance at a comprehensive scale. To do so, we gathered all genome assemblies of *M. tuberculosis* (5,504), *S. aureus* (9,331), and *P. aeruginosa* (2,802) available on the NCBI RefSeq bacterial genome repository [11], and removed poor quality genomes. For each panel,

we generated random binary phenotypes. For a detailed time and memory assessment, we built several sub-panels from these three large panels at size points of 100, 250, 500, 1,000, 2,500, 5,000 and 9,000 genomes. To build these sub-panels, we sampled genomes uniformly from the panels. To take into account the variability among subsamplings, each sub-panel was randomly built 10 times.

Resistome-based association studies

We benchmarked DBGWAS against a targeted approach to ensure its ability to retrieve all expected resistance determinants. We thus performed association studies under the same model, using as input a collection of known causal resistance SNPs and genes, defining the resistome.

In this validation study, we used bugwas with the same phenotypes and population structure matrix W, so the resistome-based analyses and DBGWAS only differ by their input variant matrix (unitigs versus SNPs or genes presence/absence).

For *P. aeruginosa* resistome, we use a variant matrix previously described [7], which includes presence/absence of known resistance gene variants, as well as the SNPs called against these reference gene variants. For *M. tuberculosis* resistome, we built the variant matrix using the same approach as for *P. aeruginosa* [7]: we called the SNPs from a list of 32 known resistance genes and promoters [34, 67, 73]. The time and memory usage required for the complete analysis (from the mapping of the resistance genes and positions on the genome assemblies to the association study) are provided in Tables 2 and 3.

We sort the annotated features by q-values. S6 and S7 Tables summarise all top variants using their q-value ranks, while Tables 2 and 3 report the annotations of all variants with a q-value < 0.05 for *P. aeruginosa* levofloxacin and *M. tuberculosis* streptomycin resistance, respectively.

k-mer-based GWAS

pyseer. We installed pyseer [6, 36] commit ID `d17602500a4530b0e68a679ed675fdb12942f56f` (9 commits ahead of pyseer v1.1.1). pyseer pipeline is composed of four steps: 1) k-mer counting; 2) population structure estimation; 3) running pyseer; 4) downstream analysis. To use the correct parameters, we followed the pyseer tutorial (https://pyseer.readthedocs.io/en/master/tutorial.html). For k-mer counting, we used fsm-lite (https://github.com/nvalimak/fsm-lite), filtering out all k-mers with a minor allele frequency smaller than 1%. For population structure estimation, we used Mash v2.0 [74]. To run pyseer, we used 8 cores and a LRT p-value threshold of 0.05. Downstream analysis involved getting the k-mers which exceeded the significance threshold (which can be found using the `scripts/count_patterns.py` script), sorting them by LRT p-value, blasting them against the two databases presented in the Interpretation of significant unitigs (step 3) subsection, and keeping the best hit for each k-mer. For reproducibility purposes, the scripts we used to run pyseer can be found at https://gitlab.com/leoisl/DBGWAS_support/tree/master/scripts/pySEER.

HAWK. We firstly ran HAWK [13] v0.9.8-beta, as it allows correcting for population structure. Unfortunately, it was unable to find the known causal variants reported for *P. aeruginosa* levofloxacin and *M. tuberculosis* streptomycin resistances by other methods (see Tables 2 and 3). We therefore kept in our benchmarks an earlier version, HAWK v0.8.3-beta, which presented better qualitative performance for these two evaluated panels. HAWK pipeline is composed of five steps: 1) k-mer counting with a modified version of jellyfish [75]; 2) running HAWK; 3) assembling significant k-mers with ABYSS [76]; 4) getting statistics on the assembled sequences; 5) downstream analysis. The first four steps were performed as described in

HAWK's github page. However, in the first step, we had to remove the lower-count cutoff in `jellyfish dump` (parameter `-L`), since we are working with contigs and not reads. The last step was performed similarly as the one described for pyseer. For reproducibility purposes, the scripts we used to run HAWK v0.8.3-beta can be found at https://gitlab.com/leoisl/DBGWAS_support/tree/master/scripts/HAWK_0_8_3_beta.

Supporting information

S1 Fig. Alignment to a reference (when possible), cDBG, and k-mers obtained for similar (A) and very polymorphic genomes (B). In the first case, the 3 loci represented as polymorphic in the alignment lead to 3 bubble patterns in the cDBG, and numerous redundant k-mers. In the second case, genomes are so polymorphic that an alignment is not possible. The cDBG summarizes well the common regions and the links between them, while the collection of unique k-mers still contains redundancy.
(PDF)

S2 Fig. Effect of *SFF* on the top subgraphs generated for *S. aureus* ciprofloxacin resistance. Annotation of the first subgraphs is strictly conserved (red for *parC*, green for *gyrA*, yellow for *norA* promoter region, blue for noncoding between *glmM* and *fmtB* and violet for transposase flanking regions).
(PDF)

S3 Fig. Effect of *SFF* on the top subgraphs generated for *S. aureus* methicillin resistance. Only one subgraph, containing the *mecA* gene (highlighted in red) is generated for lower *SFF* values. Then several regions of the SCC*mec* cassette appear for $SFF = 70$, and are aggregated into a single subgraph for $SFF \geq 150$. Green subgraphs do not concern the *mecA* MGE.
(PDF)

S4 Fig. Effect of *SFF* on the top subgraphs generated for *S. aureus* penicillin resistance. Green subgraphs do not concern the *blaZ* MGE. Annotations are ordered by number of nodes carrying it. Yellow, orange and pink highlight *blaZ*, *blaR1* and *blaI*, respectively.
(PDF)

S5 Fig. Effect of *SFF* on the top subgraphs generated for *S. aureus* erythromycin resistance. Only one subgraph, describing the *ermC* and its plasmid is outputted when $SFF < 200$. Green subgraphs do not concern the *ermC* MGE.
(PDF)

S6 Fig. Effect of *SFF* on the top subgraphs generated for *P. aeruginosa* amikacin resistance. Nodes corresponding to *aac(6')* gene are shown in a blue frame. When the *SFF* parameter increases, these nodes aggregate to others genes found at least once close to *aac(6')*. The annotation of the following subgraphs are well conserved (same color legend as in S8 Fig).
(PDF)

S7 Fig. Effect of *k* on the four first subgraphs obtained for TB rifampicin resistance. With a *k* value varying between 21 and 41, the first 3 subgraphs always have the same ordering, shape and annotation, as well as comparable q-values, although smaller q-values are observed for lower values of *k*. The number of significant unitigs per subgraph is also well conserved. The fourth top-rated subgraphs are not always the same: the *gyrA* mutation appears at a lower rank when *k* is smaller.
(PDF)

S8 Fig. Effect of *k* on the five first subgraphs obtained for *P. aeruginosa* amikacin resistance. When *k* varies, the plasmid (yellow) and the mercury reductase and transposase (blue) remain among the five top-rated subgraphs. However, *k* has an effect on the aggregation of subgraphs corresponding to different genetic events: the mutation on *aac(6')* gene (blue frame) always appears in the first subgraph but is merged with the large mercury reductase and transposase subgraph for $k = 27$, 39 and 41. The order of the subgraphs also varies with *k*: up to four ranks for some subgraphs, and others leave the top-5 list.
(PDF)

S9 Fig. Large scale analysis on computational resources usage. This figure describes how DBGWAS scales in terms of time and memory usage for large datasets, containing up to 9,000 genomes. The large panels used here are described in the Large panels subsection of the Methods section. To understand better DBGWAS performance behaviour, we present performance curves for each panel at size points of 100, 250, 500, 1,000, 2,500, 5,000 and 9,000 genomes. The executions were done in a cluster, instead of a single machine, and used 8 cores each. In order to reduce subsampling and machine heterogeneity problems, each sub-panel was randomly built 10 times and we present the time and memory usage for all these executions. Although these two measures not only depends on the number of input genomes but also on their length and complexity, this figure allows estimations of the computational resources usage on small and large panels with different genome plasticities.
(PDF)

S1 Table. DBGWAS time and maximal memory load on a single core. All runs presented in this table were executed with the default parameters, without optional steps (lineage effect analysis nor annotation of subgraphs), on a single *Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz* core. The datasets are described in the Datasets subsection of the Methods section. DBGWAS ran in less than 2,5 hours for all experiments in our benchmark. The maximum memory load (given between parenthesis in the Runtime column) was 11 GB of RAM. The panel size and genome length (given between parenthesis in the Panel column) did not drive alone the running performances; the genome complexity played an important role as well. To view the gain in performance of DBGWAS when running on multiple (8) cores, see S2 Table.
(PDF)

S2 Table. Benchmarking DBGWAS, pyseer and HAWK: Comparison of time and maximal memory load. The total execution time is presented with the maximal memory consumption in parenthesis, in order of GBs. For pyseer and HAWK, the time and memory for each step is also detailed. All tools were ran on a same machine with 8 *Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz* cores, 315 GB of RAM and 1 TB of disk space. Each execution used all the 8 available cores. The datasets are described in the Datasets subsection of the Methods section. However, for the three large panels (Large TB, Large SA, and Large PA), here we just chose a random 2,500-genome sub-panel. Moreover, DBGWAS was ran with the default parameters, without optional steps (lineage effect analysis nor annotation of subgraphs). The parameters for pyseer and HAWK were the ones described in the k-mer-based GWAS subsection of the Methods section. We did not consider the time and memory consumed in the last step for these two tools (downstream analysis). The runs taking more than 5 days to finish were interrupted and are shown as *Timeout*. The runs that exceeded 1 TB of disk space were interrupted and are shown as *DQE* (Disk Quota Exceeded).
(PDF)

S3 Table. DBGWAS results for *M. tuberculosis* resistance to antibiotics. For each antibiotic, top subgraphs were reported with their rank, the q-value of the unitig with the lowest q-value

($\min_q$), the corresponding estimated effect (estimated β of the linear model) and the number of susceptible (resp. resistant) strains harbouring this unitig (count per phenotype). The type of event represented by the subgraph, its annotation and some comments and references on this annotation were also provided. Comments were coloured if the annotation was previously described in antibiotic resistance literature: in green if this description concerned the tested antibiotic, in orange otherwise.
(XLS)

S4 Table. DBGWAS results for *S. aureus* resistance to antibiotics. For each antibiotic, top subgraphs were reported with their rank, the q-value of the unitig with the lowest q-value ($\min_q$), the corresponding estimated effect (estimated β of the linear model) and the number of susceptible (resp. resistant) strains harbouring this unitig (count per phenotype). The type of event represented by the subgraph, its annotation and some comments and references on this annotation were also provided. Comments were coloured if the annotation was previously described in antibiotic resistance literature: in green if this description concerned the tested antibiotic, in orange otherwise.
(XLS)

S5 Table. DBGWAS results for *P. aeruginosa* resistance to antibiotics. For each antibiotic, top subgraphs were reported with their rank, the q-value of the unitig with the lowest q-value ($\min_q$), the corresponding estimated effect (estimated β of the linear model) and the number of susceptible (resp. resistant) strains harbouring this unitig (count per phenotype). The type of event represented by the subgraph, its annotation and some comments and references on this annotation were also provided. Comments were coloured if the annotation was previously described in antibiotic resistance literature: in green if this description concerned the tested antibiotic, in orange otherwise.
(XLS)

S6 Table. Resistome-based association study results for *M. tuberculosis* resistance to antibiotics. For each antibiotic, the 10 first features most associated to the phenotype were reported, with their rank, q-value, and estimated effect (estimated β of the linear model). The type of targeted variant, with its gene annotation were also provided. Comments were coloured if the annotation was previously described in antibiotic resistance literature: in green if this description concerned the tested antibiotic, in orange otherwise. The last column presents the corresponding subgraphs found by DBGWAS, with their rank and $\min_q$.
(XLS)

S7 Table. Resistome-based association study results for *P. aeruginosa* resistance to antibiotics. For each antibiotic, the 10 first features most associated to the phenotype were reported, with their rank, q-value, and estimated effect (estimated β of the linear model). The type of targeted variant, with its gene annotation were also provided. Comments were coloured if the annotation was previously described in antibiotic resistance literature: in green if this description concerned the tested antibiotic, in orange otherwise. The last column presents the corresponding subgraphs found by DBGWAS, with their $\min_q$.
(XLS)

S8 Table. Number of subgraphs generated using different significance thresholds. This table shows the number of subgraphs generated when defining the significant unitigs as the ones with the 100 lowest q-values (default *SFF* = 100, 'top 100') or when using a 5% false discovery rate (FDR) threshold (*SFF* = 0.05, '5% FDR'). Different datasets lead to different q-values, even by several orders of magnitude. For instance, a single FDR threshold leads to

selecting a large number of unitigs generating several hundreds subgraphs for SA (*S. aureus*) panel.
(PDF)

S1 Appendix. Evaluation of association models.
(PDF)

Acknowledgments

The authors thank Jean-Baptiste Veyrieras, Sarah Earle, Chieh-Hsi Wu and Daniel Wilson, as well as Jean-Pierre Flandrois, Manolo Gouy, Stéphane Schicklin and Ghislaine Guigon for their insightful comments. The authors also thank the reviewers for their accurate comments and suggestions, which helped to improve the quality of the manuscript.

Author Contributions

Conceptualization: Magali Jaillard, Maud Tournoud, Vincent Lacroix, Laurent Jacob.

Data curation: Pierre Mahé.

Formal analysis: Magali Jaillard, Leandro Lima, Pierre Mahé, Laurent Jacob.

Investigation: Magali Jaillard, Leandro Lima, Laurent Jacob.

Methodology: Magali Jaillard, Leandro Lima, Laurent Jacob.

Project administration: Pierre Mahé, Laurent Jacob.

Software: Magali Jaillard, Leandro Lima, Laurent Jacob.

Supervision: Maud Tournoud, Laurent Jacob.

Validation: Magali Jaillard, Leandro Lima, Alex van Belkum.

Visualization: Magali Jaillard, Leandro Lima.

Writing – original draft: Magali Jaillard, Leandro Lima, Alex van Belkum, Laurent Jacob.

Writing – review & editing: Magali Jaillard, Leandro Lima, Maud Tournoud, Pierre Mahé, Alex van Belkum, Vincent Lacroix, Laurent Jacob.

References

1. Farhat MR, Shapiro BJ, Kieser KJ, Sultana R, Jacobson KR, Victor TC, et al. Genomic analysis identifies targets of convergent positive selection in drug-resistant *Mycobacterium tuberculosis*. Nature genetics. 2013; 45(10):1183–1189. https://doi.org/10.1038/ng.2747 PMID: 23995135
2. Sheppard SK, Didelot X, Meric G, Torralbo A, Jolley KA, Kelly DJ, et al. Genome-wide association study identifies vitamin B5 biosynthesis as a host specificity factor in Campylobacter. Proceedings of the national academy of sciences. 2013; 110(29):11923–11927. https://doi.org/10.1073/pnas.1305559110
3. Alam MT, Petit RA, Crispell EK, Thornton TA, Conneely KN, Jiang Y, et al. Dissecting vancomycin-intermediate resistance in *Staphylococcus aureus* using genome-wide association. Genome biology and evolution. 2014; 6(5):1174–1185. https://doi.org/10.1093/gbe/evu092 PMID: 24787619
4. Chewapreecha C, Marttinen P, Croucher NJ, Salter SJ, Harris SR, Mather AE, et al. Comprehensive identification of single nucleotide polymorphisms associated with beta-lactam resistance within pneumococcal mosaic genes. PLoS genetics. 2014; 10(8):e1004547. https://doi.org/10.1371/journal.pgen.1004547 PMID: 25101644
5. Earle SG, Wu CH, Charlesworth J, Stoesser N, Gordon NC, Walker TM, et al. Identifying lineage effects when controlling for population structure improves power in bacterial association studies. Nature microbiology. 2016; p. 16041. https://doi.org/10.1038/nmicrobiol.2016.41 PMID: 27572646

6. Lees JA, Vehkala M, Välimäki N, Harris SR, Chewapreecha C, Croucher NJ, et al. Sequence element enrichment analysis to determine the genetic basis of bacterial phenotypes. Nature communications. 2016; 7:12797. https://doi.org/10.1038/ncomms12797 PMID: 27633831
7. Jaillard M, van Belkum A, Cady KC, Creely D, Shortridge D, Blanc B, et al. Correlation between phenotypic antibiotic susceptibility and the resistome in *Pseudomonas aeruginosa*. International journal of antimicrobial agents. 2017;. https://doi.org/10.1016/j.ijantimicag.2017.02.026 PMID: 28554735
8. Page AJ, Cummins CA, Hunt M, Wong VK, Reuter S, Holden MT, et al. Roary: rapid large-scale prokaryote pan genome analysis. Bioinformatics. 2015; 31(22):3691–3693. https://doi.org/10.1093/bioinformatics/btv421 PMID: 26198102
9. Zhang H, Li D, Zhao L, Fleming J, Lin N, Wang T, et al. Genome sequencing of 161 *Mycobacterium tuberculosis* isolates from China identifies genes and intergenic regions associated with drug resistance. Nature genetics. 2013; 45(10):1255–1260. https://doi.org/10.1038/ng.2735 PMID: 23995137
10. Blair JM, Webber MA, Baylay AJ, Ogbolu DO, Piddock LJ. Molecular mechanisms of antibiotic resistance. Nature reviews microbiology. 2015; 13(1):42–51. https://doi.org/10.1038/nrmicro3380 PMID: 25435309
11. Haft DH, DiCuccio M, Badretdin A, Brover V, Chetvernin V, O'Neill K, et al. RefSeq: an update on prokaryotic genome annotation and curation. Nucleic acids research. 2017; 46(D1):D851–D860. https://doi.org/10.1093/nar/gkx1068
12. Le Bras Y, Collin O, Monjeaud C, Lacroix V, Rivals É, Lemaitre C, et al. Colib'read on galaxy: a tools suite dedicated to biological information extraction from raw NGS reads. GigaScience. 2016; 5(1):1. https://doi.org/10.1186/s13742-015-0105-2
13. Rahman A, Hallgrímsdóttir I, Eisen M, Pachter L. Association mapping from sequencing reads using k-mers. eLife. 2018; 7:e32920. https://doi.org/10.7554/eLife.32920 PMID: 29897334
14. Read TD, Massey RC. Characterizing the genetic basis of bacterial phenotypes using genome-wide association studies: a new direction for bacteriology. Genome medicine. 2014; 6(11):109. https://doi.org/10.1186/s13073-014-0109-z PMID: 25593593
15. Power RA, Parkhill J, de Oliveira T. Microbial genome-wide association studies: lessons from human GWAS. Nature reviews genetics. 2017; 18(1):41–50. https://doi.org/10.1038/nrg.2016.132 PMID: 27840430
16. de Bruijn N. A combinatorial problem. Proceedings of the koninklijke nederlandse akademie van wetenschappen Series A. 1946; 49(7):758.
17. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proceedings of the national academy of sciences. 2001; 98(17):9748–9753. https://doi.org/10.1073/pnas.171285098
18. Zhang W, Chen J, Yang Y, Tang Y, Shang J, Shen B. A practical comparison of *de novo* genome assembly software tools for next-generation sequencing technologies. PloS one. 2011; 6(3):e17915. https://doi.org/10.1371/journal.pone.0017915 PMID: 21423806
19. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. *De novo* assembly and genotyping of variants using colored de Bruijn graphs. Nature Genetics. 2012; 44(2):226–232. https://doi.org/10.1038/ng.1028 PMID: 22231483
20. Hooper DC, Jacoby GA. Mechanisms of drug resistance: quinolone resistance. Annals of the New York academy of sciences. 2015; 1354(1):12–31. https://doi.org/10.1111/nyas.12830 PMID: 26190223
21. Lowy FD. Antimicrobial resistance: the example of *Staphylococcus aureus*. Journal of clinical investigation. 2003; 111(9):1265. https://doi.org/10.1172/JCI18535 PMID: 12727914
22. Piton J, Petrella S, Delarue M, André-Leroux G, Jarlier V, Aubry A, et al. Structural insights into the quinolone resistance mechanism of *Mycobacterium tuberculosis* DNA gyrase. PLoS one. 2010; 5(8):e12245. https://doi.org/10.1371/journal.pone.0012245 PMID: 20805881
23. Lambert P. Mechanisms of antibiotic resistance in *Pseudomonas aeruginosa*. Journal of the royal society of medicine. 2002; 95(Suppl 41):22. PMID: 12216271
24. UniProt consortium. UniProt: the universal protein knowledgebase. Nucleic acids research. 2017; 45(D1):D158–D169. https://doi.org/10.1093/nar/gkw1099 PMID: 27899622
25. Lambert T, Ploy M, Courvalin P. A spontaneous point mutation in the *aac(6')-Ib'* gene results in altered substrate specificity of aminoglycoside 6'-N-acetyltransferase of a *Pseudomonas fluorescens* strain. FEMS microbiology letters. 1994; 115:297–304. PMID: 8138142
26. Lee H, Cho S, Bang H, Lee J, Bai G, Kim S, et al. Exclusive mutations related to isoniazid and ethionamide resistance among *Mycobacterium tuberculosis* isolates from Korea. The international journal of tuberculosis and lung disease. 2000; 4(5):441–447. PMID: 10815738

27. Farhat MR, Sultana R, Iartchouk O, Bozeman S, Galagan J, Sisk P, et al. Genetic determinants of drug resistance in *Mycobacterium tuberculosis* and their diagnostic value. American journal of respiratory and critical care medicine. 2016; 194(5):621–630. PMID: 26910495

28. Flandrois JP, Lina G, Dumitrescu O. MUBII-TB-DB: a database of mutations associated with antibiotic resistance in *Mycobacterium tuberculosis*. BMC bioinformatics. 2014; 15(1):107. https://doi.org/10.1186/1471-2105-15-107 PMID: 24731071

29. IWG-SCC consortium. Classification of staphylococcal cassette chromosome *mec* (SCC*mec*): guidelines for reporting novel SCC*mec* elements. Antimicrobial agents and chemotherapy. 2009; 53(12):4961–4967. https://doi.org/10.1128/AAC.00579-09 PMID: 19721075

30. Gordon N, Price J, Cole K, Everitt R, Morgan M, Finney J, et al. Prediction of *Staphylococcus aureus* antimicrobial resistance by whole-genome sequencing. Journal of clinical microbiology. 2014; 52(4):1182–1191. https://doi.org/10.1128/JCM.03117-13 PMID: 24501024

31. Westh H, Hougaard D, Vuust J, Rosdahl V. Prevalence of erm gene classes in erythromycin-resistant *Staphylococcus aureus* strains isolated between 1959 and 1988. Antimicrobial agents and chemotherapy. 1995; 39(2):369–373. https://doi.org/10.1128/AAC.39.2.369 PMID: 7726500

32. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, et al. GenBank. Nucleic acids research. 2012; 41(D1):D36–D42. https://doi.org/10.1093/nar/gks1195 PMID: 23193287

33. Bi D, Xie Y, Tai C, Jiang X, Zhang J, Harrison EM, et al. A site-specific integrative plasmid found in *Pseudomonas aeruginosa* clinical isolate HS87 along with a plasmid carrying an aminoglycoside-resistant gene. PloS one. 2016; 11(2):e0148367. https://doi.org/10.1371/journal.pone.0148367 PMID: 26841043

34. Palomino JC, Martin A. Drug resistance mechanisms in *Mycobacterium tuberculosis*. Antibiotics. 2014; 3(3):317–340. https://doi.org/10.3390/antibiotics3030317 PMID: 27025748

35. Davis JJ, Boisvert S, Brettin T, Kenyon RW, Mao C, Olson R, et al. Antimicrobial resistance prediction in PATRIC and RAST. Scientific reports. 2016; 6:27930. https://doi.org/10.1038/srep27930 PMID: 27297683

36. Lees J, Galardini M, Bentley SD, Weiser JN, Corander J. pyseer: a comprehensive tool for microbial pangenome-wide association studies. Bioinformatics. 2018; p. bty539.

37. Traore H, Fissette K, Bastian I, Devleeschouwer M, Portaels F. Detection of rifampicin resistance in *Mycobacterium tuberculosis* isolates from diverse countries by a commercial line probe assay as an initial indicator of multidrug resistance. The international journal of tuberculosis and lung disease. 2000; 4(5):481–484. PMID: 10815743

38. Illakkiam D, Shankar M, Ponraj P, Rajendhran J, Gunasekaran P. Genome sequencing of a mung bean plant growth promoting strain of *P. aeruginosa* with biocontrol ability. International journal of genomics. 2014; 2014. https://doi.org/10.1155/2014/123058 PMID: 25184130

39. Ali-Ahmad A, Fadel F, Sebban-Kreuzer C, Ba M, Pélissier GD, Bornet O, et al. Structural and functional insights into the periplasmic detector domain of the GacS histidine kinase controlling biofilm formation in *Pseudomonas aeruginosa*. Scientific reports. 2017; 7(1):11262. https://doi.org/10.1038/s41598-017-11361-3 PMID: 28900144

40. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, et al. Computational pan-genomics: status, promises and challenges. Briefings in bioinformatics. 2016; p. bbw089.

41. Paten B, Novak AM, Eizenga JM, Garrison E. Genome graphs and the evolution of genome inference. Genome research. 2017; 27(5):665–676. https://doi.org/10.1101/gr.214155.116 PMID: 28360232

42. Baaijens JA, El Aabidine AZ, Rivals E, Schönhuth A. *De novo* assembly of viral quasispecies using overlap graphs. Genome research. 2017; 27(5):835–848. https://doi.org/10.1101/gr.215038.116 PMID: 28396522

43. Jaillard M. Fine mapping of antibiotic resistance determinants. PhD thesis. 2018;in preparation.

44. Dunne WM Jr, Jaillard M, Rochas O, Van Belkum A. Microbial genomics and antimicrobial susceptibility testing. Expert review of molecular diagnostics. 2017; 17(3):257–269. https://doi.org/10.1080/14737159.2017.1283220

45. Kos VN, Déraspe M, McLaughlin RE, Whiteaker JD, Roy PH, Alm RA, et al. The resistome of *Pseudomonas aeruginosa* in relationship to phenotypic susceptibility. Antimicrobial agents and chemotherapy. 2014; p. AAC–03954. https://doi.org/10.1128/AAC.03954-14 PMID: 25367914

46. Bradley P, Gordon NC, Walker TM, Dunn L, Heys S, Huang B, et al. Rapid antibiotic-resistance predictions from genome sequence data for *Staphylococcus aureus* and *Mycobacterium tuberculosis*. Nature communications. 2015; 6:10063. https://doi.org/10.1038/ncomms10063 PMID: 26686880

47. Moradigaravand D, Palm M, Farewell A, Mustonen V, Warringer J, Parts L. Precise prediction of antibiotic resistance in *Escherichia coli* from full genome sequences. bioRxiv. 2018; p. 338194.

48. Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, et al. ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. Genome research. 2008; 18(5):810–820. https://doi.org/10.1101/gr.7337908 PMID: 18340039

49. Zerbino D, Birney E. Velvet: algorithms for *de novo* Short Read Assembly Using De Bruijn Graphs. Genome research. 2008;. https://doi.org/10.1101/gr.074492.107 PMID: 18349386

50. Chikhi R, Limasset A, Medvedev P. Compacting de Bruijn graphs from sequencing data quickly and in low memory. Bioinformatics. 2016; 32(12):i201–i208. https://doi.org/10.1093/bioinformatics/btw279 PMID: 27307618

51. Drezen E, Rizk G, Chikhi R, Deltel C, Lemaitre C, Peterlongo P, et al. GATB: genome assembly & analysis tool box. Bioinformatics. 2014; 30(20):2959–2961. https://doi.org/10.1093/bioinformatics/btu406 PMID: 24990603

52. Limasset A, Rizk G, Chikhi R, Peterlongo P. Fast and scalable minimal perfect hashing for massive key sets. arXiv 2017;.

53. Balding DJ. A tutorial on statistical methods for population association studies. Nature reviews genetics. 2006; 7(10):781–791. https://doi.org/10.1038/nrg1916 PMID: 16983374

54. Zhou X, Stephens M. Efficient multivariate linear mixed-model algorithms for genome-wide association studies. Nature methods. 2014; 11(4):407. https://doi.org/10.1038/nmeth.2848 PMID: 24531419

55. Widmer C, Lippert C, Weissbrod O, Fusi N, Kadie C, Davidson R, et al. Further improvements to linear mixed models for genome-wide association studies. Scientific reports. 2014; 4. https://doi.org/10.1038/srep06874 PMID: 25387525

56. Falush D, Bowden R. Genome-wide association mapping in bacteria? Trends in microbiology. 2006; 14(8):353–355. https://doi.org/10.1016/j.tim.2006.06.003 PMID: 16782339

57. Collins C, Didelot X. A phylogenetic method to perform genome-wide association studies in microbes that accounts for population structure and recombination. PLOS Computational Biology. 2018; 14(2):1–21. https://doi.org/10.1371/journal.pcbi.1005958

58. Zhou X, Stephens M. Genome-wide efficient mixed-model analysis for association studies. Nature genetics. 2012; 44(7):821–824. https://doi.org/10.1038/ng.2310 PMID: 22706312

59. Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the royal statistical society Series B (Methodological). 1995; p. 289–300.

60. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC bioinformatics. 2009; 10(1):421. https://doi.org/10.1186/1471-2105-10-421 PMID: 20003500

61. Zankari E, Hasman H, Cosentino S, Vestergaard M, Rasmussen S, Lund O, et al. Identification of acquired antimicrobial resistance genes. Journal of antimicrobial chemotherapy. 2012; 67(11):2640–2644. https://doi.org/10.1093/jac/dks261 PMID: 22782487

62. Lakin SM, Dean C, Noyes NR, Dettenwanger A, Ross AS, Doster E, et al. MEGARes: an antimicrobial resistance database for high throughput sequencing. Nucleic acids research. 2017; 45(D1):D574–D580. https://doi.org/10.1093/nar/gkw1009 PMID: 27899569

63. Gupta SK, Padmanabhan BR, Diene SM, Lopez-Rojas R, Kempf M, Landraud L, et al. ARG-ANNOT, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes. Antimicrobial agents and chemotherapy. 2014; 58(1):212–220. https://doi.org/10.1128/AAC.01310-13 PMID: 24145532

64. Franz M, Lopes CT, Huck G, Dong Y, Sumer O, Bader GD. Cytoscape.js: a graph theory library for visualisation and analysis. Bioinformatics. 2015; 32(2):309–311. https://doi.org/10.1093/bioinformatics/btv557 PMID: 26415722

65. van Belkum A, Soriaga LB, LaFave MC, Akella S, Veyrieras JB, Barbu EM, et al. Phylogenetic distribution of CRISPR-Cas systems in antibiotic-resistant *Pseudomonas aeruginosa*. mBio. 2015; 6(6):e01796–15. https://doi.org/10.1128/mBio.01796-15 PMID: 26604259

66. Organization WH. Global tuberculosis report. Geneva: WHO Press Release. 2017;Licence: CC BY-NCSA 3.0 IGO.

67. Gygli SM, Borrell S, Trauner A, Gagneux S. Antimicrobial resistance in *Mycobacterium tuberculosis*: mechanistic and evolutionary perspectives. FEMS microbiology reviews. 2017; 41(3):354–373. https://doi.org/10.1093/femsre/fux011 PMID: 28369307

68. Wattam AR, Davis JJ, Assaf R, Boisvert S, Brettin T, Bun C, et al. Improvements to PATRIC, the all-bacterial bioinformatics database and analysis resource center. Nucleic acids research. 2016; 45(D1):D535–D542. https://doi.org/10.1093/nar/gkw1017 PMID: 27899627

69. Mlynarczyk A, Mlynarczyk G, Jeljaszewicz J. The genome of *Staphylococcus aureus*: a review. Zentralblatt für Bakteriologie. 1998; 287(4):277–314. https://doi.org/10.1016/S0934-8840(98)80165-5 PMID: 9638861

70. Liu YY, Wang Y, Walsh TR, Yi LX, Zhang R, Spencer J, et al. Emergence of plasmid-mediated colistin resistance mechanism MCR-1 in animals and human beings in China: a microbiological and molecular biological study. The Lancet infectious diseases. 2016; 16(2):161–168. https://doi.org/10.1016/S1473-3099(15)00424-7 PMID: 26603172

71. Kung VL, Ozer EA, Hauser AR. The accessory genome of *Pseudomonas aeruginosa*. Microbiology and molecular biology reviews. 2010; 74(4):621–641. https://doi.org/10.1128/MMBR.00027-10 PMID: 21119020

72. Pirnay JP, Bilocq F, Pot B, Cornelis P, Zizi M, Van Eldere J, et al. *Pseudomonas aeruginosa* population structure revisited. PLoS one. 2009; 4(11):e7740. https://doi.org/10.1371/journal.pone.0007740 PMID: 19936230

73. Coll F, McNerney R, Preston MD, Guerra-Assunção JA, Warry A, Hill-Cawthorne G, et al. Rapid determination of anti-tuberculosis drug resistance from whole-genome sequences. Genome medicine. 2015; 7(1):51. https://doi.org/10.1186/s13073-015-0164-0 PMID: 26019726

74. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, et al. Mash: fast genome and metagenome distance estimation using MinHash. Genome biology. 2016; 17(1):132. https://doi.org/10.1186/s13059-016-0997-x PMID: 27323842

75. Marçais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 2011; 27(6):764–770. https://doi.org/10.1093/bioinformatics/btr011 PMID: 21217122

76. Jackman SD, Vandervalk BP, Mohamadi H, Chu J, Yeo S, Hammond SA, et al. ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. Genome research. 2017; 27(5):768–777. https://doi.org/10.1101/gr.214346.116 PMID: 28232478

Chapter 7

Comparative assessment of long-read error-correction software applied to RNA-sequencing data

Preamble

Key points

- Long-read transcriptome sequencing is hindered by high error rates that affect analyses such as the identification of isoforms, exon boundaries, open reading frames, and the creation of gene catalogues;
- This review evaluates the extent to which existing long-read DNA error correction methods are capable of correcting cDNA Nanopore reads;
- Existing tools significantly lower the error rate, but they also significantly perturb gene family sizes and isoform diversity.

Status

Submitted to the journal *Briefings in Bioinformatics* [76], currently under major review. It will be resubmitted before the defence.

Author contributions

L. is first author in this paper.

PREPRINT

Comparative assessment of long-read error-correction software applied to RNA-sequencing data

Leandro Lima [1,2,3,*], Camille Marchet [4], Ségolène Caboche [5], Corinne Da Silva [6], Benjamin Istace [6], Jean-Marc Aury [6], Hélène Touzet [4] and Rayan Chikhi [4]

[1]Univ Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Evolutive UMR5558 F-69622 Villeurbanne, France
[2]EPI ERABLE - Inria Grenoble, Rhône-Alpes, France
[3]Università di Roma "Tor Vergata", Roma, Italy
[4]CNRS, Université de Lille, CRIStAL UMR 9189, Lille, France
[5]Université de Lille, CNRS, Inserm, CHU Lille, Institut Pasteur de Lille, U1019, UMR8204, Center for Infection and Immunity of Lille, Lille, France
[6]Genoscope, Institut de biologie Francois-Jacob, Commissariat à l'Energie Atomique (CEA), Université Paris-Saclay, Evry, France

Abstract

Motivation: Long-read sequencing technologies offer promising alternatives to high-throughput short read sequencing, especially in the context of RNA-sequencing. However these technologies are currently hindered by high error rates that affect analyses such as the identification of isoforms, exon boundaries, open reading frames, and the creation of gene catalogues. Due to the novelty of such data, computational methods are still actively being developed and options for the error-correction of RNA-sequencing long reads remain limited.

Results: In this article, we evaluate the extent to which existing long-read DNA error correction methods are capable of correcting cDNA Nanopore reads. We provide an automatic and extensive benchmark tool that not only reports classical error-correction metrics but also the effect of correction on gene families, isoform diversity, bias toward the major isoform, and splice site detection. We find that long read error-correction tools that were originally developed for DNA are also suitable for the correction of RNA-sequencing data, especially in terms of increasing base-pair accuracy. Yet investigators should be warned that the correction process perturbs gene family sizes and isoform diversity. This work provides guidelines on which (or whether) error-correction tools should be used, depending on the application type.

Benchmarking software: `https://gitlab.com/leoisl/LR_EC_analyser`

Key words: Long reads, RNA-sequencing, Nanopore, Error correction, Benchmark

1 INTRODUCTION

Recent advances in long-read sequencing technology have enabled the sequencing of RNA molecules, using either cDNA-based or direct RNA protocols from Oxford Nanopore (referred to as *ONT* or *Nanopore*) and Pacific Biosciences (*PacBio*). The Iso-Seq protocol from PacBio consists in a size selection step, sequencing of cDNAs, and finally a set of computational steps that produce sequences of full-length transcripts. ONT has three different experimental protocols for sequencing RNA molecules: cDNA transformation with amplification, direct cDNA (with or without amplification), and direct RNA.

Long-read sequencing is increasingly used in transcriptome studies (Sedlazeck *et al.*, 2018; Wang *et al.*, 2016; Byrne *et al.*, 2017; Oikonomopoulos *et al.*, 2016) as they better describe exon/intron combinations (Sedlazeck *et al.*, 2018). For instance the Iso-seq protocol has been used for isoform identification, including transcripts identification (Wang *et al.*, 2016), de novo isoform discovery (Li *et al.*, 2017) and fusion transcript detection (Weirather *et al.*, 2015). Nanopore has recently been used for isoform identification (Byrne *et al.*, 2017) and quantification (Oikonomopoulos *et al.*, 2016).

The sequencing throughput of long-read technologies is significantly increasing over the years. It is now conceivable to sequence a full eukaryote transcriptome using either only long reads, or a combination of high-coverage long and short (Illumina) reads. Unlike the Iso-Seq protocol that requires extensive *in silico* processing prior to primary analysis (Sahlin *et al.*, 2018), raw Nanopore reads can in principle be readily analyzed. Direct RNA reads also permit the analysis of base modifications (Workman *et al.*, 2018), unlike all other cDNA-based sequencing technologies. There also exist circular sequencing techniques for Nanopore such as INC-Seq (Li *et al.*, 2016) which aim at reducing error rates, at the expense of a special library preparation. With raw long reads, it is up to the primary analysis software (typically a mapping algorithm) to deal with sequences that have significant per-base error rate, currently around 13% (Weirather *et al.*, 2017).

In principle, a high error rate complicates the analysis of transcriptomes especially for the accurate detection of exon boundaries, or the quantification of similar isoforms and paralogous genes. Reads need to be aligned unambiguously and with high

base-pair accuracy to either a reference genome or transcriptome. Indels (i.e. insertions/deletions) are the main type of errors produced by long-read technologies, and they confuse aligners more than substitution errors (Sović *et al.*, 2016). Many methods have been developed to correct errors in RNA-seq reads, mainly in the short-read era (Tong *et al.*, 2016; Song and Florea, 2015). They no longer apply to long reads because they were developed to deal with low error rates, and principally substitutions. However, a new set of methods have been proposed to correct genomic long reads. There exist two types of long-read error-correction algorithms, those using information from long reads only (*self* or *non-hybrid* correction), and those using short reads to correct long reads (*hybrid* correction). In this article, we will report on the extent to which state-of-the-art tools enable to correct long noisy RNA-seq reads produced by Nanopore sequencers.

Several tools exist for error-correcting long reads, including ONT reads. Even if the error profiles of Nanopore and PacBio reads are different, the error rate is quite similar and it is reasonable to expect that tools originally designed for PacBio data to also perform well on recent Nanopore data. There is, to the best of our knowledge, very little prior work that specifically addresses error-correction of RNA-seq long reads. A notable exception is the PBcR tool, which is mainly designed for genomes but is also evaluated on a Iso-Seq transcriptome (Koren *et al.*, 2012). Here we will take the standpoint of evaluating DNA long-read error-correction tools on RNA-seq data, an application that was likely not considered by the respective tools authors.

We evaluate the following DNA hybrid correction tools: LoRDEC (Salmela and Rivals, 2014), NaS (Madoui *et al.*, 2015), PBcR (Koren *et al.*, 2012), proovread (Hackl *et al.*, 2014); and the following DNA self-correction tools: Canu (Koren *et al.*, 2017), daccord (Tischler and Myers, 2017), LoRMA (Salmela *et al.*, 2016), MECAT (Xiao *et al.*, 2017), pbdagcon (Chin *et al.*, 2013). A majority of hybrid correction methods employ mapping strategies to place short fragments on long reads and correct long read regions using the related short read sequences. But some of them rely on graphs to create a consensus that is used for correction. These graphs are either k-mer graphs (de Bruijn graphs), or nucleotide graphs resulting from multiple alignments of sequences (partial order alignment). For self-correction methods, strategies using the aforementioned graphs are the most common. LSCPlus, a RNA-seq correction tool designed for PacBio reads, was not evaluated as the software webpage was unreachable (Hu *et al.*, 2016). We have selected what we believe is a representative set of tools but there also exist other tools that were not evaluated in this study, e.g. HALC (Bao and Lan, 2017), Falcon_sense (Chin *et al.*, 2016), HG-Color (Morisse *et al.*, 2018), HECIL (Choudhury *et al.*, 2018), MIRCA (Kchouk and Elloumi, 2016), Jabba (Miclotte *et al.*, 2016), nanocorr (Goodwin *et al.*, 2015), nanopolish (Loman *et al.*, 2015), and Racon (Vaser *et al.*, 2017).

Other works have evaluated error correction tools in the context of DNA sequencing. LRCStats evaluates error-correctors in a simulated framework, without the need to align corrected reads (La *et al.*, 2017). A technical report from Bouri and Lavenier (2017) provides an extensive evaluation of PacBio/Nanopore error-correction tools, in the context of de novo assembly. Perhaps the closest work to ours is the AlignQC software (Weirather *et al.*, 2017), which provides a set of metrics for the evaluation of RNA-sequencing long-read dataset quality. In Weirather *et al.* (2017) a comparison is provided between Nanopore and PacBio RNA-sequencing datasets in terms of error patterns, isoform identification and quantification. While Weirather *et al.* (2017) did not compare error-correction tools, we will use and extend AlignQC metrics for that purpose.

In this article, we will focus on the qualitative and quantitative measurements of error-corrected long reads, with transcriptomic features in mind. First we examine basic metrics of error-correction, e.g. mean length, base accuracy, homopolymers errors, and performance (running time, memory) of the tools. Then we ask several questions that are specific to transcriptome applications: (i) how is the number of detected genes, and more precisely the number of genes within a gene family, impacted by read error correction? (ii) Can error correction significantly change the number of reads mapping to genes or transcripts, possibly affecting downstream analysis based on these metrics? (iii) Do error-correction tools perturb isoform diversity, e.g. by having a correction bias towards the major isoform? (iv) What is the impact of error correction on identifying splice sites? To answer these questions, we provide an automatic framework (LC_EC_analyser, see Methods) for the evaluation of transcriptomic error-correction, that we apply to nine different error-correction tools.

2 RESULTS

2.1 Error-correction tools

Tables 1 and 2 present the main characteristics of respectively the hybrid and non-hybrid error-correction tools that were considered in this study. For the sake of reproducibility, in the Supplementary Material Section S1 are described all the versions, dependencies, and parameters. Note that these error-correction tools were all tailored for DNA-seq data except for PBcR. PBcR was ran only in hybrid mode, as the authors suggest using Canu over the non-hybrid mode.

2.2 Evaluation datasets

Our evaluation dataset consists of a single 1D Nanopore run using the cDNA preparation kit of RNA material taken from a mouse brain. We obtained 1,256,967 Nanopore 1D reads representing around 2 Gbp of data with an average size of 1650 bp and a N50 of 1885 bp. An additional Illumina dataset containing 58 million paired-end 151 bp reads was generated using a different cDNA protocol. The Nanopore and Illumina reads from the mouse RNA sample are available in the ENA repository under the following study: PRJEB25574.

2.3 Error-correction improves base accuracy and affects the number of detected genes

Tables 3 and 4 show an evaluation of error-correction based on AlignQC results, for the hybrid and non-hybrid tools, respectively. The per-base error rate is 13.7% in raw reads, 0.3-4.5% for reads corrected using hybrid methods and 2.9-6.4% with self-correctors. As expected the correction rate is better for hybrid correctors leading to a per-base error rate lower than 1% (except for LoRDEC and Proovread/untrimmed, which was equal to 4.5% and 2.6% resp.) because they use additional information from short Illumina reads

Table 1. Main characteristics of the hybrid error correction tools considered in this study

	LoRDEC	NaS	PBcR	Proovread
Reference	Salmela and Rivals (2014)	Madoui *et al.* (2015)	Koren *et al.* (2012)	Hackl *et al.* (2014)
Context	DNA	DNA	mRNA or DNA	DNA
Technology	PacBio or ONT	ONT	PacBio or ONT	PacBio
Main algorithmic idea	Construction of short read dBG, path search between k-mers in long reads	Recruitment of short reads by alignment to long reads, assembly of short reads to correct the long reads	Alignment of short reads to long reads and consensus.	Alignment of short reads to long reads and consensus.

Table 2. Main characteristics of the non-hybrid (self) error correction tools considered in this study

	Canu	daccord	LoRMA	MECAT	pbdagcon
Reference	Koren *et al.* (2017)	Tischler and Myers (2017)	Salmela *et al.* (2016)	Xiao *et al.* (2017)	Chin *et al.* (2013)
Context	DNA	DNA	DNA	DNA	DNA
Technology	PacBio or ONT	PacBio	PacBio or ONT	PacBio or ONT	PacBio
Main algorithmic idea	All-versus-all read overlap, filtering, alignment, DAG from the alignments, highest weight path search.	Multiple dBGs built from overlapping window of long reads alignments, consensus per window	Path search in dBG and multi-iterations.	k-mer based read matching, pairwise alignment between matched reads, alignment-based consensus calling on trivial regions, local POG-based consensus calling on complicated regions.	Align long reads to ”backbone” sequences, correction by iterative directed acyclic graph consensus calling from the multiple sequence alignments.

to correct the long reads. The error rate is around 4-6% for self-correction algorithms, except for LoRMA that reached 2.91%. A detailed error-rate analysis will be carried in Section 2.4.

In terms of number of reads after the correction step, LoRDEC, Proovread/untrimmed, daccord/untrimmed, and pbdagcon returned a number of reads similar to that of the uncorrected (raw) reads. All other softwares split and/or discard reads, likely because full-length error-correction was deemed impossible in some reads. PBcR and LoRMA tend to split reads into two or more shorter reads during the correction step, as they return $\sim$2x more reads after correction that are also shorter (mean length of respectively 776bp and 497bp, versus 2011bp in raw reads) and overall have significantly less bases in total (loss of respectively 298Mbp and 553Mbp). Canu and MECAT mostly discarded reads (30-33%) resulting in 14-25% less bases in total, with comparable mean length to other tools. Apart from LoRDEC, Proovread/untrimmed, and daccord (trimmed and untrimmed) for which only 85-94% of reads were mapped, corrected reads from all the other tools were mapped at a rate of 98.2-99.4%, showing a significant improvement over raw reads (mapping rate of 83.5%).

Apart from Canu, tools with high mean read length (*i.e.* LoRDEC, Proovread/untrimmed, daccord/untrimmed) showed the lowest percentages of mapped reads, indicating that trimming, splitting or discarding reads seems necessary in order to obtain shorter but overall less error-prone reads. A similar conclusion can be reached by comparing the results of trimmed and untrimmed versions of the same tool: reads corrected with Proovread and daccord in trimmed versions showed higher numbers of mapped reads and bases, and lower per-base error rates. However trimmed reads become 300-600 bases shorter on average, and around 2,000 genes are no longer detected. Therefore it is unclear whether trimming should always be performed by error-correctors in a transcriptomic context.

An important observation is that almost all tools, except for LoRDEC and Proovread/untrimmed, lost at least 1,000 genes after correction. Moreover, three of the tools that have the

Table 3. Statistics of hybrid error correction tools on the 1D run RNA-seq dataset. To facilitate the readability of this table and the next ones, we highlighted values that we deemed satisfactory in green colour, borderline in brown, and unsatisfactory in red, noting that such a classification is somewhat arbitrary.

	Raw	LoRDEC	NaS	PBcR	Proovread	Proovread trimmed
nb of reads	741k	741k	619k	1321k	738k	626k
mapped reads	83.5%	85.5%	98.7%	99.2%	85.5%	98.9%
mean length	2011	2097	1931	776	2117	1796
nb of bases	1313M	1394M	1179M	1015M	1400M	1112M
mapped bases[a]	89.0%	90.6%	97.5%	99.2%	92.4%	99.5%
per-base error rate[b]	13.72%	4.50%	0.38%	0.67%	2.65%	0.33%
nb of detected genes	16.8k (33.9%)	16.8k (33.9%)	15.0k (30.2%)	15.6k (31.4%)	16.6k (33.4%)	14.6k (29.5%)

[a] As reported by AlignQC. Percentage of bases aligned among mapped reads, taken by counting the M parts of CIGAR strings in the BAM file. Bases in unmapped reads are not counted.
[b] As reported by AlignQC, using a sample of 1 million bases from aligned reads segments.

Table 4. Statistics of non-hybrid error correction tools on the 1D run RNA-seq dataset.

	Raw	Canu	daccord	daccord trimmed	LoRMA	MECAT	pbdagcon
nb of reads	741k	519k	675k	840k	1540k	495k	778k
mapped reads	83.5%	99.1%	92.5%	94.0%	99.4%	99.4%	98.2%
mean length	2011	2193	2102	1476	497	1995	1473
nb of bases	1313M	1126M	1350M	1212M	760M	980M	1137M
mapped bases[a]	89.0%	92.0%	92.5%	94.7%	99.2%	96.9%	97.0%
per-base error rate[b]	13.72%	6.43%	5.20%	4.12%	2.91%	4.49%	5.65%
nb of detected genes	16.8k (33.9%)	12.4k (24.9%)	15.5k (31.3%)	13.9k (28.1%)	6.8k (13.7%)	10.4k (20.9%)	13.2k (26.5%)

[a] As reported by AlignQC. Percentage of bases aligned among mapped reads, taken by counting the M parts of CIGAR strings in the BAM file. Bases in unmapped reads are not counted.
[b] As reported by AlignQC, using a sample of 1 million bases from aligned reads segments.

highest number of detected genes (LoRDEC, Proovread/untrimmed, daccord/untrimmed) also have the lowest percentage of mapped reads, hinting that error correction might reduce gene diversity in favor of lower error-rate. It is noteworthy that for some tools (e.g. Canu, MECAT, LoRMA), the number of detected genes can drop by 26%-59% compared to the number of genes reported in raw reads.

Overall, no correction tool outperforms the others across all metrics. We note that a reasonable balance appears to be achieved by NaS and Proovread/trimmed, and that overall, hybrid correctors tend to outperform self-correctors.

2.4 Detailed error-rate analysis

The high error-rate of transcriptome long reads significantly complicates their primary analysis (Križanović *et al.*, 2018). While Section 2.3 presented a general per-base error rate, this section breaks down sequencing errors into several types and examines how each error-correction tool deals with them. The data presented here is a compilation of AlignQC results. Note that AlignQC computed the following metrics only on reads that could be aligned, thus unaligned reads are not counted, yet they may possibly be the most erroneous ones. AlignQC also subsampled aligned reads to around 1 million number of bases to calculate the presented values.

2.4.1 Deletions are the most problematic sequencing errors

Table 5 shows the error rate in the raw reads and in the corrected reads for each tool. In raw reads, deletions are the most prevalent type of errors (7.4% of bases), closely followed by subsitutions (5.1%), then insertions (1.2%). LoRDEC is the least capable of correcting mismatches (2% of them remaining), even though it is a hybrid tool. This is possibly related to the large amount of uncorrected reads in its output, 90k reads out of 741k (12%, as computed by exactly matching raw reads to corrected reads). The other hybrid tools result in less than 1% of substitution errors. Surprisingly, the non-hybrid tools also presented very low mismatches rates: all of them showed rates lower than 1%, except for Canu (1.33%) and daccord/untrimmed (1.1%). This suggests that the rate of systematic substitution errors in ONT data is low, as self-correctors were able to achieve results comparable to the

Table 5. Error rate in the raw reads and in the corrected reads for each tool, on the 1D run RNA-seq dataset, computed from 1M random aligned bases.

	Raw	LoRDEC	NaS	PBcR	Proovread	Proovread trimmed	Canu	daccord	daccord trimmed	LoRMA	MECAT	pbdagcon	pbdagcon trimmed
Error rate	13.72%	4.50%	0.38%	0.67%	2.65%	0.33%	6.43%	5.20%	4.12%	2.91%	4.49%	5.65%	5.71%
Mismatch	5.11%	2.04%	0.20%	0.18%	0.93%	0.13%	1.33%	1.10%	0.67%	0.37%	0.35%	0.50%	0.49%
Deletion	7.40%	2.15%	0.09%	0.30%	1.51%	0.18%	4.82%	3.82%	3.27%	2.51%	4.08%	5.06%	5.17%
Insertion	1.20%	0.32%	0.08%	0.19%	0.22%	0.03%	0.28%	0.28%	0.19%	0.03%	0.06%	0.09%	0.05%

Table 6. Homopolymer error rate in the raw reads and in the corrected reads for each tool, on the 1D run RNA-seq dataset, computed from 1M random aligned bases.

	Raw	LoRDEC	NaS	PBcR	Proovread	Proovread trimmed	Canu	daccord	daccord trimmed	LoRMA	MECAT	pbdagcon	pbdagcon trimmed
Homop. deletion	2.96%	0.77%	0.02%	0.10%	0.46%	0.04%	2.46%	2.14%	2.05%	1.82%	2.05%	2.26%	2.26%
Homop. insertion	0.38%	0.08%	0.01%	0.02%	0.06%	0.01%	0.08%	0.06%	0.03%	0.01%	0.01%	0.02%	0.01%

hybrid ones, even without access to Illumina reads. Still, the three best performing tools were all hybrid (Proovread/trimmed, PBcR and NaS), which should therefore be preferred for applications that require very low mismatch rates.

The contrast between self and hybrid tools is more visible on deletion errors. All hybrid tools outperformed the non-hybrid ones. Although in the hybrid ones, LoRDEC (2.15%) and Proovread/untrimmed (1.51%) still showed moderate rates of deletions, NaS, Proovread/trimmed and PBcR were able to lower the deletion error rate from 7.4% to less than 0.3%. All non-hybrid tools presented a high rate (3% or more) of deletion errors, except LoRMA (2.51%). This comparison suggests that ONT reads exhibit systematic deletions, that cannot be corrected without the help of Illumina data. The contribution of homopolymer errors will be specifically analyzed in Section 2.4.2. Considering insertion errors, all tools performed equally well. It is worth noting that more non-hybrid tools (LoRMA, pbdagcon/untrimmed, pbdagcon/trimmed and MECAT) achieved sub-0.1% insertions than hybrid tools (NaS and Proovread/trimmed).

Overall, hybrid tools outperformed non-hybrid ones in terms of error-rate reduction. However, the similar results obtained by both types of tools when correcting mismatches and insertions, and the contrast in correcting deletions, seem to indicate that the main advantage of hybrid correctors over self-correctors is the removal of systematic errors using Illumina data.

2.4.2 Homopolymer insertions are overall better corrected than deletions In this section we further analyze homopolymers indels, *i.e.* insertion or deletion errors consisting of a stretch of the same nucleotide. Table 6 shows that homopolymer deletions are an order of magnitude more abundant in raw reads than homopolymer insertions. It is worth noting that, by comparing the values for the raw reads in Tables 5 and 6, homopolymers are involved in 40% of all deletions and 31% of all insertions.

A closer look at Table 6 reveals that hybrid error correctors outperform non-hybrid ones, as expected, mainly as homopolymer indels are likely systematic errors in ONT sequencing. Hybrid correctors correct them using Illumina reads that do not contain such biases. Moreover, all tools performed well on correcting homopolymer insertions, reducing the rate from 0.38% to less than 0.1%. In particular, the hybrid tools NaS and Prooovread/trimmed, as well as the non-hybrid ones LoRMA, MECAT and pbdagcon/trimmed reached 0.01% homopolymer insertion error rate. Regarding homopolymer deletions, hybrid tools return less than 0.5% of them, except LoRDEC (0.77%). Non-hybrid tools performed more pooly, returning 1.8-2.4% of homopolymers deletion errors – a small improvement over the raw reads.

NaS and Proovread/trimmed showed the best reduction of homopolymers indels. It is also worth noting that hybrid correctors are able to correct homopolymer deletions even better than non-homopolymer deletions. For instance the ratio of homopolymer deletions over all deletions is 40% in raw reads, and decreases for all hybrid correctors, dropping to 20.2% for NaS and 25.6% for Proovread/trimmed, but increases to at least 43.8% (pbdagcon/trimmed) up to 72.6% (LoRMA) in non-hybrid tools (see Supplementary Material Section S3).

2.5 Error-correction perturbs the number of reads mapping to the genes and transcripts

Downstream RNA-sequencing analyses typically rely on the number of reads mapping to each gene and transcript for quantification, differential expression analysis, etc. In the rest of the paper, we define the **coverage** of a gene or a transcript as the

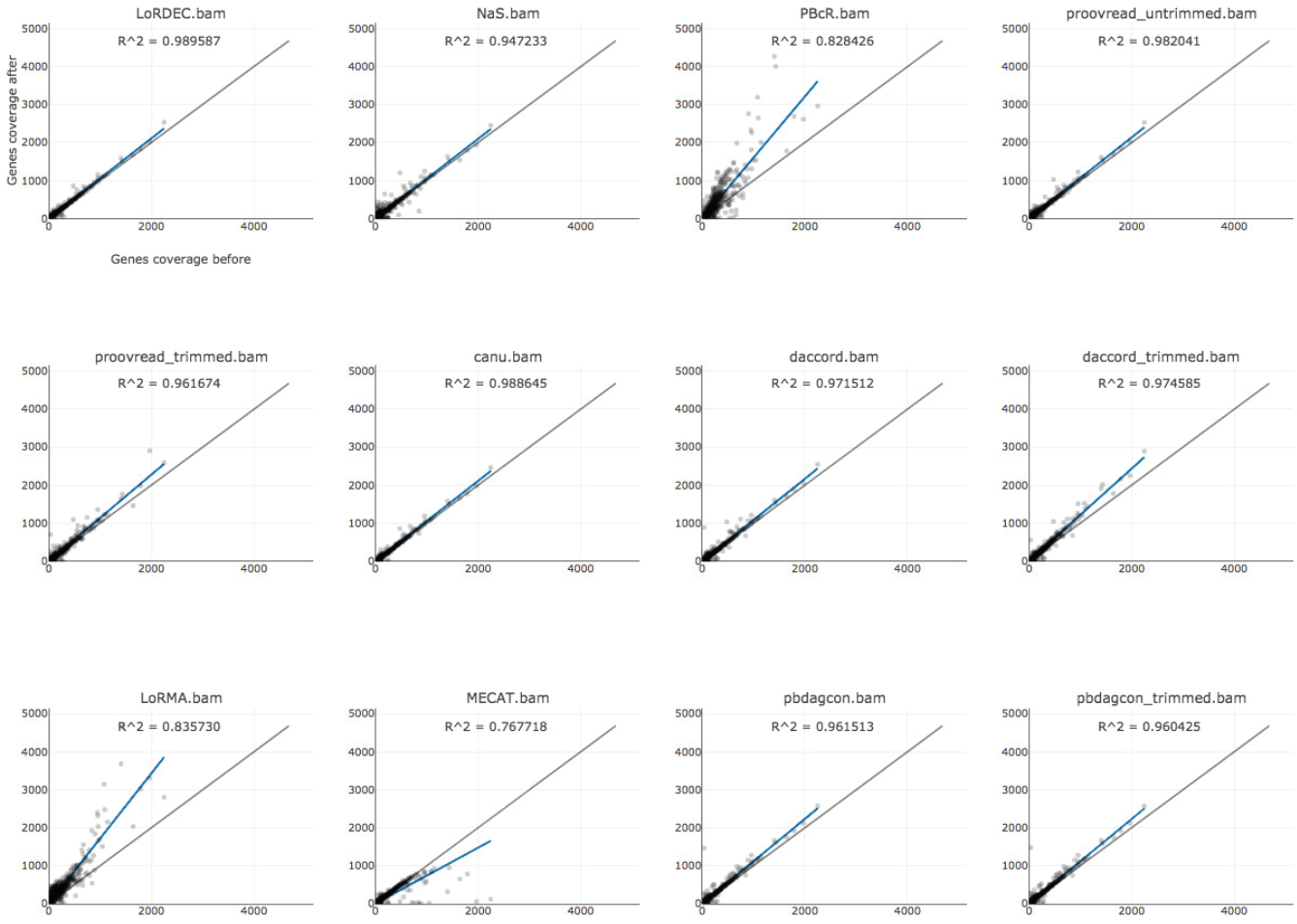


Fig. 1. Number of reads mapping to genes (C_G) before and after correction for each tool. The genes taken into account here were expressed in either the raw dataset or after the correction by the given tool.

number of reads mapping to it. For short we will refer to those coverages as C_G and C_T, respectively. In this section we investigate if the process of error correction can perturb C_G and C_T, which in turn would affect downstream analysis. Note that error correction could potentially slightly increase coverage, as uncorrected reads that were unmapped can become mappable after correction. Figure 1 shows the C_G before and after correction for each tool. PBcR is the only hybrid corrector that significantly inflates C_G, probably due to read splitting (see Section 2.3). Among self-correctors, LoRMA also inflates this value (also due to read splitting), while MECAT presents the lowest correlation and a significant drop in C_G. Besides these three tools, all the others present good correlation and the expected slight increase in C_G due to better mapping. All tools systematically presented a similar trend and lower correlation values on C_T (see Supplementary Material Figure S1), in comparison to C_G. This is expected, as it is harder for a tool to correct a read into its true isoform than into its true gene. The behaviour of the tools in the isoform level are in coherence with their behaviour in the gene level (C_G): PBcR and LoRMA inflates C_T; MECAT deflates; and all the others present a slight increase.

2.6 Error-correction perturbs gene family sizes

Tables 3 and 4 indicate that error correction results in a lower number of detected genes. In this section we explore the impact of error-correction on paralogous genes. By paralogous **gene family**, we denote a set of paralogs computed from Ensembl (see Section 4.3). Figure 2 represents the changes in sizes of paralogous gene families before and after correction for each tool, in terms of number of genes expressed within a given family. Overall, error-correctors do not strictly preserve the sizes of gene families. Correction more often shrinks families of paralogous genes than it expands them, likely due to erroneous correction in locations that are different between paralogs. In summary, 36-86% of gene families are kept of the same size by correctors, 1-12% are expanded and 6-61% are shrunk. Supplementary Material Figure S2 shows the magnitude of expansion/shrinkage for each gene family.

2.7 Error-correction perturbs isoform diversity

We further investigated whether error-correction introduces a bias towards the major isoform of each gene. Note that AlignQC does not directly address this question. To answer it, we computed the

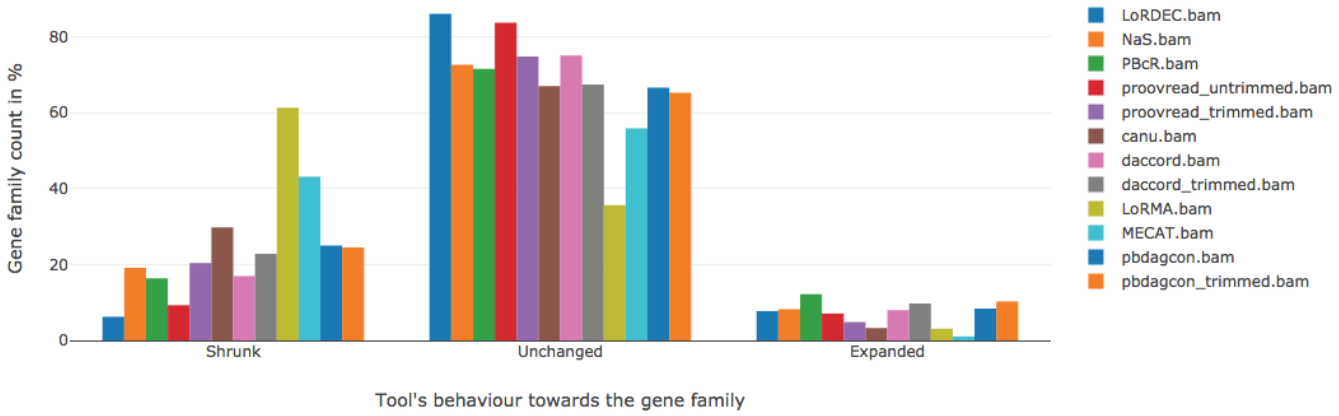


Fig. 2. Summary of gene family size changes across error-correction tools.

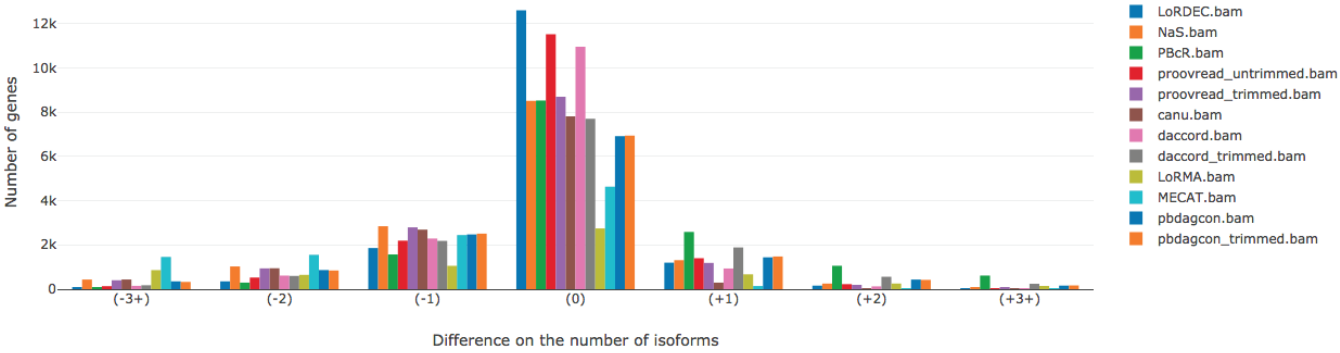


Fig. 3. Histogram of genes having more or less isoforms after error-correction.

following metrics: number of isoforms detected in each gene before and after correction by alignment of reads to genes, coverage of lost isoforms in genes having at least 2 expressed isoforms, and coverage of the major isoform before and after correction.

2.7.1 The number of isoforms varies before and after correction Figure 3 shows the number of genes that have the same number of isoforms after correction, or a different number of isoforms (-3, -2, -1, +1, +2, +3). In this Figure, only the genes that are expressed in both the raw and the corrected reads (for each tool) are taken into consideration. The negative (resp. positive) values indicate that isoforms were lost (resp. gained). We observe that a considerable number of genes (1k-3k) lose at least one isoform in all tools, which suggests that current methods reduce isoform diversity during correction. NaS and MECAT tend to lose isoforms the most, and PBcR identifies the highest number of new isoforms after correction. It is however unclear whether these lost and new isoforms are real (present in the sample) or due to mapping ambiguity. For instance, PBcR splits corrected reads into shorter sequences that may map better to other isoforms.

Overall, the number of isoforms is mostly unchanged in daccord/untrimmed, LoRDEC and Proovread/untrimmed. We observe that, counter-intuitively, trimming has a slight effect on the number of detected isoforms for Proovread and daccord but not for pbdagcon.

2.7.2 Multi-isoform genes tend to lose lowly-expressed isoforms after correction Figure 4 explores the relative coverage of isoforms that were possibly lost after correction, in genes having two or more expressed isoforms. The relative coverage of a transcript is the number of raw reads mapping to it over the number of raw reads mapping to its gene in total. Only the genes that are expressed in both the raw and the error-corrected reads (for each tool) are taken into consideration here. We anticipated that raw reads that map to a minor isoform are typically either discarded by the corrector, or modified in such a way that they now map to a different isoform, possibly the major one. The effect is indeed relatively similar across all correctors, except for MECAT that tends to remove a higher fraction of minor isoforms, and LoRDEC that tends to be the most conservative. This result suggests that current error-correction tool overall do not conservatively handle reads that belong to low-expression isoforms.

2.7.3 Coverage of the major isoform before and after correction To follow-up on the previous subsection, we investigate whether the coverage of the **major isoform**, *i.e.* the isoform with the highest expression in the raw dataset, increased after correction. In Figure 5, We observe that the coverage of the major isoform generally slightly increases after correction, except for MECAT, where its coverage decreases, likely due to a feature of MECAT's own correction algorithm. This indicates that error-correction tools

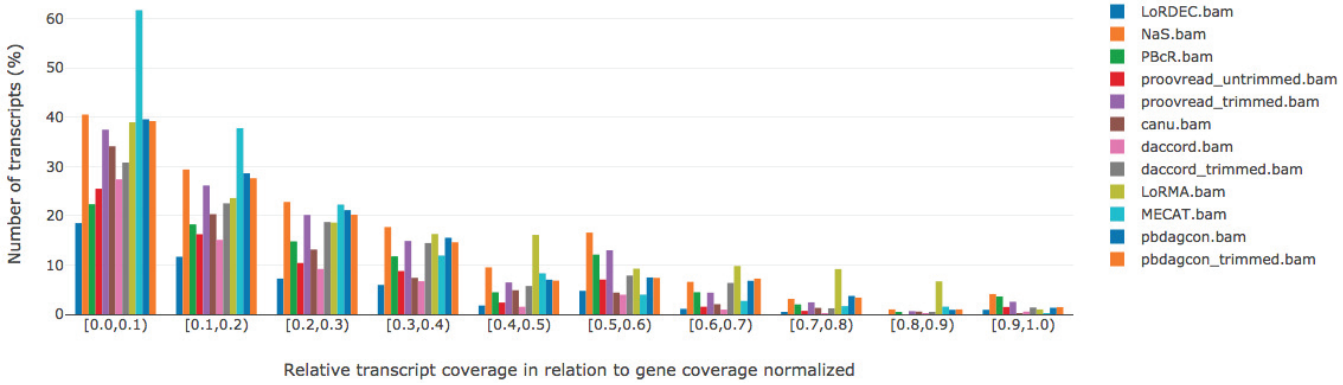


Fig. 4. Histogram of isoforms that are lost after correction, in relation to their relative transcript coverage, in genes that have 2 or more isoforms. The y axis reflects the percentage of isoforms lost in each bin. Absolute values can be found in the Supplementary Material Figure S3.

tend to correct reads towards the major isoform, but the effect is not pronounced. This is expected as the sum of expression of minor isoforms is, by nature, a small fraction of the total gene expression. Apart from LoRMA, MECAT and PBcR, where the correlations of the major isoform coverages are spurious (r^2 ¡ 0.77), other correctors tend to preserve this coverage after correction (r^2=0.90-0.96), with LoRDEC and Canu showing the highest correlations (96%). It is noteworthy that correction biases with respect to the major isoform do not appear to be specific to self correctors nor to hybrid correctors, but an effect that happens in both types of correctors.

2.7.4 Correction towards the major isoform is more prevalent when the alternative exon is small In order to observe if particular features of alternative splicing have an impact on error-correction methods, we designed a simulation over two controlled parameters: skipped exon length and isoform relative expression ratio. Using a single gene, we created a mixture of two simulated alternative transcripts: one constitutive, one exon-skipping. Several simulated read datasets were created with various relative abundances between major and minor isoform (in order to model a local differential in splicing isoform expression), and sizes of the skipped exon. Due to the artificial nature and small size of the datasets, many of the error-correction methods could not be run. We thus tested these scenarii on a subset of the correction methods.

In Figure 6, we distinguish results from hybrid and self-correctors, presented with respectively 100x coverage of short reads and 100x coverage of long reads, and only 100x coverage of long reads. Results on more shallow coverage (10x) and impact of simulation parameters on corrected reads sizes are presented in Supplementary Material Sections S7 and S8. Overall, hybrid correctors are less impacted by isoform collapsing than self-correctors. LoRDEC shows the best capacity to preserve isoforms in presence of alternatively skipped exons. However with less coverage, *e.g.* due to low-expressed genes and rare transcripts, all tools tend to mis-estimate the expression of isoforms (see Supplementary Material). Self-correctors generally have a minimum coverage threshold (only daccord could be run on the 10x coverage dataset of long reads, with rather erratic results, see Supplementary Material). Even with higher coverage, not all correctors achieve to correct this simple instance. Among all correctors, only LoRDEC seems to report the expected number of each isoforms consistently in all scenarios. We could not derive any clear trend concerning the relative isoform ratios, even if the 90% ratio seems to be in favor of overcorrection towards the major isoform. Skipped exon length seems to impact both hybrid and self correctors, small exons being a harder challenge for correctors.

2.8 Error-correction affects splice site detection

The identification of splice sites from RNA-seq data is an important but challenging task (Kaisers *et al.*, 2017). When mapping reads to a (possibly annotated) reference genome, mapping algorithms typically guide spliced alignments using either a custom scoring function that takes into account common splices sites patterns (e.g. GT-AG), and/or a database of known junctions. With long reads, the high error rate make precise splice site detection even more challenging, as indels (see Section 2.4) confuse aligners, shifting predicted spliced alignments away from true splice sites.

In this section, we evaluate how well splice sites are detected before and after error-correction. Figure 7 shows the number of correctly and incorrectly mapped splice sites for the raw and corrected reads, as computed by AlignQC. One would expect that a splice site is correctly detected when little to no errors are present in reads mapping around it. Thus, as expected, the hybrid error correction tools present a clear advantage over the non-hybrid ones, as they better decrease the per-base error rate. In the uncorrected reads, 27% of the splice sites were incorrectly mapped, which is brought down to between 0.28% (Proovread/trimmed) and 2.43% (LoRDEC) with hybrid correction tools. Among non-hybrid tools, LoRMA presented the lowest proportion of incorrectly detected splice sites (3.04%), however it detects 3.5-7x less splice sites (280k) than the other tools (which detect around 1-2 million splice sites). The other non-hybrid tools incorrectly detected splice sites at a rate between 5.61% (daccord/trimmed) and 11.95% (Canu). A detailed analysis of the incorrectly mapped splice sites can be found in the Supplementary Material Section S9.

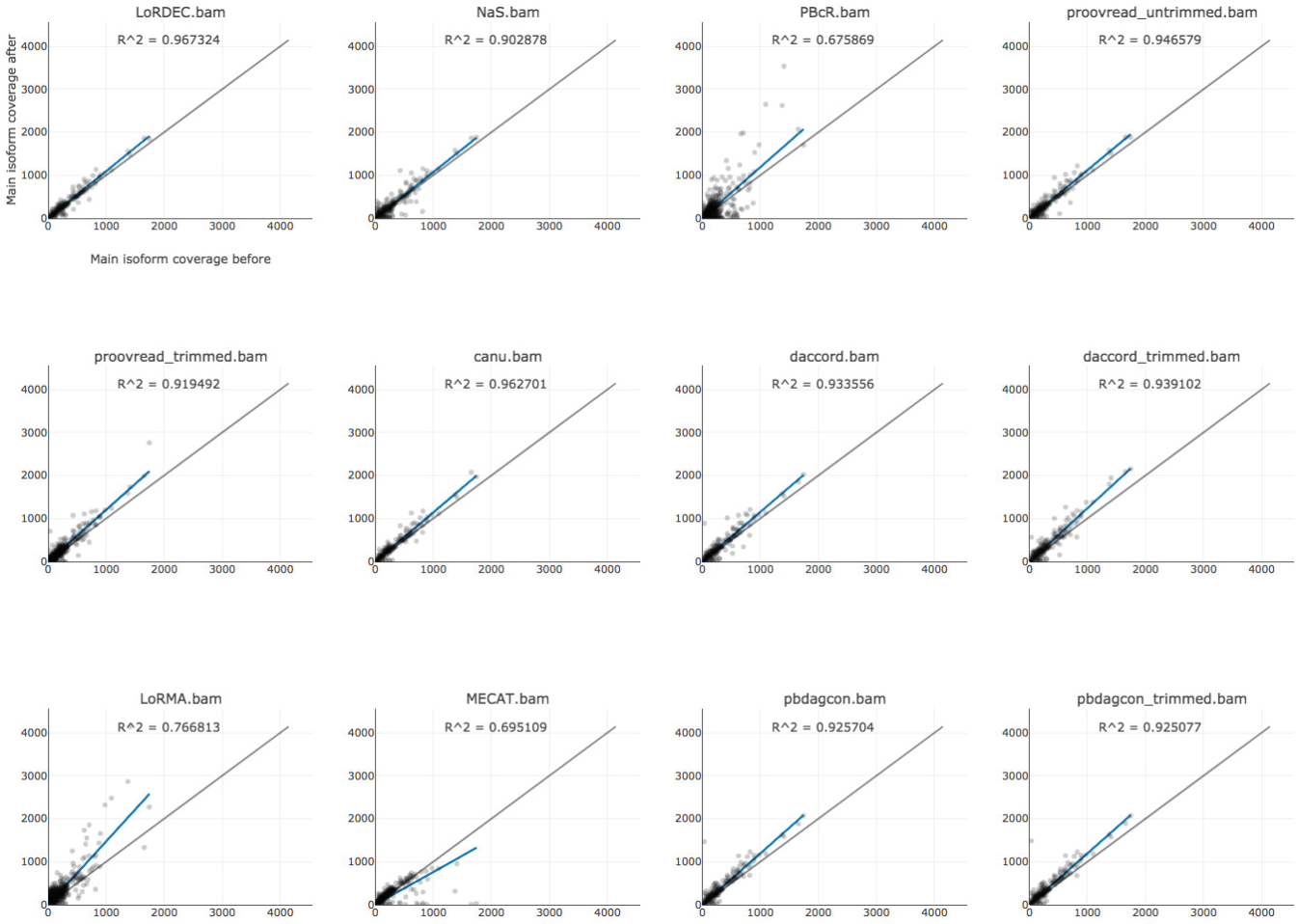


Fig. 5. Coverage of the major isoform of each gene before and after error-correction. The x-axis reflects the number of reads mapping to the major isoform of a gene before correction, and the y-axis is after correction.

2.9 Running time and memory usage of error-correction tools

Table 7 shows the running time and memory usage of all evaluated tools, measured using `GNU time`. The running time shown is the elapsed wall clock time (in hours) and the memory usage is the maximum resident set size (in gigabytes). All tools were ran with 32 threads. Overall, all tools were able to correct the dataset within 0.3-7 hours except for PBcR, NaS and Proovread, which took 63-116 hours, but also achieved the three lowest post-correction error rates in Table 3. In terms of memory usage, all tools required less than 10 GB of memory except PBcR, proovread and LoRMA, which required 53-166 GB. It is worth noting, however, that hybrid error correctors have to process massive Illumina datasets, which contributes to them taking higher CPU and memory usage for correction.

3 DISCUSSION

This work shed light on the versatility of long-read DNA error-correction methods, which can be successfully applied to error-correction of RNA-sequencing data as well. In our tests, error rates can be reduced from 13.7% in the original reads down to as low as 0.3% in the corrected reads. This is perhaps an unsurprising realization as the error-correction of RNA-sequencing data presents similarities with DNA-sequencing data, however this comes with a collection of caveats that we described in the Results section. Most importantly, the number of genes detected by alignment of corrected reads to the genome was reduced significantly by most error-correction methods. Furthermore, depending on the method, error-correction results have a more or less pronounced bias towards correction to the major isoform for each gene, jointly with a loss of the most lowly-expressed isoforms. We provided a software that enables automatic benchmarking of long-read RNA-sequencing error-correction software, in the hope that future error-correction methods will take advantage of it to avoid biases.

The summary statistics of error-corrected data (number of corrected reads, mean length, percentage of mapped reads, per-base error rate, number of detected genes) reveal that no tool outperforms

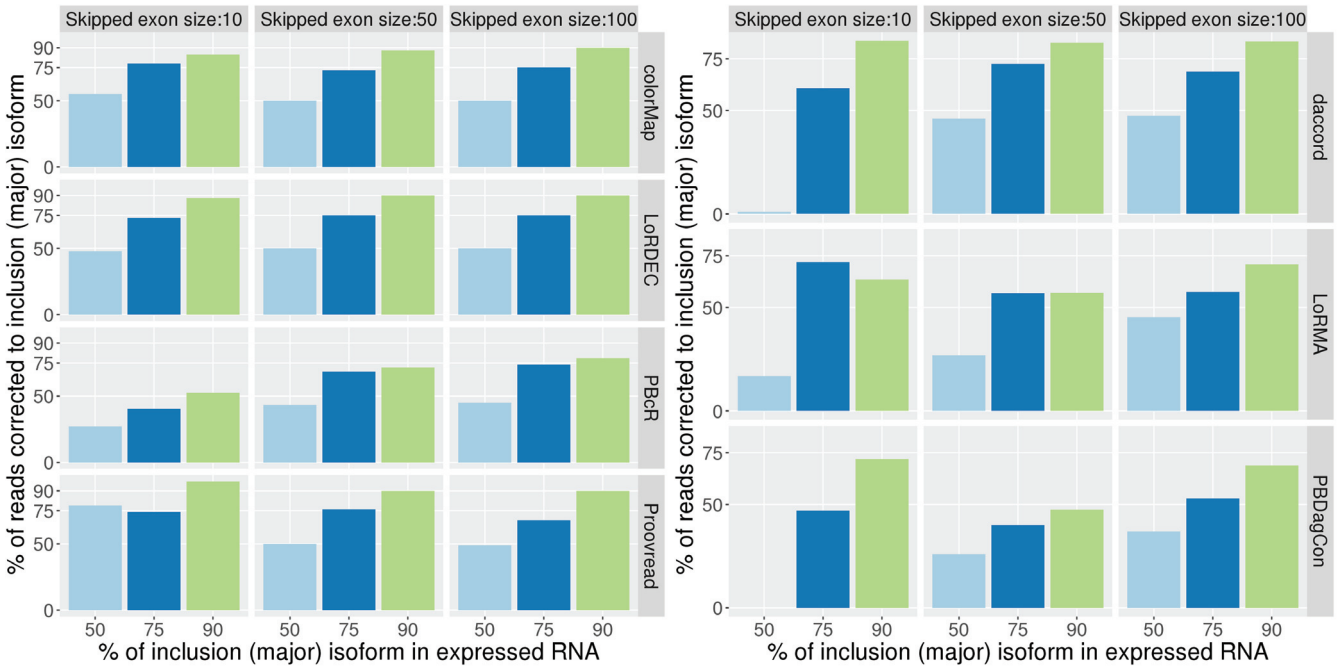


Fig. 6. Mapping of simulated raw and error-corrected reads to two simulated isoforms, and measurements of the percentage of reads mapping to the major isoform. The two isoforms represent an alternatively skipped exon of variable size: 10 bp, 50 bp, 100bp. Left: isoform structure conservation using 100X short reads coverage and 10X long reads, using three error-correction programs, one per row: LoRDEC, PBcR, proovread. Right: same with three self-correctors and 100X long reads: daccord, LoRMA and pbdagcon. Columns are alternative exon sizes. Bars are plots for each isoform ratio (50%; 75% and 90%) on the x-axis. On the y-axis, the closer a bar is to its corresponding ratio value on the x, the better. For instance, the bottom left light blue bar corresponds to a 50% isoform ratio with an exon of size 10, and we do not retrieve a 50% ratio after correction with Proovread (the bar does not go up to 50% on the vertical axis, but around 75% instead). The same layout applies to the right plot, where self-correctors are presented.

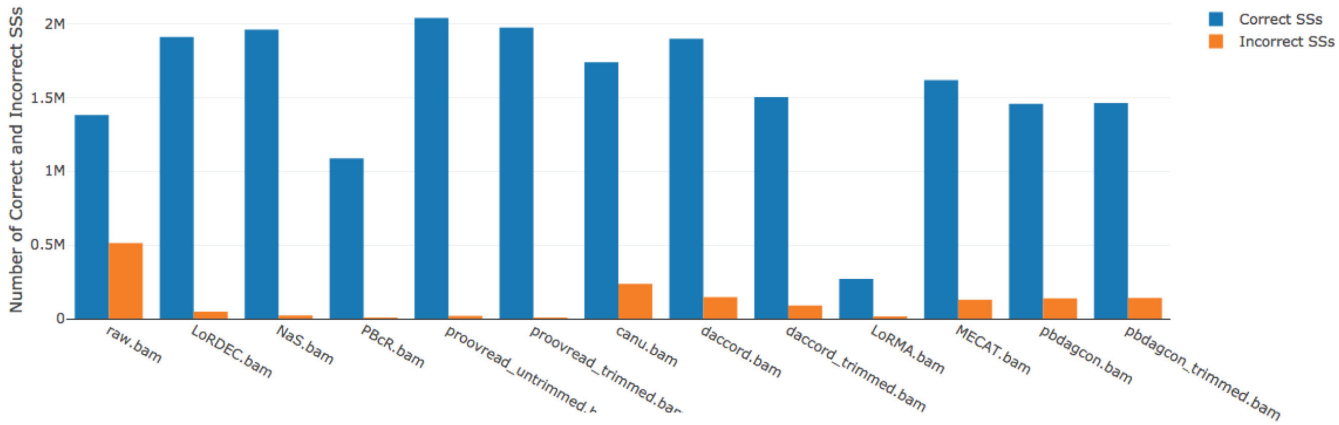


Fig. 7. Statistics on the correctly and incorrectly mapped splice sites (abbreviated SSs) for the uncorrected (raw) and corrected reads.

the others across all metrics, yet a reasonable balance is achieved by NaS and Proovread/trimmed, and that hybrid correction tools generally outperformed the self-correctors.

Detailed error-rate analysis showed that while hybrid correctors have lower error rates than self-correcters, the latter achieved comparable performance to the former in correcting substitutions and insertions. Deletions appear to be caused by systematic sequencing errors, making them fundamentally hard (or even impossible) to address in a self-correction setting. Moreover PBcR, NaS, and Proovread are the most resource-intensive error-correction tools, but also are the only correctors able to reduce base error rate below 0.7%.

We note that LoRDEC, PBcR, Proovread/untrimmed, daccord/untrimmed, and to a lower extent NaS, were able to preserve the number of detected genes better than other correctors. Among those, LoRDEC, Proovread/untrimmed and daccord/untrimmed appear to

Table 7. Running time and memory usage of error-correction tools on the 1D run RNA-seq dataset

	LoRDEC	NaS[1]	PBcR[2]	Proovread	Canu	daccord[3]	daccord trimmed[3]	LoRMA[4]	MECAT	pbdagcon[3]	pbdagcon[3] trimmed
Running time	2.4h	63.2h	116h	107.1h	0.7h	6.9h (7.4h)	6.6h (7.1h)	3.4h	0.3h	5.7h (6.2h)	5.6h (6.1h)
Memory usage	5.6GB	3GB	166.5GB	53.6GB	2.2GB	6.9GB (27.2GB)	6.8GB (27.2GB)	79GB	9.9GB	6.4GB (27.2GB)	6.4GB (27.2GB)

[1]NaS was ran in batches on a different system (TGCC cluster) than other tools; total running time was estimated based on subset of batches.
[2]PBcR was ran on a machine different from the others.
[3]daccord and pbdagcon need DAZZ_DB and DALIGNER to be ran before performing their correction. DAZZ_DB execution time and memory usage was disregarded due to being negligible. DALIGNER, however, took 0.5h and 27.2Gb of RAM. The runtime in parenthesis denotes the runtime of the tool + DALIGNER. The memory usage in parenthesis denotes the maximum memory usage between the tool and DALIGNER.
[4]LoRMA was using more than its allocated 32 cores in some (short) periods of time during the run.

also better preserve the number of detected isoforms better than other correctors. All tools tend to lose lowly-expressed isoforms after correction. This is expected, as these tools were mainly tailored to process DNA data where heterogeneous coverage is not expected. Furthermore, hybrid correctors outperformed self-correctors in the correction of errors near splice site junctions.

As a result, we conclude that no evaluated corrector is the most suited in all situations, and the choice should be guided by the downstream analysis. For quantification, we have shown that error-correction introduces undesirable coverage biases, as per Section 2.5, therefore we would recommend avoiding this step altogether. For isoform detection, LoRDEC, Proovread/untrimmed (hybrid) and daccord/untrimmed (non-hybrid) appear to be the methods of choice as they result in the the highest number of detected genes in Tables 3 and 4 and also preserve the number of detected isoforms as per Section 2.7. For splice site detection, we recommend using hybrid correctors, preferably NaS, PBcR or Proovread, as per Section 2.8. The same three tools (however, Proovread should be in trimmed mode) are also recommended if downstream analyses require very low general error rate. Finally for all other applications, NaS and Proovread/trimmed achieve a reasonable balance across all metrics.

In our analysis, we used a single mapping software (GMAP) to align raw and error-corrected reads, as in previous benchmarks (Weirather *et al.*, 2017; Križanović *et al.*, 2018). We note that other long-read mapping software have since been published, e.g. minimap2 (Li, 2018), which may increase the percentage of mapped read across all methods.

Furthermore, we only focused our evaluation on a single data type: 1D cDNA Nanopore data, using Illumina data for hybrid correction. While it would be natural to also evaluate PacBio data, we note that data from the PacBio Iso-Seq protocol is of different nature as the reads are pre-corrected by circular consensus.

As a side note, AlignQC reports that raw reads contained 1% of chimeric reads, i.e. either portions of reads that align to different loci, or to overlapping loci. The number of chimeric reads after error-correction remains in the 0.7%-1.3% range except for PBcR (0.1%), Proovread/trimmed (0.1%), MECAT (0.1%) and LoRMA (0.04%), which either correctly split reads or discarded chimeric ones.

In the evaluation of tools, we did not record the disk space used by each method, yet we note that it may be a critical factor for some tools (e.g. Canu) on larger datasets. We note also that genes that have low Illumina coverage are unlikely to be well corrected by hybrid correctors. Therefore our comparison does not take into account differences in coverage biases between Illumina and Nanopore data, which may benefit self-correctors. Finally, transcript and gene coverages are derived from the number of long reads aligning to a certain gene/transcript. This method enables to directly relate the results of error-correction to transcript/gene counts, but we note that in current RNA-seq analysis protocols, transcript/gene expression is still generally evaluated using short reads.

4 METHODS

4.1 Nanopore library preparation and sequencing

RNA MinION sequencing cDNA were prepared from 4 aliquots (250ng each) of mouse commercial total RNA (brain, Clontech, Cat# 636601), according to the Oxford Nanopore Technologies (Oxford Nanopore Technologies Ltd, Oxford, UK) protocol "1D cDNA by ligation (SQK-LSK108)". The data generated by MinION software (MinKNOW 1.1.21, Metrichor 2.43.1) were stored and organized using a Hierarchical Data Format. FASTA reads were extracted from MinION HDF files using poretools (Loman and Quinlan, 2014).

4.2 Illumina library preparation and sequencing

RNA-Seq library preparations were carried out from 500 ng total RNA using the TruSeq Stranded mRNA kit (Illumina, San Diego, CA, USA), which allows mRNA strand orientation (sequence reads occur in the same orientation as anti-sense RNA). After quantification by qPCR, each library was sequenced using 151 bp paired end reads chemistry on a HiSeq4000 Illumina sequencer. Reads were filtered *in silico* to remove mtRNA and rRNA using `BLAT` and `est2genome`.

4.3 Reference-based evaluation of long read error correction

A tool coined LR_EC_analyser, available at `https://gitlab.com/leoisl/LR_EC_analyser`, was developed using the Python language to analyze the output of long reads error correctors. The required arguments are the BAM files of the raw and corrected reads aligned to a reference annotated genome, as well as the reference genome in

Fasta file format and the reference annotation in GTF file format. A file specifying the paralogous gene families can also be provided if plots on gene families should be created. The main processing involves running the AlignQC software (Weirather *et al.*, 2017) (`https://github.com/jason-weirather/AlignQC`) on the input BAMs and parsing its output to create custom plots. It then aggregates information into a HTML report. For example, Tables 3 – 6 are compilations from AlignQC results, as well as Figure 7. Figures 1 – 5 were created processing text files built by AlignQC called "Raw data" in their output. In addition, an in-depth gene and transcript analysis can be performed using the IGV.js library (`https://github.com/igvteam/igv.js`). In this paper, we did not include all plots and tables created by the tool. To visualise the full latest reports, visit `https://leoisl.gitlab.io/LR_EC_analyser_support/`.

More specifically, in this work we aligned the raw and corrected reads to the Ensembl r87 Mus Musculus unmasked reference genome using the GMAP software (version 2017-05-08 with parameters `-n 10`) (Wu and Watanabe, 2005). The GMAP parameters map those from the original AlignQC publication (Weirather *et al.*, 2015). Gene families were computed by selecting all paralogs from Ensembl r87 mouse genes with 80%+ identity. Note that paralogs from the same family may have significantly different lengths, and no threshold was applied with respect to coverage. The complete selection procedure is reported here: `https://gitlab.com/leoisl/LR_EC_analyser/blob/master/GettingParalogs.txt`.

4.4 Simulation framework for biases evaluation

In the simulation framework of Section 2.7.4, exons length and number were chosen according to resemble what is reported in eukaryotes (Sakharkar *et al.*, 2004) (8 exons, 200 nucleotides). A skipped exon, whose size can vary, was introduced in the middle of the inclusion isoform. Skipped exon can have a size of 10, 50 or 100 nt. We also allowed the ratio of minor/major isoforms (M/m) to vary. For a coverage of C and a ratio M/m, the number of reads coming from the major isoform is MC and the number of minor isoform reads is mC. We chose relative abundances ratios for the inclusion isoform as such: $90/10$, $75/25$ and $50/50$. All reads are supposed to represent the full-length isoform. Finally for hybrid correction input, short reads of length 150 were simulated along each isoform, with 10X and 100X coverage.

During the simulation, we produced two versions of each read. The reference read is the read that represents exactly its isoform, without errors. The uncorrected read is the one in which we introduced errors. We used an error rate and profile that mimics observed R9.4 errors in ONT reads (total error rate of $\sim$13%, broken down as $\sim$5% of substitutions, $\sim$1% of insertions and $\sim$7% of deletions). After each corrector was applied to the read set, we obtained a triplet (reference, uncorrected, corrected) read that we used to assess the quality of the correction under several criteria.

We mapped the corrected reads on both exclusion and inclusion reference sequences using a fast Smith-Waterman implementation (Zhao *et al.*, 2013), from which we obtained a SAM file. It is expected that exclusion corrected reads will map on exclusion reference with no gaps, and that a deletion of the size of the skipped exon will be reported when mapping them to the inclusion. For each read, if it could be aligned to one of the two reference sequences in one block (according to the CIGAR), then we assigned it to to this reference. If more blocks were needed, we assigned the read to the reference sequence with which the cumulative length of gaps is the loweest. We also reported the ratio between corrected reads size of each isoform kind and the real expected size of each reference isoform.

KEY POINTS

- Long-read transcriptome sequencing is hindered by high error rates that affect analyses such as the identification of isoforms, exon boundaries, open reading frames, and the creation of gene catalogues.
- This review evaluates the extent to which existing long-read DNA error correction methods are capable of correcting cDNA Nanopore reads.
- Existing tools significantly lower the error rate, but they also significantly perturb gene family sizes and isoform diversity.

ACKNOWLEDGEMENTS

LL acknowledges CNPq/Brazil for the support. This work was performed using the computing facilities of the CC LBBE/PRABI and the France Génomique e-infrastructure (ANR-10-INBS-09-08).

FUNDING

This work was supported by the French National Research Agency [ANR ASTER, ANR-16-CE23-0001]; and the Brazilian Ministry of Science, Technology and Innovation (in portuguese, Ministério da Ciência, Tecnologia e Inovação - MCTI) through the National Counsel of Technological and Scientific Development (in portuguese, Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq), under the Science Without Borders (in portuguese, Ciências Sem Fronteiras) scholarship grant [process number 203362/2014-4 to LL].

COMPETING INTERESTS

JMA is one of the authors of the NaS error-correction tool (Madoui *et al.*, 2015). However, this study was designed and performed with no bias towards this particular tool. JMA is part of the MinION Access Programme (MAP) and received travel and accommodation expenses to speak at Oxford Nanopore Technologies conferences.

BIOGRAPHICAL NOTE

All authors are part of the ASTER project (ANR ASTER) with the purpose of developing algorithms and software for analyzing third-generation sequencing data.

REFERENCES

Bao, E. and Lan, L. (2017). HALC: High throughput algorithm for long read error correction. *BMC Bioinformatics*, **18**(1), 204.

Bouri, L. and Lavenier, D. (2017). Evaluation of long read error correction software. Technical report, INRIA Rennes - Bretagne Atlantique ; GenScale.

Byrne, A., Beaudin, A. E., Olsen, H. E., Jain, M., Cole, C., Palmer, T., DuBois, R. M., Forsberg, E. C., Akeson, M., and Vollmers, C. (2017). Nanopore long-read RNAseq reveals widespread transcriptional variation among the surface receptors of individual B cells. *Nature Communications*, **8**, 16027.

Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., Turner, S. W., and Korlach, J. (2013). Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods*, **10**(6), 563–569.

Chin, C.-S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., O'Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., Cramer, G. R., Delledonne, M., Luo, C.,

Ecker, J. R., Cantu, D., Rank, D. R., and Schatz, M. C. (2016). Phased diploid genome assembly with single-molecule real-time sequencing. *Nature Methods*, **13**(12), 1050–1054.

Choudhury, O., Chakrabarty, A., and Emrich, S. J. (2018). HECIL: A Hybrid Error Correction Algorithm for Long Reads with Iterative Learning. *Scientific Reports*, **8**(1), 9936.

Goodwin, S., Gurtowski, J., Ethe-Sayers, S., Deshpande, P., Schatz, M. C., and McCombie, W. R. (2015). Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, **25**(11), 1750–6.

Hackl, T., Hedrich, R., Schultz, J., and Förster, F. (2014). proovread : large-scale high-accuracy PacBio correction through iterative short read consensus. *Bioinformatics*, **30**(21), 3004–3011.

Hu, R., Sun, G., and Sun, X. (2016). LSCplus: a fast solution for improving long read accuracy by short read alignment. *BMC bioinformatics*, **17**(1), 451.

Kaisers, W., Ptok, J., Schwender, H., and Schaal, H. (2017). Validation of Splicing Events in Transcriptome Sequencing Data. *International journal of molecular sciences*, **18**(6).

Kchouk, M. and Elloumi, M. (2016). Efficient Hybrid De Novo Error Correction and Assembly for Long Reads. In *2016 27th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 88–92. IEEE.

Koren, S., Schatz, M. C., Walenz, B. P., Martin, J., Howard, J. T., Ganapathy, G., Wang, Z., Rasko, D. A., McCombie, W. R., Jarvis, E. D., and Phillippy, A. M. (2012). Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature Biotechnology*, **30**(7), 693–700.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive ¡i¿k¡/i¿ -mer weighting and repeat separation. *Genome Research*, **27**(5), 722–736.

Križanović, K., Echchiki, A., Roux, J., and Šikić, M. (2018). Evaluation of tools for long read RNA-seq splice-aware alignment. *Bioinformatics*, **34**(5), 748–754.

La, S., Haghshenas, E., and Chauve, C. (2017). LRCstats, a tool for evaluating long reads correction methods. *Bioinformatics*, **33**(22), 3652–3654.

Li, C., Chng, K. R., Boey, E. J. H., Ng, A. H. Q., Wilm, A., and Nagarajan, N. (2016). INC-Seq: accurate single molecule reads using nanopore sequencing. *GigaScience*, **5**(1), 34.

Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**(18), 3094–3100.

Li, J., Harata-Lee, Y., Denton, M. D., Feng, Q., Rathjen, J. R., Qu, Z., and Adelson, D. L. (2017). Long read reference genome-free reconstruction of a full-length transcriptome from Astragalus membranaceus reveals transcript variants involved in bioactive compound biosynthesis. *Cell Discovery*, **3**, 17031.

Loman, N. J. and Quinlan, A. R. (2014). Poretools: a toolkit for analyzing nanopore sequence data. *Bioinformatics*, **30**(23), 3399–3401.

Loman, N. J., Quick, J., and Simpson, J. T. (2015). A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature Methods*, **12**(8), 733–735.

Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015). Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC genomics*, **16**(1), 327.

Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., and Fostier, J. (2016). Jabba: hybrid error correction for long sequencing reads. *Algorithms for Molecular Biology*, **11**(1), 10.

Morisse, P., Lecroq, T., and Lefebvre, A. (2018). Hybrid correction of highly noisy long reads using a variable-order de Bruijn graph. *Bioinformatics*.

Oikonomopoulos, S., Wang, Y. C., Djambazian, H., Badescu, D., and Ragoussis, J. (2016). Benchmarking of the Oxford Nanopore MinION sequencing for quantitative and qualitative assessment of cDNA populations. *Scientific Reports*, **6**(1), 31602.

Sahlin, K., Tomaszkiewicz, M., Makova, K. D., and Medvedev, P. (2018). Deciphering highly similar multigene family transcripts from Iso-Seq data with IsoCon. *Nature Communications*, **9**(1), 4601.

Sakharkar, M. K., Chow, V. T. K., and Kangueane, P. (2004). Distributions of exons and introns in the human genome. *In silico biology*, **4**(4), 387–93.

Salmela, L. and Rivals, E. (2014). LoRDEC: accurate and efficient long read error correction. *Bioinformatics*, **30**(24), 3506–3514.

Salmela, L., Walve, R., Rivals, E., and Ukkonen, E. (2016). Accurate self-correction of errors in long reads using de Bruijn graphs. *Bioinformatics*, **33**(6), btw321.

Sedlazeck, F. J., Lee, H., Darby, C. A., and Schatz, M. C. (2018). Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*, **19**(6), 329–346.

Song, L. and Florea, L. (2015). Rcorrector: efficient and accurate error correction for Illumina RNA-seq reads. *GigaScience*, **4**(1), 48.

Sović, I., Šikić, M., Wilm, A., Fenlon, S. N., Chen, S., and Nagarajan, N. (2016). Fast and sensitive mapping of nanopore sequencing reads with GraphMap. *Nature communications*, **7**, 11307.

Tischler, G. and Myers, E. W. (2017). Non Hybrid Long Read Consensus Using Local De Bruijn Graph Assembly. *bioRxiv*, page 106252.

Tong, L., Yang, C., Wu, P.-Y., and Wang, M. D. (2016). Evaluating the impact of sequencing error correction for RNA-seq data with ERCC RNA spike-in controls. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, volume 2016, pages 74–77. IEEE.

Vaser, R., Sović, I., Nagarajan, N., and Šikić, M. (2017). Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, **27**(5), 737–746.

Wang, B., Tseng, E., Regulski, M., Clark, T. A., Hon, T., Jiao, Y., Lu, Z., Olson, A., Stein, J. C., and Ware, D. (2016). Unveiling the complexity of the maize transcriptome by single-molecule long-read sequencing. *Nature Communications*, **7**, 11708.

Weirather, J. L., Afshar, P. T., Clark, T. A., Tseng, E., Powers, L. S., Underwood, J. G., Zabner, J., Korlach, J., Wong, W. H., and Au, K. F. (2015). Characterization of fusion genes and the significantly expressed fusion isoforms in breast cancer by hybrid sequencing. *Nucleic Acids Research*, **43**(18), e116–e116.

Weirather, J. L., de Cesare, M., Wang, Y., Piazza, P., Sebastiano, V., Wang, X.-J., Buck, D., and Au, K. F. (2017). Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research*, **6**, 100.

Workman, R. E., Tang, A., Tang, P. S., Jain, M., Tyson, J. R., Zuzarte, P. C., Gilpatrick, T., Razaghi, R., Quick, J., Sadowski, N., Holmes, N., Jesus, J. G. d., Jones, K., Snutch, T. P., Loman, N. J., Paten, B., Loose, M. W., Simpson, J. T., Olsen, H. E., Brooks, A. N., Akeson, M., and Timp, W. (2018). Nanopore native RNA sequencing of a human poly(A) transcriptome. *bioRxiv*, page 459529.

Wu, T. D. and Watanabe, C. K. (2005). GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, **21**(9), 1859–1875.

Xiao, C.-L., Chen, Y., Xie, S.-Q., Chen, K.-N., Wang, Y., Han, Y., Luo, F., and Xie, Z. (2017). MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *Nature Methods*, **14**(11), 1072–1074.

Zhao, M., Lee, W.-P., Garrison, E. P., and Marth, G. T. (2013). SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications. *PLoS ONE*, **8**(12), e82138.

Chapter 8

Assemblying alternative splicing events from short reads guided by accurate long reads

Preamble

Key points

- In preparation.

Status

In preparation, not being submitted before the defence.

Author contributions

L. is first author in this paper.

Assemblying alternative splicing events from short reads guided by accurate long reads

Leandro Lima*, Camille Sessegolo*, Blerina Sinaimeri*, Said Sadique Adi*, Marie-France Sagot*, Vincent Lacroix*

1 Introduction and background

Alternative splicing (AS) is an essential process in eukaryotic organisms, as evidenced by 90% of human multi-exonic genes undergoing it [31,49]. The study of AS can help to understand the transcriptome diversity expressed in a given set of cells in a particular condition, helping on the comprehension of diseases, development stages, stress-response mechanisms, etc. Despite its importance, AS remains underestimated, even in model species [49,43].

The most commonly used technique to study transcriptomes, and consequently AS, is through RNA sequencing. Many tools were developed to process RNA-seq reads when a reference genome or transcriptome is available. As examples, we can cite: i) splice-aware mappers [6,50,19,25,8,20]; ii) reference-based assemblers [47,45,15,36]; iii) reference-based algorithms to estimate expression levels [38,22,33,5,32].

The context this work is inserted in, however, concerns non-model species, where reference genomes or transcriptomes are not available. In this case, most *de novo* pipelines try to identify and quantify full-length isoforms by assembling RNA-seq reads, such as Oases [44], SOAPdenovo-Trans [51], Trans-ABySS [39] and Trinity [13]. The main advantages of *de novo* methods over reference-based methods are: i) they do not require any read-reference alignments and can therefore be applied when the genomic sequence is not available, is gapped, highly fragmented or substantially altered, as in cancer cells [13]; ii) they enable to discover transcripts that are missing or incomplete in the reference [16]. Their disadvantages include: i) the assembly of short reads is itself difficult, and only the most abundant transcripts are likely to be fully assembled [16]; ii) reconstruction heuristics are usually employed, which may lead to missing infrequent alternative transcripts while highly similar transcripts are likely to be assembled into a single transcript [29,27]; iii) they require more computational power than reference-based strategies.

As described, assembling full-length transcripts from short reads without a reference genome is challenging. Indeed, when the sequenced reads are short, and two transcripts have similar expression levels with a long enough constitutive region (longer than the fragments' length) flanked by two variable regions, the reads do not provide enough information to phase the two variable regions reliably, and any choice could be arguable. A recent solution to this problem was conceived due to advances in the sequencing technology. Such advances resulted in the maturity of third generation sequencers, *e.g.* PacBio and Oxford Nanopore, capable of sequencing long reads. In the RNA context, these technologies are being increasingly used as they better describe exon/intron combinations, and frequently sequence full-length transcripts, thus usually eliminating the assembly step and its related problems.

However, in many applications, the focus can be restricted to the exon level. Identifying which exons can be alternatively spliced is already very valuable. It has been shown that local assembly of AS events is more sensitive and precise than global assembly strategies from short read

* Authors list is provisional and is subject to modifications before the submission of this work.

data [43,27,4], due to the fact that assembling only the local variations between mRNAs is easier than assembling full-length transcripts, which requires discriminating very similar expressed transcripts, due to AS and expression of paralogous genes.

Therefore, we can say that long reads enable the study of full-length transcripts, while short reads are more appropriate for local assembly approaches. Indeed, long reads could also be used to study local events, like AS. However, there are two main issues with this approach. The first is that the cost of sequencing a long read is orders of magnitude higher than sequencing a short read. As such, usually just a fraction of the transcriptome, mostly the highly expressed isoforms, are covered by long reads. Nevertheless, a comprehensive AS study requires a deep sequencing in order to capture non-highly expressed mRNAs, and to correctly quantify all identified events. Short-read sequencing can dig deeper in the transcriptome, describing AS events not present in the long reads. The shallowness of long reads sequencing can be alleviated through special library preparations, such as normalization of the RNA libraries, to reduce over-represented transcripts [21]. Moreover, degradation of mRNA targets selected to be sequenced can be eliminated through 5'-cap selection, thus guaranteeing the sequencing of full-length mRNAs. Such techniques might decrease the throughput, but will better describe the transcriptome diversity in a set of cells. Even so, short reads are still able to dig deeper in the transcriptome. The second issue with using long reads to study AS is that third generation sequencing is currently hindered by high error rates that affect the identification of isoforms, exon boundaries, open reading frames, and the creation of gene catalogues. Although error-correction in long RNA-seq datasets is possible with correction algorithms tailored for the genomic context, such methods usually tend to truncate the transcriptome, an undesirable side effect [26]. However, accurate long reads can still be obtained natively, mainly through Pacific Biosciences (PacBio) SMRT Iso-Seq sequencing [38]. An alternative to Iso-seq data is to employ circular sequencing techniques for Nanopore, such as INC-Seq [23].

Although long-read sequencing is currently shallow and not as comprehensive as short-read sequencing to describe AS events, they are able to describe the complete structure of mRNAs, which is hard or impossible, in some cases, with short reads. The full-length sequencing of a given transcript provides a backbone or a guide to assemble AS events around the transcript. In this work, we therefore explore a hybrid AS assembly method, which makes use of both short and long reads, in order to list AS events in a comprehensive manner, thanks to short reads, guided by the full-length context provided by the long reads. Hybrid assembly of both types of RNA-seq reads in a *de novo* context has already been explored. Trinity [13] v2.0.2 release onwards improves the last step of assembly, the Butterfly algorithm, to better integrate long read support and to improve on the assembly of complex isoforms, particularly those containing internally repetitive sequences [14]. IDP-denovo [12] first assembles short reads into short-reads scaffolds (SR-scaffolds) through existing *de novo* assemblers of short read data only, then align long reads to SR-scaffolds to extend and fill potential gaps between the latter. Unaligned long reads are not discarded, but grouped into gene clusters. The extended SR-scaffolds and the gene clusters are used to create a pseudo-reference of exonic regions, *i.e.* a reference containing only the expressed regions for each gene, allowing the identification of alternative exon usage and splice sites. Finally, isoform abundance estimation is performed using IDP [2]. However, as previously shown in [43,27,4], the local assembly of AS events is more sensitive and precise than full-length transcriptome assembly strategies, when the input is only short reads. Therefore, we expect that this remains true also in the hybrid assembly scenario. By focusing on the specific goal of assembling only AS events, and not full-length transcripts, we predict that the method here described will be faster, more sensitive and precise than methods that focus on the hybrid global assembly of short and long RNA-seq reads, such as Trinity and IDP.

2 Methods

Our method receives as input shallow long reads, and deep short reads (both in Fasta or Fastq formats), and outputs local alternative splicing events that are described in the short reads, but not in the long reads, in Fasta format. It is composed by four main steps: 1) hybrid DBG construction; 2) exact mapping of long reads to the hybrid DBG; 3) Unitig Linking Graph construction; 4) alternative splicing events enumeration. The next sections explain each step in detail. We start however by providing some basic definitions.

2.1 Basic definitions

The next sections use the following basic definitions. Given a graph G, and a vertex $v \in G$, the out-neighbours (in-neighbours) of v in G are denoted by $N^+_G(v)$ ($N^-_G(v)$). We shall usually simplify all the notations by omitting the graph argument, when this is clear from the context. As such, the previous notations can be simplified to $N^+(v)$ and $N^-(v)$. In a DBG G built with a given value of k, the k-mer represented by the vertex $v \in G$ is denoted by $kmer(v)$. The abundance of a vertex $v \in G$, denoted by $a(v)$, is the number of times $kmer(v)$ appears in the reads datasets used to build G. The *relative* out-abundance (in-abundance) of an arc $e = (s,t) \in G$ is $ra^+(e) = a(t)/\sum_{v \in N^+(s)} a(v)$ ($rc^-(e) = a(t)/\sum_{v \in N^-(t)} a(v)$). A compressed de Bruijn graph (cDBG) C is obtained from a DBG G by replacing all the linear paths p in G by a vertex u such that the sequence associated to u (denoted by $seq(u)$) in C is the sequence spelled by p in G. The vertices of C are called unitigs. The size of a unitig u, denoted by $|u|$, equals the size of $seq(u)$. Observe that G and C encode the same information, but in practice the latter can be more efficiently stored in memory and algorithms usually run faster. A walk $w = v_1, v_2, ..., v_k$ of k vertices in a graph G is a sequence of vertices of G such that $(v_i, v_{i+1}) \in G$ for $1 \leq i \leq k-1$. A path is a walk with no repeated vertices.

2.2 Hybrid DBG construction

We now detail how we build a hybrid bicoloured DBG from both the short and long reads. We start by building a DBG G_S from the deep short reads. We assume that these reads contain few errors, *e.g.* 0.1%, which is common in the Illumina technology, the most used second-generation sequencer. We deal with sequencing errors by using two cut-offs. As commonly done in genomics, we first remove from the graph the non-solid k-mers. Solid k-mers are the vertices $v \in G_S | a(v) \geq a_{min}$, where a_{min} is the minimum abundance solidity threshold (parameter `-min_abundance`, defaulting to 2), representing a counting floor for the k-mers that are believed to be correctly sequenced (*i.e.* does not include a sequencing error). The second cut-off is a relative one, which is commonly applied in tools processing second generation RNA-seq reads, such as Trinity [13] and KisSplice [43]. The objective of the relative cut-off is to remove errors in highly-expressed transcripts. We do so by detecting and removing the arcs $e \in G_S | ra^+(e) < ra_{min}$ or $ra^-(e) < ra_{min}$, where ra_{min} is the minimum relative abundance threshold (parameter `-rel_cutoff`, defaulting to 0.02). By default, we apply low values for both cutoffs so that we do not miss infrequent isoforms.

Next, we build the DBG G_L from the long reads. However, we do not perform any sequencing error removal procedures on G_L. The main reason is that the sequencing is much more shallow, and applying the same cutoffs as in short reads would result in losing out many reads (*i.e.* many transcripts are supported by only one read). Our method is primarily designed for perfect long

reads. We discuss later how sequencing errors in long reads are expected to affect the performance of the method.

Finally, we build a hybrid DBG G in which we merge both graphs G_S and G_L. To do so, we first retrieve the unitigs of G_S and G_L, and then we build G by using such unitigs as input. Moreover, we colour each vertex $v \in G$ with the colour red, if it stems from the short reads, blue if it stems from the transcripts, or purple if it stems from both datasets. Finally, in order to be computationally efficient in the downstream steps, we compress the DBG G into the cDBG C by replacing its linear paths by unitigs. During this compression, we also associate each k-mer of G to the unitig it belongs to in C, using a vector $kmer2Unitig$ (more specifically, we associate each k-mer identifier to a unitig identifier). This will allow us to map a sequence to C efficiently in the next two steps.

We observe that if there are too many sequencing errors in the long reads, *i.e.* one at every k bases, both graphs G_S and G_L will hardly have common regions, and thus the merging of these graphs will not be appropriate. Therefore, our method works optimally for error-free long reads. Moreover, it is assumed to work partially (to be demonstrated) when the error rate is below 1 error every k bases, which can obtained using PacBio SMRT Iso-Seq sequencing [38], or Nanopore INC-Seq sequencing [23], or through error-correction algorithms [26]. In any case, the high-error-rate issue of long reads is being actively addressed by the community, through error-correction methods, special library preparation protocols, and advances in the sequencing technology. The expectation is thus that the long reads error rate will decrease significantly in the short future, while Illumina sequencing will continue to improve on reducing the per read cost to remain competitive. This is the situation in which this method performs appropriately.

In our implementation, all the aforementioned graphs are built using the GATB library [9].

2.3 Exact mapping of long reads to the hybrid DBG

For each long read l given as input, we map l to C by retrieving a walk $w(l) \in C$ spelling out l. To do so efficiently, we use the implementation of a minimum perfect hash function (MPHF) on the set of k-mers [28] used to build C. The MPHF allows to retrieve the identifier of a given k-mer in constant time in most cases[1]. As such, to map a long read l, we iterate through each k-mer of l, querying its identifier using the MPHF, and associating the k-mer identifier to its unitig identifier using the vector $kmer2Unitig$ (in constant time). As we have $\mathcal{O}(|l|)$ k-mers in l, this procedures takes expected $\mathcal{O}(|l|)$ time. We note here that, since we assume that the input long reads are accurate, and thus no sequencing-error-removal procedure was applied when building G_L, we do not need to take into account inexact mappings: there will always exist a walk $w \in C$ spelling out each long read. If few sequencing errors are indeed present in long reads, they will be interpreted as small variations (*e.g.* SNPs or indels), or will eventually be simplified in the next step. However, the vector $kmer2Unitig$ can lead to heavy memory usage. It is possible to store it more compactly by associating just a sample of the k-mers in each unitig. If such k-mers are spaced by a constant distance, we have a multiplicative reduction in memory consumption, while keeping the same asymptotic time for mapping. This simple idea is based on the sampling of suffix array positions in the FM-index [10], and is also already implemented in the sparse Pufferfish index [1]. For now, our method still lacks the implementation of this feature.

[1] The query time of the MPHF described by Limasset *et al.* in [28] can deteriorate to the query time of classical hash tables, in the worst case, but this is extremely rare to happen in practice.

2.4 Unitig Linking Graph construction

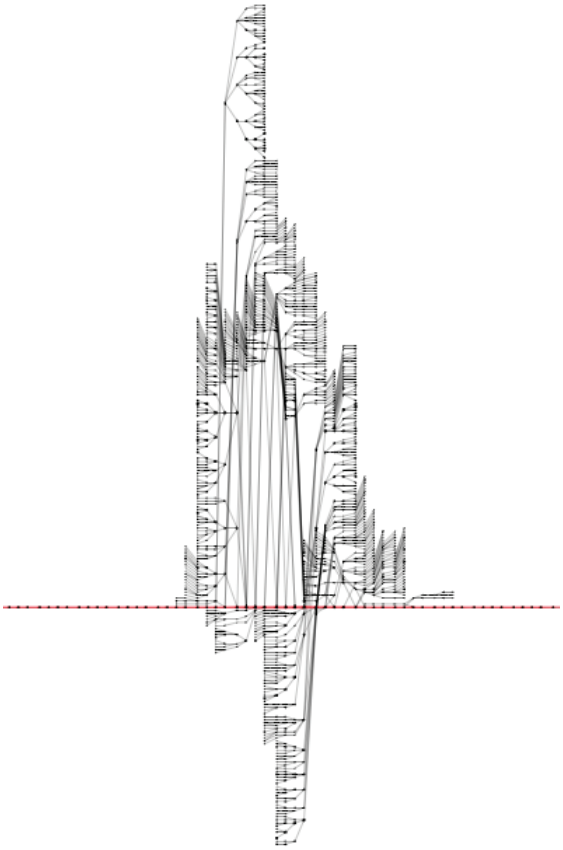
The Unitig Linking Graph (ULG) is an abstraction of the cDBG, which removes the complex parts of the graph, and connects the remaining parts using the read information, adding the range information given by single-end reads back to the graph, which was lost when cutting the reads into k-mers. The most complex parts of the graph are often associated to high-copy-number and low-divergence repeats (*i.e.* repeats that are present in many copies with very high similarity between them), which cannot be easily processed by an assembly algorithm(see Figure 1).

The main goal of the ULG is to solve repeats larger than k, but shorter than the reads' length. Longer repeats cannot be reliably solved using second generation data, since the read length is not enough to span such repeats. Observe that since our goal is to find alternative paths that are not contained in the long reads, we cannot use the long-range information given by long reads to solve the repeats present in short-read data only.

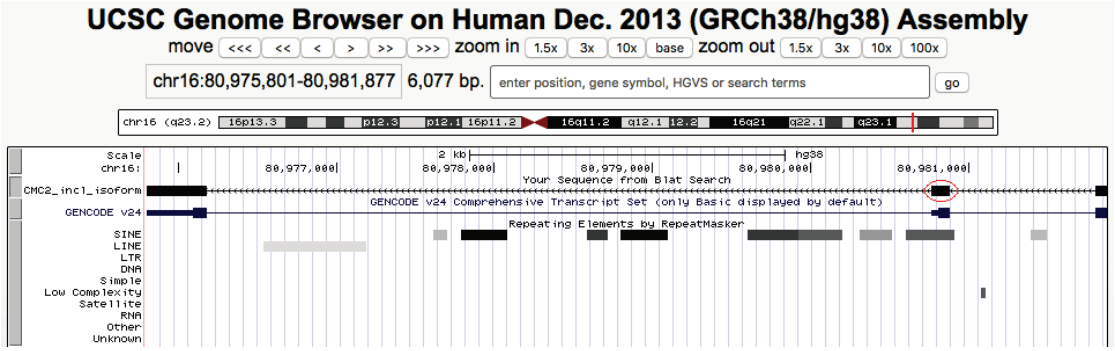
The ULG shares similarities with several approaches that were conceived to add the read information back to the DBG in a reference-free context. The first approaches to do so were based on using multiple values of k to build a DBG, instead of only one [35,34,3,30,24]. A general framework for methods based on this strategy is to build contigs using increasing values of k, and to combine the produced contigs into one final assembly. The inputs to these DBG constructions with different values of k can be the raw reads, the contigs built with $k' < k$ or a mixture of both. As k increases, more read information is integrated into the assembled contigs. More recently, some studies encode the read information directly into the graph, *e.g.* [17,41,48]. The main difference between the ULG and these approaches is that the ULG removes the complex, highly branching parts of the graph, and connects only the well-assembled unitigs through the read information, while the others work on the whole graph. As such, the ULG is less general than the aforementioned approaches, and it is also not an option when its application cannot afford the removal of such complex regions, such as genomic variant calling in population graphs [48]. On the other hand, it can simplify the downstream assembly process, and translate into faster methods, as it is built directly upon a simplified cDBG, and not on the full DBG.

The ULG U is built from a cDBG C. Figure 2 exemplifies this process. The vertices in U are the unitigs that are considered trustful. Trustful unitigs are long-enough unitigs so that we can consider them well assembled. It also means that they have a low branching concentration, as branches in DBGs split linear paths, thus creating smaller unitigs. The algorithmic choice of trusting longer unitigs is in accordance to Lima *et al.* in [27], who show that regions with high branching concentration in DBGs are related to repeats, and processing them can lead to spurious assemblies. A strategy to avoid traversing and assembling such regions is to simply use only the long-enough unitigs. Formally, a unitig u is trustful if u satisfies one of the following conditions: a) u is red and $|u| \geq k + min_{red}$ or b) u is purple or blue and $|u| \geq k + min_{blue}$, where min_{red} is the minimum size of a red unitig to be considered trustful (parameter `minSizeRedUnitigs`, defaulting to 15) and min_{blue} is the minimum size of a purple or blue unitig to be considered trustful (parameter `minSizeBlueUnitigs`, defaulting to 5). In Figure 2(a), we highlight the trustful and non-trustful unitigs in a cDBG.

Removing the non-trustful unitigs and keeping only the trustful ones will substantially disconnect the graph, making any assembly very fragmented. For example, in Figure 2(a), we will have six isolated unitigs if we do so. This means that trustful unitigs are potentially connected through complex, repeat-induced regions, which were removed in the ULG. In order to retrieve back the connections, we map the short reads to the cDBG (using the same procedure described in Section 2.3). An example of such mapping can be seen in Figure 2(b). Given a read r, described by a walk $w(r)$



(a)



(b)

Fig. 1. (a) A subgraph in a de Bruijn graph with neighbourhood 5 around an exonized ALU. The correct assembly is shown as the red path. The complex region is due to the presence of ALUs in transcriptomic data. ALUs are mainly present in introns, but some have been exonised. They are present in high numbers in transcriptomic data because some pre-mRNA is sequenced together with mRNAs. Both exonic and intronic ALUs contribute to the complex structure of the DBG; (b) The UCSC Genome Browser shows that this region corresponds to an exonized ALU (circled in red) in the CMC2 human gene. The flanking exons are also shown for a better visualisation. The data used in this figure corresponds to RNA-seq reads from the MCF7 cell line (with depletion of DDX5 and DDX17) [4].

in the cDBG, for each pair of consecutive trustful unitigs $(u, v) \in w(r)$, we get the substring s in r connecting u to v and add a s-labeled arc $e = (u, v, s)$ to U. For example, if r maps orderly to the trustful unitigs u, v, w and x, we shall build arcs between u and v, v and w, and w and x. The orderly mapping of r to the set of trustful unitigs is denoted by $m(r) = u, v, w, x$. Furthermore, if several different reads map to the same pair of consecutive unitigs u and v, we might have several arcs $e = (u, v, s)$, with different labels s, in U. Constructing these arcs between trustful unitigs based on short-read mapping is exemplified in Figure 2(c).

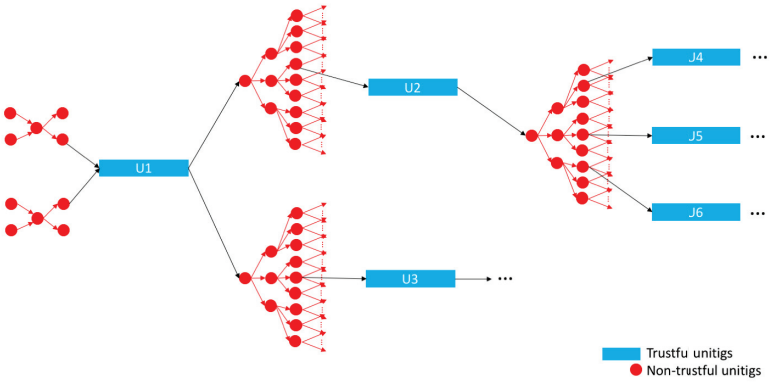
Finally, we still might lose some range information by constructing arcs only between consecutive unitigs, *e.g.* we lost the information given by the read r that u is connected to x passing through v and w. In order to recover part of this information, we store for each trustful unitig u a set $h(u)$ such that $v \in h(u) \leftrightarrow \exists$ a read $r|u \rightarrow ... \rightarrow v \rightarrow ... \in m(r)$. In other words, $h(u)$ contains the set of vertices v such that there is at least one read mapping to u and later to v. The information provided by $h(u)$ can be used as hints and guide the assembly algorithm on how to solve a repeat-induced region using the range information provided by the short reads. Figure 2(c) shows $h(U1)$ for the depicted example. In this figure, we can also see how the hints can help to choose the correct path during assembly in an efficient way. If we start by assembling $U1 \rightarrow U2$, the hints from $U1$ indicate that it is more reliable to continue the assembly towards $U4$ and $U6$. The assembly $a_1 = U1 \rightarrow U2 \rightarrow U5$ should be considered with caution since there are no reads supporting the link from $U1$ to $U5$ (while we have for $U4$ and $U6$). Such assembly could be wrong if, for example, $U1$ and $U2$ stem from a gene G_1 and $U2$ and $U5$ stem from another gene G_2 ($U2$ is a conserved region in an inter-gene repeat present in G_1 and G_2). There is also the possibility of a_1 being a correct assembly, and due to a read coverage or read length problem, we have no reads linking $U1$ to $U5$. Thus, the usefulness of the hints of a unitig depends on the read coverage and read length.

Observe that there is still an arc coupling problem that is not solved by the ULG. For example, in assembly $a_2 = U1 \rightarrow U2 \rightarrow U4$ in Figure 2, we should choose the green arc (with label `ACTTG`) to connect $U1$ and $U2$ for a_2 to be coherent with Read 1. However, empirical observations in RNA-seq data suggest that the labels of different arcs between two trustful unitigs differ only by some SNPs or indels, or represent more complex allelic differences, like tandem satellite repeats with different copy number[2]. In our method, since we are not interested in these variations, we further simplified the ULG by keeping only the most frequent arc between two unitigs.

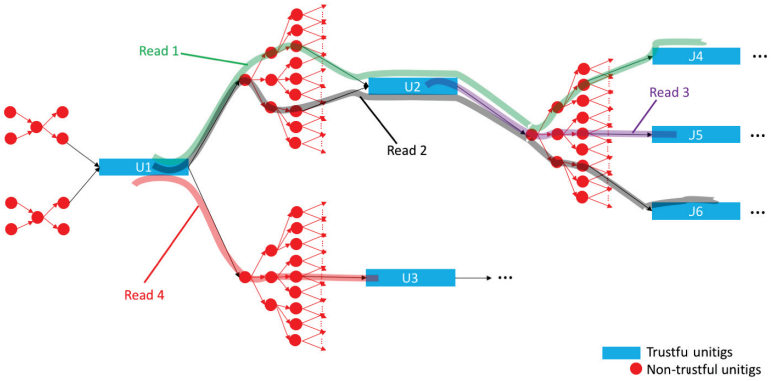
Extensions of the ULG to represent proper read threading, for example, can be done. Each element of the hints set can be a path spelled by a read. This would allow for properly spanning multiple copies of a same repeat, for instance. However, this naive strategy would be heavy in memory. The approach implemented in the Linked de Bruijn Graph (LdBG), described in [48], in which $h(v)$ would store only the paths starting in v can be low in memory and enough to represent the full read information. In fact, the LdBG [48] is more general than the ULG, since it operates on the full DBGs, and has proper read threading. However, the LdBG can be more costly than the ULG, as all the complex regions are kept in the graph[3].

[2] Note to the reviewers: we shall add a section in the Supplementary Material in the full version of the paper with data to backup this claim.

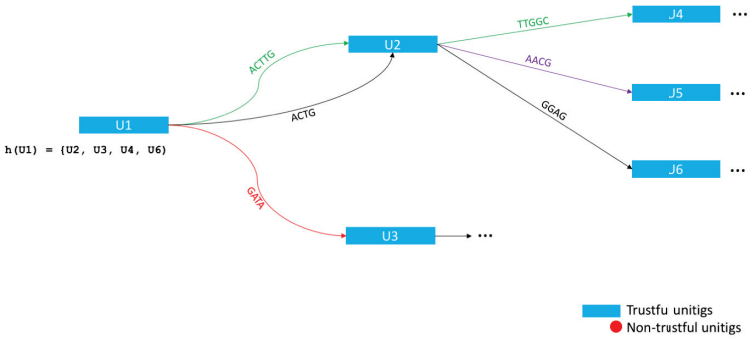
[3] Note to the reviewers: the authors were unaware of [48] during the conception and implementation of the ULG. It might be more proper to describe the ULG as based on, or as a specialization of, the LdBG.



(a)



(b)



(c)

Fig. 2. Building the ULG from a cDBG. (a) The cDBG with the trustful and non-trustful unitigs identified; (b) The mapping of four reads on the cDBG; (c) The construction of the ULG: 1) the non-trustful unitigs are removed; 2) the trustful unitigs are linked by labelled arcs reflecting the read mapping; 3) the hints for each vertex v, $h(v)$, is created. In this example, only $h(U1)$ is shown for simplicity.

2.5 Alternative splicing events enumeration

In order to enumerate the AS events that are present in the short reads and absent in the long reads, we iterate through the mapping of each long read l, and find alternative paths in the ULG flanked by two unitigs stemming from l. This section describes in detail this procedure.

Let C be the cDBG (created in Step 1 - see Section 2.2), l a long read, $w(l)$ a walk in C spelling out t (computed in Step 2 - see Section 2.3), and U the ULG (created in Step 3 - see Section 2.4). We iterate through the vertices $v = w_i \in w(l)$, enumerating all (v, u)-alternative paths in U, which are paths starting in v, following trustful unitigs not belonging to l, and finally reaching a vertex $u = w_j \in w(l) | i < j$, *i.e.*, the index of v in w is smaller than the index of u in w. The main intuition is that v and u compose (part of) the flanking exons of the alternative splicing event we want to find. We require v and u to be (i) trustful and (ii) purple. Clearly, (i) has to be satisfied for u and v to belong to the ULG U. We require (ii) so that the flanking sequences are expressed in both short and long reads. Moreover, to bound our search space, we do not explore paths longer than a given threshold, and we halt the enumeration once we've listed a good amount of AS events between the two flanking vertices. Lastly, we make use of the hints of the ULG to guide our assembly.

In order to understand our algorithm in detail, consider the following definitions. Given a unitig $u \in U$, $c(u)$ denotes the number of k-mers u contains, *i.e.* $c(u) = |u| - k + 1$. Given an arc $e = (u, v, s) \in U$, $c(e)$ denotes the length of the label of e, *i.e.* $c(e) = |s|$. In general, the c function denotes the cost of traversing a given unitig or arc. Given a path $p = x \to y \to ... \to z$ of vertices of U, $c(p)$ is the cost of all its vertices and arcs, and $seq(p)$ denotes the sequence obtained by assembling p. $ED(s_1, s_2)$ denotes the edit (or Levenshtein) distance between two sequences s_1 and s_2. $SG\text{-}ED(s_1, s_2)$ denotes the semi-global edit distance between s_1 and s_2, *i.e.* we compute the edit distance between s_1 and s_2 not penalizing for edit operations in both extremes of s_1 and s_2. $SG\text{-}ED$ is more appropriate than ED to measure the difference rate between two sequences assuming that one of them is a lot longer than the other. Finally, $d(s, t, U)$ denotes the length of the shortest path from s to t in U.

The proposed algorithm is a single-source multi-target path-enumeration algorithm with two constraints. The first is a length constraint ℓ ($\ell = 2000$ by default) on the assembled sequences to bound the search space. Events longer than ℓ will not be found by the algorithm. The second is a biologically-motivated constraint on the splicing complexity to further reduce the search space. During the enumeration algorithm, an alternative path $p = u \to w \to ... \to v$ is defined as a novel splicing event if: (i) there is no other path $p' \in AP(u, v)$ that is very similar to p, where $AP(u, v)$ contains all alternative paths found so far between u and v; (ii) p is not contained in any long read. More specifically, in condition (i) we verify if $ED(seq(p), seq(p'))/max(|seq(p)|, |seq(p')|) >= minED$, for all $p' \in AP(u, v)$, where $minED$ is a minimum edit distance threshold to consider that two paths represent different splicing events ($minED = 0.05$ by default). To efficiently implement condition (ii), we verify if $SG\text{-}ED(seq(p), l) >= minED$, for all long reads l containing at least one of the flanking unitigs u or v. Finally, for each pair of flanking unitigs u and v, we list at most SC ($SC = 10$ by default) splicing events. This constraint is reasonable since we hardly have more than SC alternative transcripts between two constitutive exons. For instance, human genes have on average 6.95 transcript variants per gene, and most genes have at most 10 transcript variants [42]. Moreover, the fact that a gene has many transcript variants does not imply it has complex local AS events: combinations of several local AS events and alternative transcript initiation and termination sites can contribute multiplicatively to the number of transcript variants. We further note that, in the current state of this work, these parameters are still being tuned and are under evaluation.

Furthermore, when exploring the search space of all feasible alternative paths, we make use of the ULG's hints, described in Section 2.4, to drive the assembly towards the sequences most supported by the read information. To do so, when building an alternative path p in our enumeration algorithm, we keep track of how many times each vertex $v \in U$ was included in the hints of each $u \in p$ in a vector HC. In other words, $HC(v) = \sum_{u \in p} |h(u) \cap \{v\}|$. When faced with the choice of which unitigs to follow to extend a path $p = u \rightarrow ... \rightarrow v$, we explore the neighbours w such that $HC(w)$ is the highest. Indeed, this could lead us to miss some lowly covered transcripts, but this conservative algorithmic choice reduces the number of misassemblies. A more permissive strategy can be executed by setting a higher value for SC. We observe that this strategy is similar to the oldest link approach described in the LdBG [48]. However, we are more permissive, since our goal is to enumerate several alternative paths between two vertices, whereas the LdBG focus on finding a long linear path explaining the genome.

Finally, Algorithm 1 describes our alternative path enumeration procedure in detail. For the time analysis of Algorithm 1, consider the following definitions. Let U be a graph, and let n and m be the number of vertices and arcs in U, respectively. U^R is the reverse graph of U, *i.e.* U^R is a copy of U but with the direction of the arcs reversed. Let $ssd(v, U)$ be the Dijkstra's Shortest Path First algorithm [7], that computes the distance from a single source v to all vertices in U in $O(m + n \log n)$ time. In order to simplify this time analysis, we will ignore the time required to output an alternative path once we reach a target (*i.e.* we are not taking into account here the time spent when executing lines 5-19). We will only determine the asymptotic delay of finding alternative paths. Updating and restoring the HC (in lines 22 and 37, respectively) can be done in $O(n)$ time with a count vector indexed by the vertices. We can associate a boolean vector to a path p in order to query the membership of a vertex w in p in $O(1)$ time. Thus the constraint in line 24 can be done in $O(1)$ time. For the constraint in line 25, we can add an artificial vertex t' to U^R such that $N^+_{U^R}(t') = T$ and $c(t', t \in T) = 0$, and precompute $ssd(t', U^R)$ before line 24 in $O(m + n \log n)$ time. By doing so, $\exists t \in T | c(p) + c(v, w) + d(w, t, U) \leq \ell$ can be evaluated in $O(1)$ time, since $min_{t \in T}\{d(w, t, U)\} = d(t', w, U^R)$, which is already precomputed. Thus, lines 24 and 25 run in $O(n)$ total time, since $|N^+_U(v)| = O(n)$, with a $O(m + n \log n)$ preprocessing time. We observe that all vertices $w \in p$ should not be in U^R at the time of the precomputation of $ssd(t', U^R)$. It is not hard to see that the other lines in Algorithm 1 take at most linear time, apart from the recursive call. Since every time we execute Algorithm 1, we add a vertex to the path p and $|p| \leq n$, then the delay to find alternative paths is $O(n * (m + n \log n))$. It is not hard to reduce the problem of listing bounded length (s, t)-paths in directed graphs [37] to our problem of finding alternative paths in the ULG. The delay of the most efficient algorithm for this first problem is also $O(n * (m + n \log n))$ [37], matching our. An improvement in our algorithm would also imply an improvement on the most efficient algorithm for the single-source K-shortest paths in a directed graph [52,37], which dates back from the 1970s. As such, we consider that improving even further the delay of our enumeration algorithm is far from being trivial, and out of the scope of this paper.

3 Preliminary results

In order to check if our method works, we validated it in sample datasets. This also gave us some good test cases to help solve eventual bugs and direct the development. The results described in this section will not be part of the final version of this paper. An implementation of our method can be found in `https://gitlab.inria.fr/lishisoa/EYTA`, but we warn that it is yet in active development, not finished, and unstable.

3.1 Simulated dataset on one gene and five transcripts

We first ran our method on the human gene NEU1, which contains five transcripts. The long reads set was composed only by one of the five transcripts (ENST00000229725.4), and short reads were simulated using ART_ILLUMINA [18] at 50x coverage for each transcript. In this small example, there were four intron retentions, one alternative acceptor and one exon skipping event to be found, which all were. Figure 3 details these results.

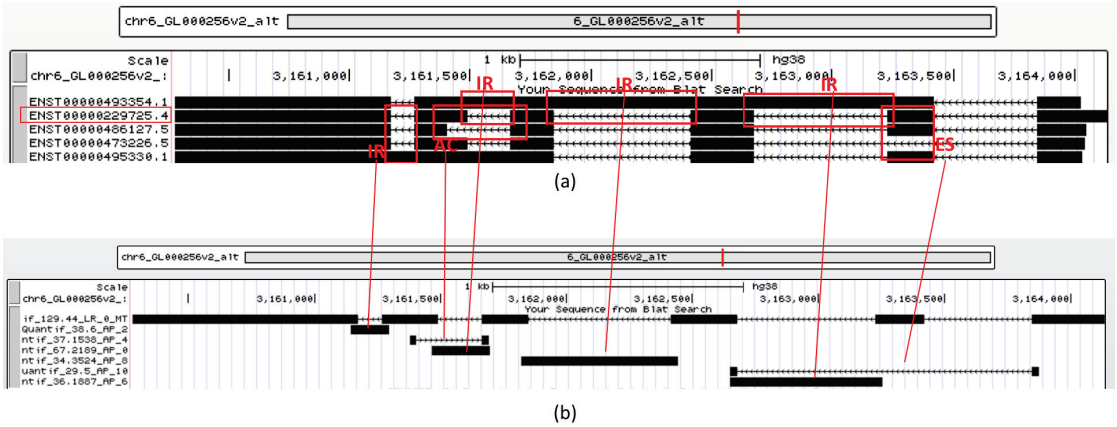


Fig. 3. (a) The five transcripts from the human gene NEU1, aligned to the human reference genome and visualized using the UCSC Genome Browser. The events to be found are in red boxes. IR stands for Intron Retention, AC for Alternative Acceptor, and ES for Exon Skipping. (b) The output of our method - the long read and its alternative paths. All events were found in this very simple example.

3.2 Simulated dataset on the whole human chromosome 1

Our next benchmark comprised a simulated dataset on the whole human chromosome 1. We restricted ourselves to protein-coding, multi-transcript, and non-paralogous genes, obtaining 488 genes. We did not simulate long reads, and we took 10% of the transcripts of each gene as our long reads set. We then simulated 30x coverage 150-bp single-end short reads from all transcripts using wgsim [18], with no error-rate. Our ground truth, built using ASTALAVISTA [11], is composed by all pairwise known AS events in these genes, where only one of the isoforms of the event is in the aforementioned long reads set. We understand that this setting is unrealistic in some aspects. First, real datasets contain paralogous genes, and if they are similar enough (*i.e.* presenting several common regions with more than k bases), we will confuse the expression of two different paralogous genes as alternative transcripts of a single gene. Second, our reads are perfectly accurate. Third, we have a uniform and homogeneous short-read coverage of all transcripts. On the other hand, we are also overestimating the alternative splicing level of each gene by simulating all of its transcripts with short reads. Therefore, this simulated dataset is expected to be more complex in this aspect than real datasets. Nonetheless, this composes a good test dataset to improve our implementation

and have a first performance check of our method. Should this benchmark be included in the final version of the paper, we shall improve on these unrealistic characteristics.

Considering this benchmark, we currently obtain a recall (proportion of ASTALAVISTA AS events found by our method over the total number of ASTALAVISTA AS events) of 99.6% and a precision (proportion of found events corresponding to ASTALAVISTA AS events over the total number of found events) of 88.7%. We plan to improve our method with this benchmark by clarifying the 11.3% false positive events we currently have. We could already verify that many of these false positive events correspond to misassemblies that happen when we have alternative transcript termination (or initiation) sites coupled with AS events as shown in Figure 4. Unfortunately, as short reads are unable to describe the full structure of mRNAs, it is not clear how to avoid such misassemblies for now.

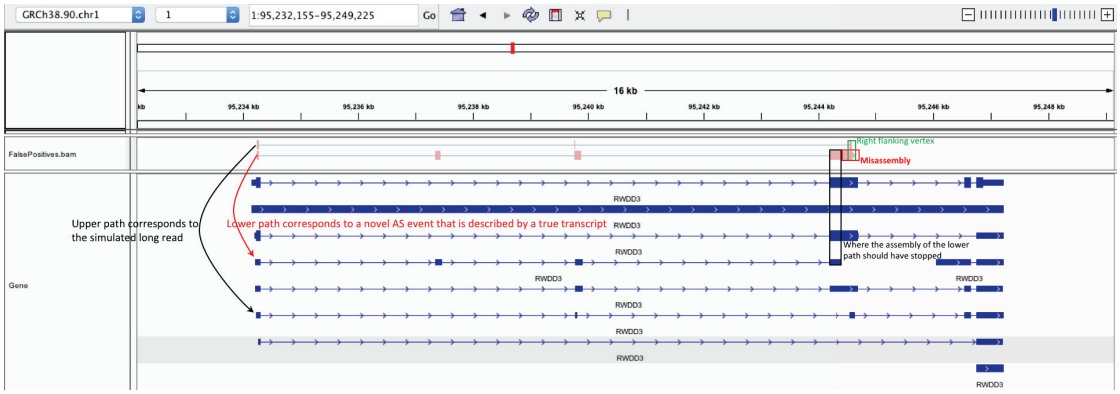


Fig. 4. A false positive event in human gene RWDD3. The misassembly is due to an alternative transcript termination site coupled with AS events. The last exon of the transcript that corresponds to the lower path of the event is part of an internal exon of other transcripts. The event itself is flanked by the first exon of the gene, and this longer exon. By comparing the transcripts that correspond to the upper and lower paths of the event, we can see that we do not have a common right flanking exon that would define the alternative splicing event (the small common region shared between the second exon of the upper path and the third exon of the lower path is shorter than k, and thus does not induce a right flanking vertex in the ULG). Unfortunately, as short reads are unable to describe the full structure of mRNAs, it is not clear to infer that the assembly of the lower path should have stopped earlier. Visualisation done with IGV [40].

4 Perspectives

In this section, we describe our perspectives for this work, which we plan to develop in order to publish it.

4.1 Methodological perspectives

– Implement paired-end read mode in the ULG;

- Compare the ULG approach against a multi-k approach and to the LdBG [48];
- Optionally enumerate AS events described uniquely by the long reads;
- Investigate some enumeration techniques to allow for a faster search of novel AS events. One such idea is attributing a weight to every vertex in the graph, and increasing the weight of the vertices of an alternative path when it is found. We then prioritize lighter paths in the search for novel AS events.

4.2 Benchmarking perspectives

We plan to benchmark our method on samples sequenced with both PacBio Iso-seq and Illumina. We first intend to run it on a human sample, as we can make use of the most complete annotations available to validate the results of our method. We then plan also to run on a sample containing a more comprehensive Iso-seq sequencing. A good option that we have found so far is the data presented in [21], which contains PacBio Iso-seq data from the brain tissue of an adult J-Line chicken, which was also sequenced using Illumina. More importantly, the long reads RNA library was normalized in [21] to reduce over-represented transcripts, which appears to have provided a transcriptome coverage efficiency of more than 5 times that of a previous study [46]. If our method manages to find novel AS events even on normalized Iso-seq data, then its performance will be even better than on non-normalized datasets, which is far more commonly used. As the chicken genome is not as well annotated as the human genome, many events that we find might not have been described previously. We will then validate our predictions by searching for canonical splice sites. Finally, we will compare the performance of our method against IDP-denovo [12] and Trinity [13] in hybrid mode.

Algorithm 1 Alternative path enumeration algorithm

function ENUMERATE_ALTERNATIVE_PATHS(ULG U, source u, set of targets T, current unitig v, current path p, length threshold ℓ, hints counter HC, splicing complexity SC, splicing events $splEvs$, alternative paths APs, long reads set L, edit distance threshold $minED$)
  $p \leftarrow p + v$ //add v to the end of p

  //check if we reached a target
  if $v \in T$ **then**
    //yes. Check if we still did not list SC (u, v)-alternative-splicing events
    if $|splEvs(u, v)| < SC$ **then**
      //no, we can try to list this one
      //check if the assembled alternative path compose a new splicing event, *i.e.* it
      //has a large enough edit distance with all the previously found (u, v)-alternative-
      //paths and it is not contained in a long read
      if $alternativePathIsANovelAS(p, U, APs, L)$ **then**
        //novel AS event found between u and v - output the path and its assembly
        output p and $seq(p)$
        $splEvs(u, v) \leftarrow splEvs(u, v) \bigcup \{p\}$
      end if
      $APs(u, v) \leftarrow APs(u, v) \bigcup \{p\}$ //add p to all alternative paths found between u and v so far
    end if
    $p \leftarrow p - v$ //remove v from the end of p
    return
  end if

  //here we did not reach a target - keep building the alternative path towards a target
  update HC due to the addition of v in p

  //find the set of neighbours N to be explored
  $N \leftarrow \{w \in N_U^+(v) |$ 1) $w \notin p$; //path constraint
  2) $\exists t \in T | c(p) + c(v, w) + d(w, t, U) \leq \ell\}$ //length constraint
  $H_{max} \leftarrow max_{n \in N} HC(n)$ //H_{max} denotes the highest hint
  $N \leftarrow \{n \in N | HC(n) = H_{max}\}$ //we update N to ensure an assembly guided by the highest hints
  if $H_{max} = 1$ **then**
    //vertices added to p before v did not give any hints
    //hardest case in assembly
    //guide the assembly by the SC-longest unitigs
    $N \leftarrow n \in N | n$ is one of the SC-th longest unitigs in N
  end if

  //explore each neighbour recursively
  for $n \in N$ **do**
    ENUMERATE_ALTERNATIVE_PATHS(U, u, T, n, p, ℓ, HC, SC, $splEvs$, APs, $minED$)
  end for

  restore HC to the previous state
  $p \leftarrow p - v$ //remove v from the end of p
end function

References

1. Fatemeh Almodaresi, Hirak Sarkar, Avi Srivastava, and Rob Patro. A space and time-efficient index for the compacted colored de bruijn graph. *Bioinformatics*, 34(13):i169–i177, jun 2018.
2. K. F. Au, V. Sebastiano, P. T. Afshar, J. D. Durruthy, L. Lee, B. A. Williams, H. van Bakel, E. E. Schadt, R. A. Reijo-Pera, J. G. Underwood, and W. H. Wong. Characterization of the human ESC transcriptome by hybrid sequencing. *Proceedings of the National Academy of Sciences*, 110(50):E4821–E4830, nov 2013.
3. Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 19(5):455–477, may 2012.
4. Clara Benoit-Pilven, Camille Marchet, Emilie Chautard, Leandro Lima, Marie-Pierre Lambert, Gustavo Sacomoto, Amandine Rey, Audric Cologne, Sophie Terrone, Louis Dulaurier, Jean-Baptiste Claude, Cyril Bourgeois, Didier Auboeuf, and Vincent Lacroix. Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data. *Scientific Reports*, 8(1), 2018.
5. Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, apr 2016.
6. Brian Bushnell et al. Bbmap: A fast, accurate, splice-aware aligner. *LBNL-7065E. Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA.*, 3 2014.
7. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.
8. A Dobin, C A Davis, F Schlesinger, J Drenkow, C Zaleski, S Jha, P Batut, M Chaisson, and T R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, January 2013.
9. Erwan Drezen, Guillaume Rizk, Rayan Chikhi, Charles Deltel, Claire Lemaitre, Pierre Peterlongo, and Dominique Lavenier. GATB: Genome assembly & analysis tool box. *Bioinformatics*, 30(20):2959–2961, jul 2014.
10. Paolo Ferragina and Giovanni Manzini. An experimental study of an opportunistic index. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 269–278, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.
11. S. Foissac and M. Sammeth. ASTALAVISTA: dynamic and flexible analysis of alternative splicing events in custom gene datasets. *Nucleic Acids Research*, 35(Web Server):W297–W299, may 2007.
12. Shuhua Fu, Yingke Ma, Hui Yao, Zhichao Xu, Shilin Chen, Jingyuan Song, and Kin Fai Au. IDP-denovo: de novo transcriptome assembly and isoform annotation by hybrid sequencing. *Bioinformatics*, 34(13):2168–2176, feb 2018.
13. Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman, and Aviv Regev. Full-length transcriptome assembly from RNA-seq data without a reference genome. *Nature Biotechnology*, 29(7):644–652, may 2011.
14. Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman, and Aviv Regev. Trinity v2.0.2 release (Jan 22, 2015). `https://github.com/trinityrnaseq/trinityrnaseq/releases/tag/v2.0.2`, 2015. [Online; accessed 04-February-2019].
15. Mitchell Guttman, Manuel Garber, Joshua Z Levin, Julie Donaghey, James Robinson, Xian Adiconis, Lin Fan, Magdalena J Koziol, Andreas Gnirke, Chad Nusbaum, John L Rinn, Eric S Lander, and Aviv Regev. Ab initio reconstruction of cell type–specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nature Biotechnology*, 28(5):503–510, may 2010.

16. Brian J Haas and Michael C Zody. Advancing RNA-seq analysis. *Nature Biotechnology*, 28(5):421–423, may 2010.
17. Guillaume Holley, Roland Wittler, Jens Stoye, and Faraz Hach. Dynamic alignment-free and reference-free read compression. *Journal of Computational Biology*, 25(7):825–836, jul 2018.
18. Weichun Huang, Leping Li, Jason R. Myers, and Gabor T. Marth. ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, dec 2011.
19. Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4):357–360, mar 2015.
20. Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14(4):R36, 2013.
21. Richard I. Kuo, Elizabeth Tseng, Lel Eory, Ian R. Paton, Alan L. Archibald, and David W. Burt. Normalized long read RNA sequencing in chicken reveals transcriptome complexity similar to human. *BMC Genomics*, 18(1), apr 2017.
22. Bo Li and Colin N Dewey. RSEM: accurate transcript quantification from RNA-seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.
23. Chenhao Li, Kern Rei Chng, Esther Jia Hui Boey, Amanda Hui Qi Ng, Andreas Wilm, and Niranjan Nagarajan. INC-Seq: accurate single molecule reads using nanopore sequencing. *GigaScience*, 5(1):34, 12 2016.
24. Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, and Tak-Wah Lam. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676, jan 2015.
25. Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, may 2018.
26. Leandro Lima, Camille Marchet, Segolene Caboche, Corinne Da Silva, Benjamin Istace, Jean-Marc Aury, Helene Touzet, and Rayan Chikhi. Comparative assessment of long-read error-correction software applied to rna-sequencing data. *bioRxiv*, 2018.
27. Leandro Lima, Blerina Sinaimeri, Gustavo Sacomoto, Helene Lopez-Maestre, Camille Marchet, Vincent Miele, Marie-France Sagot, and Vincent Lacroix. Playing hide and seek with repeats in local and global de novo transcriptome assembly of short rna-seq reads. *Algorithms Mol Biol*, 12:2–2, Feb 2017.
28. Antoine Limasset, Guillaume Rizk, Rayan Chikhi, and Pierre Peterlongo. Fast and scalable minimal perfect hashing for massive key sets. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2017.
29. Jeffrey A. Martin and Zhong Wang. Next-generation transcriptome assembly. *Nature Reviews Genetics*, 12(10):671–682, sep 2011.
30. Sergey Nurk, Dmitry Meleshko, Anton Korobeynikov, and Pavel A. Pevzner. metaSPAdes: a new versatile metagenomic assembler. *Genome Research*, 27(5):824–834, mar 2017.
31. Qun Pan, Ofer Shai, Leo J Lee, Brendan J Frey, and Benjamin J Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40(12):1413–1415, nov 2008.
32. Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419, mar 2017.
33. Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature Biotechnology*, 32(5):462–464, apr 2014.
34. Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin. Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–i101, jun 2011.
35. Yu Peng, Henry C. M. Leung, S. M. Yiu, and Francis Y. L. Chin. IDBA – a practical iterative de bruijn graph de novo assembler. In *Lecture Notes in Computer Science*, pages 426–440. Springer Berlin Heidelberg, 2010.
36. Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature Biotechnology*, 33(3):290–295, feb 2015.

37. Romeo Rizzi, Gustavo Sacomoto, and Marie-France Sagot. Efficiently listing bounded length st-paths. In *Lecture Notes in Computer Science*, pages 318–329. Springer International Publishing, 2015.
38. Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature Methods*, 10(1):71–73, nov 2012.
39. Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S Butterfield, Richard Newsome, Simon K Chan, Rong She, Richard Varhol, Baljit Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A Moore, Martin Hirst, Marco A Marra, Steven J M Jones, Pamela A Hoodless, and Inanc Birol. De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7(11):909–912, oct 2010.
40. James T Robinson, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S Lander, Gad Getz, and Jill P Mesirov. Integrative genomics viewer. *Nature Biotechnology*, 29(1):24–26, jan 2011.
41. Roye Rozov, Gil Goldshlager, Eran Halperin, and Ron Shamir. Faucet: streaming de novo assembly graph construction. *Bioinformatics*, 34(1):147–154, jul 2017.
42. Jae Yong Ryu, Hyun Uk Kim, and Sang Yup Lee. Human genes with a greater number of transcript variants tend to show biological features of housekeeping and essential genes. *Molecular BioSystems*, 11(10):2798–2807, 2015.
43. G. Sacomoto, J. Kielbassa, R. Chikhi, and R. Uricaru *et al.* KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, 13(Suppl 6):S5, 2012.
44. Marcel H. Schulz, Daniel R. Zerbino, Martin Vingron, and Ewan Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, feb 2012.
45. Mingfu Shao and Carl Kingsford. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature Biotechnology*, 35(12):1167–1169, nov 2017.
46. Sean Thomas, Jason G. Underwood, Elizabeth Tseng, and Alisha K. Holloway and. Long-read sequencing of chicken transcripts and identification of new transcript isoforms. *PLoS ONE*, 9(4):e94650, apr 2014.
47. Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, may 2010.
48. Isaac Turner, Kiran V Garimella, Zamin Iqbal, and Gil McVean. Integrating long-range connectivity information into de bruijn graphs. *Bioinformatics*, 34(15):2556–2565, mar 2018.
49. Eric T. Wang, Rickard Sandberg, Shujun Luo, Irina Khrebtukova, Lu Zhang, Christine Mayr, Stephen F. Kingsmore, Gary P. Schroth, and Christopher B. Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470–476, nov 2008.
50. Thomas D. Wu, Jens Reeder, Michael Lawrence, Gabe Becker, and Matthew J. Brauer. GMAP and GSNAP for genomic sequence alignment: Enhancements to speed, accuracy, and functionality. In *Methods in Molecular Biology*, pages 283–334. Springer New York, 2016.
51. Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, and J. Wang. SOAPdenovo-trans: de novo transcriptome assembly with short RNA-seq reads. *Bioinformatics*, 30(12):1660–1666, feb 2014.
52. Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

Chapter 9

Conclusions and Perspectives

In this thesis, we developed, improved and evaluated methods to process massively sequenced data, mainly short and long RNA-sequencing reads, to help the community answer biological questions such as: which alternative splicing events can be identified from an mRNA sample studied through RNA-seq, and which genotypes might provide antibiotic resistance in a set of strains.

The main thread we followed was developing and improving methods to process sequenced data using de Bruijn graphs to contribute to the alternative splicing literature (Chapters 3 and 8, papers [75, 77]). We tackled this general problem also from a theoretical (Chapter 5, papers [1, 2]) and an analytical (Chapter 4, paper [11]) perspective. Collaborations (Chapter 6, paper [48], and Chapter 7, paper [76]) deviated us from this main thread, but we always had at least one main aspect in common with our original proposal.

In the next subsections, we describe a series of technical and personal perspectives resulting from this PhD. The technical perspectives list several works that are currently in development, but that were not yet ready to be featured in the previous chapters, or future directions that can be followed. The personal perspectives describe my general view related to the scientific context where this thesis is placed, and a more general opinion about bioinformatics.

9.1 Technical Perspectives

9.1.1 KisSplice

Enumeration algorithm

We explored another approach to enumerate $(s, *, \alpha_1, \alpha_2, b)$-bubbles in a DBG G, *i.e.* bubbles with source s and any target such that the paths p_1 and p_2 of the bubbles satisfy $|p_1| \leq \alpha_1$, $|p_2| \leq \alpha_2$, and the number of branching vertices in both p_1 and p_2 is at most b. In [77], we describe an algorithm to do so with delay $O(b^2|A(G)|)$, hereby called $AlgKS$. Exploring the following facts – i) DBGs built from DNA or RNA are naturally sparse, due to having a maximum in- and out-degree 4; ii) the current KisSplice algorithm

	b=5	b=10	b=15	b=20	b=25
AllPaths	13.9	14	15.4	33.8	167.2
AlgKS	22.8	T/o	T/o	T/o	T/o

Table 9.1: Results of the runtime of both the *AllPaths* and *AlgKS* algorithms on a human dataset, with increasing values of b. The runtimes presented are in minutes, T/o stands for Timeout, when an execution took more than 300 minutes.

normally takes a huge amount of time when b increases, and therefore b is usually a small constant (5, by default); iii) some algorithms can afford to use more memory space in exchange of less running time − we developed an algorithm, hereby called *AllPaths*, that finds all paths originating from a source vertex s satisfying the length and branching constraints, and then proceeds to list all pairs of internally disjoint paths having a same target t as bubbles. Although this is one of the naivest approaches to enumerate bubbles in a DBG, it is surprisingly fast in practice. Theoretically, *AllPaths* takes a total time of $O(P^2 * b)$ to enumerate all bubbles originating from a source vertex s, where P is the number of paths. This algorithm is theoretically better than *AlgKS* if $B = \Theta(P^2)$, but theoretically worse if $B = \mathcal{O}(P)$, where B is the number of output bubbles. Although in general $B = \mathcal{O}(P)$, empirical tests on a human dataset showed that *AllPaths* was several times faster than *AlgKS* (see Table 9.1).

We then compared *AllPaths* and *AlgKS* on a larger dataset, containing 200M reads from the Geuvadis project [61]. Using $k = 25$ and $b = 5$, *AllPaths* took 9.2 hours while *AlgKS* took 291.5 hours[1]. These results convinced us that it would be worth to add *AllPaths* to the KisSplice implementation, and it was made available on KisSplice v2.4.0. Since then, it has been employed in several papers, such as [11, 26, 79]. Further exploration in search of theoretical justifications on why *AllPaths* is more efficient than *AlgKS* in some instances is left as an open problem.

DBG construction improvement

Currently, KisSplice builds the DBG from the raw RNA-seq reads using the Minia [20] version 1 graph building procedure. We changed the graph building algorithm to GATB [30], which has shown to be more efficient in preliminary tests, but additional work should still be done to attest the gain in efficiency and to ensure the correctness of the implementation. This improvement will be featured in the next release.

Stranded Kissreads

Kissreads is the read coherency and quantification step of KisSplice and DiscoSNP [128]. It is responsible to filter out read uncoherent bubbles from the output, and to provide an

[1] We note that these executions were done in a cluster, so the specific machines where they were run could differ.

estimated quantification of each path of the bubble. In a project in collaboration with the group of Nadia Naffakh at Institut Pasteur, where stranded RNA-seq reads were available, many bubbles were identified in which each path was composed of genes located on opposite strands. This corresponds to false-positive bubbles. Such false positives had not been identified on previous datasets because we had never analysed data with overlapping genes. In principle, this is very rare in human, but not in this dataset, where the cells are infected by a virus which perturbs the termination of transcription. This dataset was a training set for our new version of Kissreads, which is now able to process stranded RNA-seq data. Some additional work is still needed to ensure the correctness of the implementation. This improvement will be featured in the next release, and should also be relevant for users working on species with dense genomes where many genes overlap.

9.1.2 Bubble generator

In Chapter 5, papers [1,2], we described the bubble generator, a compact representation of all bubbles in a directed graph. Unfortunately, the results obtained on enumerating alternative splicing events through the bubble generator did not yield results better than existing algorithms, although part of the bubbles in the generator were indeed interesting, corresponding to real events that would be hard to find with the KisSplice algorithm. We decided to not add this feature to the KisSplice implementation, since we do not think it is ready for production usage. In order to do so, some interesting directions would be to reduce the number of false positive alternative splicing events by adding more biologically-motivated constraints to the bubbles in the generator. Another direction would be to properly enumerate alternative splicing events by combining the generator bubbles appropriately. We warn, however, that this application does not seem trivial to develop. Indeed, the main application of cycle basis is preprocessing a graph to remove vertices and edges that do not belong to any cycles, in order to reduce the work of a post-processing algorithm, possibly a cycle enumerator. In the case of the bubble generator, we might have the same situation. Another interesting usage would also be to decompose complex events to bubbles in the generator, in which the user would like to know how alternative splicing events are combined to form a more complex one.

9.1.3 DBGWAS

For DBGWAS, the main perspectives are:

- Accept continuous phenotypes (almost finished on branch `https://gitlab.com/leoisl/dbgwas/tree/0.5.4`) and genotypes;
- Automated subgraph labeling, which is a feature to automatically predict DBGWAS subgraph labels into local polymorphisms or mobile genetic element (described in detail in [24], and almost finished on branch `https://gitlab.com/leoisl/dbgwas/tree/automatic_labelling`). This was mainly done by Magali Dancette and Laurent Jacob;

- Enable the user to input read datasets instead of assemblies;
- Enable the user to use customized statistical test in step 2;
- Scale to metagenomics datasets.

9.1.4 Mapping of high-error-rate long RNA-seq reads to DBGs built from short RNA-seq reads

The method presented in Chapter 8, paper [75], to enumerate alternative splicing events in the presence of both short and long reads requires accurate long reads, which can restrict its application. In order to generalize to long reads datasets containing a high error rate, a mapping of such erroneous long reads to de Bruijn or node-labeled graphs built from short reads is required. This problem is currently being addressed in the literature. Some works describe heuristics for this problem, *e.g.* LoRDEC [110], and hybridSPAdes [5], while others tackle it through a Partial Order Alignment [63] approach, with appropriate handling of cycles, such as in [55,88]. Additional works describe exact optimal algorithms [50,101], but that do not seem to scale to the size of real data graphs. In a work in progress with Said Sadique Adi (Associate Professor at Universidade Federal de Mato Grosso do Sul) and members of the team, we are currently addressing this problem by: 1) building a DBG G from the short reads; 2) mapping long reads to G naively; 3) improving the badly mapped regions of the long reads.

9.1.5 The β value for flagging repeats in RNA-seq data

We plan to extend the Branching Measure introduced in Chapter 3, paper [77], with the β value. The intuition is to better capture complex regions induced by repetitions in RNA-seq data. Let us say that we have a high-copy-number and low-divergence repeat R. Each copy of R has a high probability of being inserted in a different context (the flanking bases of each copy) in the transcriptome. As such, if we build the DBG from the RNA-seq data, we will compact all copies of R into a small subgraph, due to their low divergence. The number of different contexts flanking R would be associated with its copy number. Figure 9.1 exemplifies this process. To capture the repeats as depicted in Figure 9.1, we propose a new measure, the β value. The β value of a vertex v bounded by a distance d is $\beta_d(v) = \max_{0 \leq i \leq d} |\mathcal{W}_i(v)|$, where $\mathcal{W}_i(v)$ is the set of vertices u such that there exists a walk from v to u containing exactly i arcs. Intuitively, the repeats we want to capture will contain a high β value. More specifically, the vertices located at the borders of the repeat copies would have a very high β value, providing a hint to the assembler that it traverses, or is about to traverse, a hard-to-assemble region. It is also important to note that we should keep d to a small value, as we want to capture the complexity of the local flanking context of a repeat. A high value of d can have the undesirable side-effect of many vertices being classified as bordering repeats, even though they could be relatively distant. We also note that we compute the in-β value and the out-β value, which are the β values of a vertex considering the incoming and the outgoing arcs, respectively.

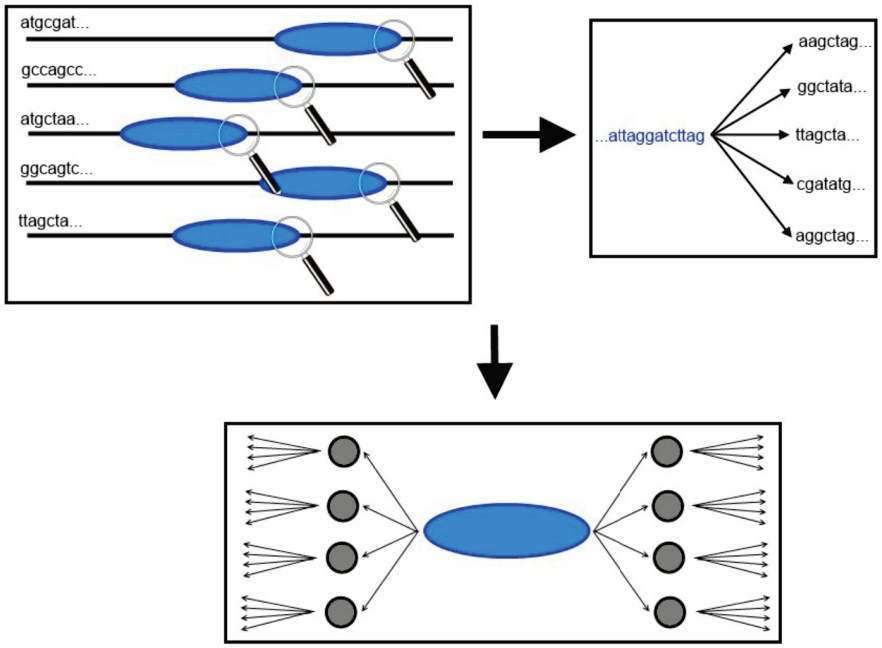


Figure 9.1: Top left: the different copies of a repeat inserted each in a different context. For simplicity, here we are assuming that all these different repeat copies are all equal (no divergence). Top right: all members of the repetition are compacted into a single sequence and their different (right) flanking contexts. Bottom: the structure that such repeat would create in a DBG. The subgraph in blue corresponds to the several copies of the repeat compacted, and the highly branching regions around it correspond to the several different flanking contexts.

As a proof-of-concept, Figures 9.2 and 9.3 show chimeric and truncated transcripts assembled by Trinity, respectively, in a real dataset. By looking at the structure of the graph, we can see that the β value would be able to flag the repeat-associated nodes in these transcripts.

This is only a proof-of-concept of the β value, which was presented by *L.* in the SeqBio 2015 Workshop (`http://www.gdr-bim.cnrs.fr/seqbio2015`). We retook this work in 2017, and it is currently implemented in a module which has the objective of identifying repeat-associated vertices of a DBG, and remove them from the graph. This module was developed by *L.*, and is part of a tool currently in development by Camille Sessegolo in her PhD, supervised by Vincent Lacroix and Arnaud Mary, which has the objective of identifying complex alternative splicing events (discussed in detail in the next subsection).

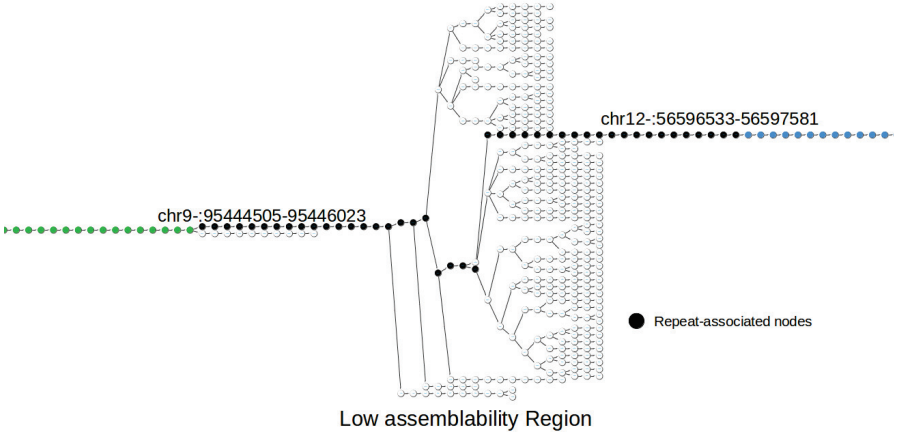


Figure 9.2: A chimeric transcript assembled by Trinity. The green nodes map to chr9, the blue nodes map to chr12, and the black nodes are associated to a repeat (an Alu). The β value of the nodes in the borders of the repeat is high.

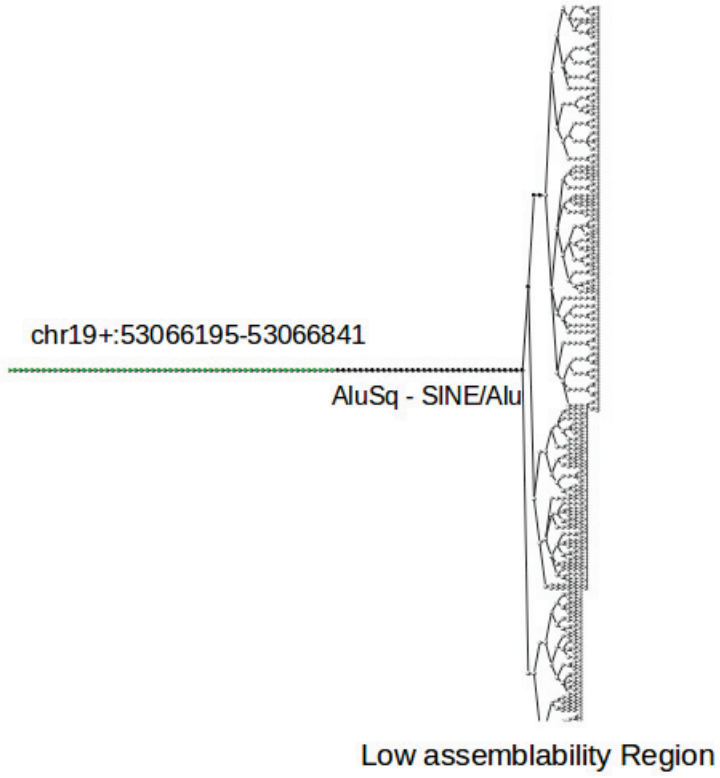


Figure 9.3: A truncated transcript assembled by Trinity. The green nodes map to chr19, and the black nodes are associated to a repeat (an Alu). The β value of the nodes in the border of the repeat is high.

9.1.6 Complex alternative splicing events enumeration

Sammeth in [112] states: i) the true splicing diversity of alternative splicing (AS) events often comprises more than two alternatives and therefore cannot be sufficiently described by pairwise comparisons (*i.e.* pairwise bubbles); ii) the majority of the exons that are observed as mutually exclusive in pairwise comparisons in fact involves at least one other alternative splice form that disagrees with their mutual exclusion; iii) similar observations as in (ii) also hold for the alternative skipping of two subsequent exons; iv) a systematic analysis of complete AS events at a large scale provides subtle insights in the mechanisms that drive (alternative) splicing. A complex or complete AS event can be described as all AS variations contained between two constitutive parts of all transcripts of a gene. As a concrete example, Figure 9.4 shows an example of a complex AS event that, if described pairwisely, will contain redundancy and incompleteness. Moreover, inferring if a pairwise AS event is differentially expressed based on their quantification can lead to incorrect conclusions if it is, in fact, a complex AS event, mainly if the pairwise event describes a variation between two non-major isoforms. Figure 9.5 shows this scenario.

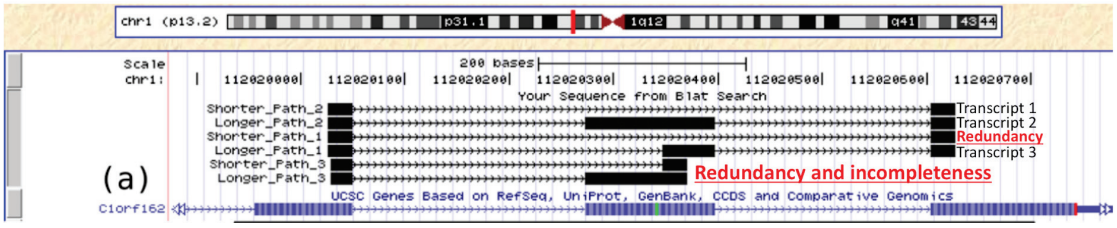


Figure 9.4: A small example showing that describing complex AS events through pairwise bubbles can lead to redundancy and incompleteness. Here, we have three transcripts: transcript 1 is the exclusion of the middle exon, transcript 2 describes its inclusion, and transcript 3 describes an alternative acceptor site. If analysed pairwisely, we will have three bubbles, where half of the paths described by these bubbles will be redundant (*i.e.* already described previously), and one bubble will be incomplete (*i.e.* their paths are subpaths of bubbles already described previously).

These observations led us to explore the identification of AS events using complex, instead of pairwise, bubbles. We thus describe in the following our definition of a complex bubble. A (s,t)-multiwise-bubble MW in a directed graph G is a subgraph of G where: i) MW is the subgraph induced by $\mathcal{AP}$, where $\mathcal{AP}$ is the set of all (s,t)-paths in G; ii) s is the only source of MW (*i.e.* only s can reach all vertices $v \in MW$); iii) t is the only target of MW (*i.e.* only t can be reached by all vertices $v \in MW$); iv) s and t are the only vertices common to all $p \in \mathcal{AP}$. A (s,t)-complex-bubble CB is a maximal multiwise-bubble. It is worth noting that we have at most $n*(n-1)$ multiwise-bubbles in a graph G, where n is the number of vertices in G, and thus at most $n*(n-1)$ complex bubbles. On the other hand, we could have $\Omega(2^n)$ pairwise bubbles in G, in the worst case. It is intuitive to verify that many pairwise bubbles can be included in a single complex bubble, but an interesting difference between both models is that if our

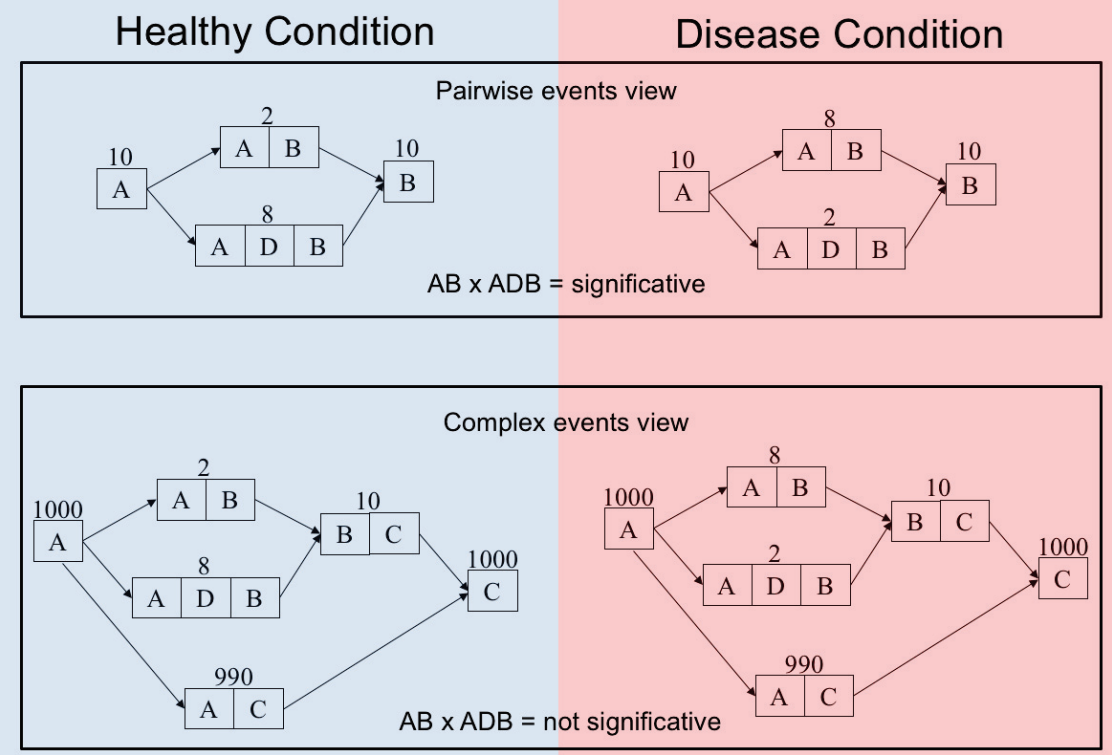


Figure 9.5: Inferring if a pairwise AS event is significant based on their quantification can lead to incorrect conclusions if it is, in fact, a complex AS event, mainly if the pairwise event describes a variation between two non-major isoforms. Here, we compare an AS event in two different conditions (healthy and disease). In the top, we have a pairwise AS event comparison, where exon D is included in 80% of the transcripts going through this event in the healthy condition, while it is excluded in 80% in the disease condition. This would give us clue that this change is significant. However, the bottom panel show us that this event is included in a more complex event, in which the major isoform excludes both exons B and D. With this larger context, we can verify that the magnitude of the change described in the top panel is very small and probably not significant.

object of study are complex bubbles, then we have a quadratic number of objects (on n) to be enumerated and analysed. Therefore, we can work on developing algorithms where the total execution time is polynomial on the size of the input, since the size of the output is now polynomial. Although we have "only" a quadratic number of objects to output, outputting a single (s,t)-complex-bubble CB can still take exponential time, if this output procedure consists in listing all paths between s and t in CB. However, we expect that linear sequences are hardly the best descriptors for such complex events, and we would like to describe them as graph objects instead, in the same spirit of [112], but in a *de-novo* and in a DBG context. We can now define the Enumerating Complex Bubbles Problem:

Enumerating Complex Bubbles Problem (ECBP)
Input: A directed graph $G = (V, A)$.
Output: All (s, t)-complex-bubbles.

Unfortunately, finding an (s, t)-complex-bubble CB in polynomial time is not easy. It can be reduced from the following NP-complete problem (see Proposition 9.2.1 (P5) in Bang-Jensen and Gutin [7]): Given three distinct vertices x,y,z in a directed graph G, decide whether G has an (x, z)-path which also contains the vertex y. In some cases, when the set of all (s, t)-walks coincides with the set of all (s, t)-paths, then the (s, t)-complex-bubble can be found in polynomial time.

Our current algorithm to tackle the ECBP has three steps:

1. Find all pairs of vertices (s, t) having a multiwise bubble;
2. Find which multiwise bubbles are complex bubbles;
3. Output each complex bubble.

Step 1 is implemented using the following proposition (proof is omitted):

Proposition 1. Given a directed graph G and two distinct vertices s and t, there exists an (s, t)-multiwise-bubble if and only if the immediate dominator of t in $\mathcal{F}_s$ is s, where $\mathcal{F}_s$ is the flow graph with root s.

Note that there is the degenerate case where if t is an out-neighbour of s, and there is only one path from s to t, then the immediate dominator of t in $\mathcal{F}_s$ is s, but s and t do not have a multiwise bubble.

Step 2 is implemented observing that an (s, t)-multiwise-bubble is an (s, t)-complex-bubble if there is no (s', t')-multiwise-bubble such that there is a path from s' to s and from t to t'.

Finally, Step 3 is implemented by outputting the subgraph induced by the nodes v than can be reached from s and that can reach t.

Given an (s, t)-complex-bubble CB, if the set of all (s, t)-walks coincides with the set of all (s, t)-paths, then the described algorithm works. Otherwise, then some of the vertices in CB might not belong to any (s, t)-path, but only to (s, t)-walks. We are currently tackling this issue by ideas of changing the definition of the complex bubbles to accept walks instead of paths, or to transform the input directed graph into an acyclic version.

It is also worth mentioning that the ECBP is related to the problem of identifying superbubbles [16, 91, 119]. However, complex bubbles do not necessarily present the matching, acyclicity and minimality properties of superbubbles. As such, it is not possible to use the algorithms designed to find superbubbles to find complex bubbles.

Unfortunately, our described model did not work on real RNA-seq data due to repeats. Repeats induce many cycles in the DBG, one interleaved in another, and as such our model needs to be improved to deal with such cases. We are improving upon this first work in a collaboration with Camille Sessegolo, a PhD student of the team supervised by Vincent Lacroix and Arnaud Mary, who took the main lead in this work.

9.2 Personal Perspectives

Regarding the post-PhD period, my general view related to the scientific context where this thesis is placed is that, unless a specific problem requires short reads, the scientific community will focus its efforts on processing long reads to tackle problems. I believe that long reads will eventually take over short reads, and that transcriptome assembly might not even be required anymore, and genome assembly will be simplified with longer and more accurate reads. However, it will take some years for long reads technologies to reach this state, so in the short term (5-10 years, or maybe longer), I believe that the methods that are able to efficiently utilize both short and long reads will define the state-of-the-art. Regarding alternative splicing, and transcriptomic variations in general, being able to completely describe the structure of a transcript in a long read is very valuable, since exons can be phased perfectly. Precise identification of isoforms, exon boundaries, open reading frames might still be challenging with the high error rate, but protocols like PacBio Isoseq already addresses this natively, creating very accurate Reads of Insert. For sure, both the Nanopore and PacBio error rate and cost will decrease, and throughput will increase in the next years, thus allowing for such applications. The Nanopore RNA direct protocol will also enable the study of transcriptomic variations without the biases and artifacts created by the cDNA synthesis step, allowing for a better understanding of the real transcriptomic variation in a cell.

Another possible scenario is Illumina sequencing remaining very competitive after several years, mainly by reducing per-base cost and increasing the throughput, justifying its use even when a biological question can be tackled by sequencing only long reads. This will imply in even larger datasets than those we have nowadays, and the demand for efficient hybrid methods that make use of succinct data structures will increase, with a large fraction probably being based on de Bruijn graphs or variants.

A more general opinion about bioinformatics after finishing this thesis is that this field requires diverse competences that are not easy to master, ranging from mathematics and theoretical computer science, to wet lab and biology. Mathematical modeling, critical analyses, efficient software implementation, and correct wet lab manipulation are not easy tasks, and I believe that effective collaborations are key to produce valuable works and to learn new concepts.

Bibliography

[1] Vicente Acuna, Roberto Grossi, Giuseppe F Italiano, Leandro Lima, Romeo Rizzi, Gustavo Sacomoto, Marie-France Sagot, and Blerina Sinaimeri. On Bubble Generators in Directed Graphs. In *WG 2017 - 43rd International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 10520 of *Lecture Notes in Computer Science*, pages 18–31, Eindhoven, Netherlands, June 2017. Springer.

[2] Vicente Acuna, Roberto Grossi, Giuseppe F Italiano, Leandro Lima, Romeo Rizzi, Gustavo Sacomoto, Marie-France Sagot, and Blerina Sinaimeri. On Bubble Generators in Directed Graphs. *Submitted to Algorithmica*, 2019.

[3] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, 5 edition, November 2007.

[4] S. Anders, A. Reyes, and W. Huber. Detecting differential usage of exons from RNA-seq data. *Genome Research*, 22(10):2008–2017, jun 2012.

[5] Dmitry Antipov, Anton Korobeynikov, Jeffrey S. McLean, and Pavel A. Pevzner. hybridSPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics*, 32(7):1009–1015, nov 2015.

[6] Kin Fai Au, Jason G. Underwood, Lawrence Lee, and Wing Hung Wong. Improving PacBio long read accuracy by short read alignment. *PLoS ONE*, 7(10):e46679, oct 2012.

[7] Jorgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

[8] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 19(5):455–477, may 2012.

[9] Ergude Bao and Lingxiao Lan. HALC: High throughput algorithm for long read error correction. *BMC Bioinformatics*, 18(1), apr 2017.

[10] Timo Beller and Enno Ohlebusch. A representation of a compressed de bruijn graph for pan-genome analysis that enables search. *Algorithms for Molecular Biology*, 11(1), jul 2016.

[11] Clara Benoit-Pilven, Camille Marchet, Emilie Chautard, Leandro Lima, Marie-Pierre Lambert, Gustavo Sacomoto, Amandine Rey, Audric Cologne, Sophie Terrone, Louis Dulaurier, Jean-Baptiste Claude, Cyril Bourgeois, Didier Auboeuf, and Vincent Lacroix. Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from RNAseq data. *Scientific Reports*, 8(1), 2018.

[12] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology*, 33(6):623–630, may 2015.

[13] Pacific Biosciences. SMRT SEQUENCING: READ LENGTHS. `https://www.pacb.com/smrt-science/smrt-sequencing/read-lengths/`, 2018. [Online; accessed 29-December-2018].

[14] Etienne Birmelé, Pierluigi Crescenzi, Rui Ferreira, Roberto Grossi, Vincent Lacroix, Andrea Marino, Nadia Pisanti, Gustavo Sacomoto, and Marie-France Sagot. Efficient bubble enumeration in directed graphs. In *String Processing and Information Retrieval*, pages 118–129. Springer Berlin Heidelberg, 2012.

[15] Benjamin J. Blencowe. Alternative splicing: New insights from global analyses. *Cell*, 126(1):37 – 47, 2006.

[16] Ljiljana Brankovic, Costas S. Iliopoulos, Ritu Kundu, Manal Mohamed, Solon P. Pissis, and Fatima Vayani. Linear-time superbubble identification algorithm for genome assembly. *Theoretical Computer Science*, 609:374–383, jan 2016.

[17] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, apr 2016.

[18] Brian Bushnell et al. Bbmap: A fast, accurate, splice-aware aligner. *LBNL-7065E. Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA.*, 3 2014.

[19] Yesesri Cherukuri and Sarath Chandra Janga. Benchmarking of de novo assembly algorithms for nanopore data reveals optimal performance of OLC approaches. *BMC Genomics*, 17(S7), aug 2016.

[20] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. In *WABI*, volume 7534 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2012.

[21] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. *Algorithms for Molecular Biology*, 8(1):22, 2013.

[22] Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, Stephen W Turner, and Jonas Korlach. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature Methods*, 10(6):563–569, 6 2013.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Science / Engineering / Math, 3ª edition, 2009.

[24] Magali Jaillard Dancette. *Fine mapping of antibiotic resistance determinants*. PhD thesis, Université Claude Bernard Lyon 1, 2018.

[25] Daryanaz Dargahi, Richard D. Swayze, Leanna Yee, Peter J. Bergqvist, Bradley J. Hedberg, Alireza Heravi-Moussavi, Edie M. Dullaghan, Ryan Dercho, Jianghong An, John S. Babcook, and Steven J.M. Jones. A pan-cancer analysis of alternative splicing events reveals novel tumor-associated splice variants of matriptase. *Cancer Informatics*, 13:CIN.S19435, jan 2014.

[26] Nadia M. Davidson, Anthony D. K. Hawkins, and Alicia Oshlack. SuperTranscripts: a data driven reference for analysis and visualisation of transcriptomes. *Genome Biology*, 18(1), aug 2017.

[27] James J. Davis, Sébastien Boisvert, Thomas Brettin, Ronald W. Kenyon, Chunhong Mao, Robert Olson, Ross Overbeek, John Santerre, Maulik Shukla, Alice R. Wattam, Rebecca Will, Fangfang Xia, and Rick Stevens. Antimicrobial resistance prediction in PATRIC and RAST. *Scientific reports*, 6:27930, 2016.

[28] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[29] A Dobin, C A Davis, F Schlesinger, J Drenkow, C Zaleski, S Jha, P Batut, M Chaisson, and T R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, January 2013.

[30] Erwan Drezen, Guillaume Rizk, Rayan Chikhi, Charles Deltel, Claire Lemaitre, Pierre Peterlongo, and Dominique Lavenier. GATB: Genome assembly & analysis tool box. *Bioinformatics*, 30(20):2959–2961, jul 2014.

[31] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, Arkadiusz Bibillo, Keith Bjornson, Bidhan Chaudhuri, Frederick Christians, Ronald Cicero, Sonya Clark, Ravindra Dalal, Alex deWinter, John Dixon, Mathieu Foquet, Alfred Gaertner, Paul Hardenbol, Cheryl Heiner, Kevin Hester, David Holden, Gregory Kearns,

Xiangxu Kong, Ronald Kuse, Yves Lacroix, Steven Lin, Paul Lundquist, Congcong Ma, Patrick Marks, Mark Maxham, Devon Murphy, Insil Park, Thang Pham, Michael Phillips, Joy Roy, Robert Sebra, Gene Shen, Jon Sorenson, Austin Tomaney, Kevin Travers, Mark Trulson, John Vieceli, Jeffrey Wegener, Dawn Wu, Alicia Yang, Denis Zaccarin, Peter Zhao, Frank Zhong, Jonas Korlach, and Stephen Turner. Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[32] Paul Flicek and Ewan Birney. Sense from sequence reads: methods for alignment and assembly. *Nature Methods*, 6:S6 EP –, Oct 2009. Review Article.

[33] Fernande Freyermuth, Frédérique Rau, Yosuke Kokunai, Thomas Linke, Chantal Sellier, Masayuki Nakamori, Yoshihiro Kino, Ludovic Arandel, Arnaud Jollet, Christelle Thibault, Muriel Philipps, Serge Vicaire, Bernard Jost, Bjarne Udd, John W. Day, Denis Duboc, Karim Wahbi, Tsuyoshi Matsumura, Harutoshi Fujimura, Hideki Mochizuki, François Deryckere, Takashi Kimura, Nobuyuki Nukina, Shoichi Ishiura, Vincent Lacroix, Amandine Campan-Fournier, Vincent Navratil, Emilie Chautard, Didier Auboeuf, Minoru Horie, Keiji Imoto, Kuang-Yung Lee, Maurice S. Swanson, Adolfo Lopez de Munain, Shin Inada, Hideki Itoh, Kazuo Nakazawa, Takashi Ashihara, Eric Wang, Thomas Zimmer, Denis Furling, Masanori P. Takahashi, and Nicolas Charlet-Berguerand. Splicing misregulation of SCN5a contributes to cardiac-conduction delay and heart arrhythmia in myotonic dystrophy. *Nature Communications*, 7(1), apr 2016.

[34] Komei Fukuda, Thomas M. Liebling, and François Margot. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry*, 8(1):1–12, jun 1997.

[35] Daniel R Garalde, Elizabeth A Snell, Daniel Jachimowicz, Botond Sipos, Joseph H Lloyd, Mark Bruce, Nadia Pantic, Tigist Admassu, Phillip James, Anthony Warland, Michael Jordan, Jonah Ciccone, Sabrina Serra, Jemma Keenan, Samuel Martin, Luke McNeill, E Jayne Wallace, Lakmal Jayasinghe, Chris Wright, Javier Blasco, Stephen Young, Denise Brocklebank, Sissel Juul, James Clarke, Andrew J Heron, and Daniel J Turner. Highly parallel direct RNA sequencing on an array of nanopores. *Nature Methods*, 15(3):201–206, jan 2018.

[36] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[37] S Goodwin, J D McPherson, and W R McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet*, 17(6):333–351, 05 2016.

[38] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind,

Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman, and Aviv Regev. Full-length transcriptome assembly from RNA-seq data without a reference genome. *Nature Biotechnology*, 29(7):644–652, may 2011.

[39] Jonathan L. Gross and Jay Yellen. *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2005.

[40] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[41] Mitchell Guttman, Manuel Garber, Joshua Z Levin, Julie Donaghey, James Robinson, Xian Adiconis, Lin Fan, Magdalena J Koziol, Andreas Gnirke, Chad Nusbaum, John L Rinn, Eric S Lander, and Aviv Regev. Ab initio reconstruction of cell type–specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nature Biotechnology*, 28(5):503–510, may 2010.

[42] Brian J Haas and Michael C Zody. Advancing RNA-seq analysis. *Nature Biotechnology*, 28(5):421–423, may 2010.

[43] Thomas Hackl, Rainer Hedrich, Jörg Schultz, and Frank Förster. proovread : large-scale high-accuracy PacBio correction through iterative short read consensus. *Bioinformatics*, 30(21):3004–3011, 11 2014.

[44] D. Hernandez, P. Francois, L. Farinelli, M. Osteras, and J. Schrenzel. De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer. *Genome Research*, 18(5):802–809, feb 2008.

[45] Guillaume Holley, Roland Wittler, Jens Stoye, and Faraz Hach. Dynamic alignment-free and reference-free read compression. *Journal of Computational Biology*, 25(7):825–836, jul 2018.

[46] X. Huang and A. Madan. CAP3: A DNA sequence assembly program. *Genome research*, 9(9):868–877, September 1999.

[47] Illumina. Illumina sequencing platforms. `https://www.illumina.com/systems/sequencing-platforms.html`, 2018. [Online; accessed 29-December-2018].

[48] Magali Jaillard, Leandro Lima, Maud Tournoud, Pierre Mahé, Alex van Belkum, Vincent Lacroix, and Laurent Jacob. A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events. *PLOS Genetics*, 14(11):1–28, 11 2018.

[49] Magali Jaillard, Alex van Belkum, Kyle C Cady, David Creely, Dee Shortridge, Bernadette Blanc, E Magda Barbu, W Michael Dunne, Gilles Zambardi, Mark Enright, Nathalie Mugnier, Christophe Le Priol, Stéphane Schicklin, Ghislaine Guigon, and Jean-Baptiste Veyrieras. Correlation between phenotypic antibiotic susceptibility and the resistome in *Pseudomonas aeruginosa*. *International journal of antimicrobial agents*, 2017.

[50] Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the complexity of sequence to graph alignment. jan 2019.

[51] Miten Jain, Sergey Koren, Karen H. Miga, Josh Quick, Arthur C. Rand, Thomas A. Sasani, John R. Tyson, Andrew D. Beggs, Alexander T. Dilthey, Ian T. Fiddes, Sunir Malla, Hannah Marriott, Tom Nieto, Justin O'Grady, Hugh E. Olsen, Brent S. Pedersen, Arang Rhie, Hollian Richardson, Aaron R. Quinlan, Terrance P. Snutch, Louise Tee, Benedict Paten, Adam M. Phillippy, Jared T. Simpson, Nicholas J. Loman, and Matthew Loose. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, January 2018.

[52] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.

[53] Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature Methods*, 7(12):1009–1015, nov 2010.

[54] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199 – 243, 2009.

[55] Vaddadi Naga Sai Kavya, Kshitij Tayal, Rajgopal Srinivasan, and Naveen Sivadasan. Sequence alignment on directed graphs. *Journal of Computational Biology*, 26(1):53–67, jan 2019.

[56] Daehwan Kim, Ben Langmead, and Steven L Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nature Methods*, 12(4):357–360, mar 2015.

[57] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14(4):R36, 2013.

[58] Sergey Koren, Michael C. Schatz, Brian P. Walenz, Jeffrey Martin, Jason T. Howard, Ganeshkumar Ganapathy, Zhong Wang, David A. Rasko, W. Richard McCombie, Erich D. Jarvis, and Adam M. Phillippy. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature Biotechnology*, 30(7):693–700, 2012.

[59] Sergey Koren, Brian P. Walenz, Konstantin Berlin, Jason R. Miller, Nicholas H. Bergman, and Adam M. Phillippy. Canu: scalable and accurate long-read assembly via adaptive k -mer weighting and repeat separation. *Genome Research*, 27(5):722–736, 5 2017.

[60] Anna Kuosmanen, Ahmed Sobih, Romeo Rizzi, Veli Mäkinen, and Alexandru I. Tomescu. On using longer rna-seq reads to improve transcript prediction accuracy. In *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies*, BIOSTEC 2016, pages 272–277, Portugal, 2016. SCITEPRESS - Science and Technology Publications, Lda.

[61] Tuuli Lappalainen, , Michael Sammeth, Marc R. Friedländer, Peter A. C. 't Hoen, Jean Monlong, Manuel A. Rivas, Mar Gonzàlez-Porta, Natalja Kurbatova, Thasso Griebel, Pedro G. Ferreira, Matthias Barann, Thomas Wieland, Liliana Greger, Maarten van Iterson, Jonas Almlöf, Paolo Ribeca, Irina Pulyakhina, Daniela Esser, Thomas Giger, Andrew Tikhonov, Marc Sultan, Gabrielle Bertier, Daniel G. MacArthur, Monkol Lek, Esther Lizano, Henk P. J. Buermans, Ismael Padioleau, Thomas Schwarzmayr, Olof Karlberg, Halit Ongen, Helena Kilpinen, Sergi Beltran, Marta Gut, Katja Kahlem, Vyacheslav Amstislavskiy, Oliver Stegle, Matti Pirinen, Stephen B. Montgomery, Peter Donnelly, Mark I. McCarthy, Paul Flicek, Tim M. Strom, Hans Lehrach, Stefan Schreiber, Ralf Sudbrak, Ángel Carracedo, Stylianos E. Antonarakis, Robert Häsler, Ann-Christine Syvänen, Gert-Jan van Ommen, Alvis Brazma, Thomas Meitinger, Philip Rosenstiel, Roderic Guigó, Ivo G. Gut, Xavier Estivill, and Emmanouil T. Dermitzakis. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506–511, sep 2013.

[62] T. Laver, J. Harrison, P.A. O'Neill, K. Moore, A. Farbos, K. Paszkiewicz, and D.J. Studholme. Assessing the performance of the oxford nanopore technologies minion. *Biomolecular Detection and Quantification*, 3:1 – 8, 2015.

[63] C. Lee, C. Grasso, and M. F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, mar 2002.

[64] John Lees, Marco Galardini, Stephen D Bentley, Jeffrey N Weiser, and Jukka Corander. pyseer: a comprehensive tool for microbial pangenome-wide association studies. *Bioinformatics*, page bty539, 2018.

[65] John A Lees, Minna Vehkala, Niko Välimäki, Simon R Harris, Claire Chewapreecha, Nicholas J Croucher, Pekka Marttinen, Mark R Davies, Andrew C Steer, Steven YC Tong, Antti Honkela, Julian Parkhill, Stephen D Bentley, and Jukka Corander. Sequence element enrichment analysis to determine the genetic basis of bacterial phenotypes. *Nature communications*, 7:12797, 2016.

[66] Bo Li and Colin N Dewey. RSEM: accurate transcript quantification from RNA-seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.

[67] Bo Li, Nathanael Fillmore, Yongsheng Bai, Mike Collins, James Thomson, Ron Stewart, and Colin Dewey. Evaluation of de novo transcriptome assemblies from RNA-Seq data. *Genome Biology*, 2014.

[68] Chenhao Li, Kern Rei Chng, Esther Jia Hui Boey, Amanda Hui Qi Ng, Andreas Wilm, and Niranjan Nagarajan. INC-Seq: accurate single molecule reads using nanopore sequencing. *GigaScience*, 5(1):34, 12 2016.

[69] Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, and Tak-Wah Lam. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676, jan 2015.

[70] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, mar 2016.

[71] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, may 2018.

[72] Yang I. Li, David A. Knowles, Jack Humphrey, Alvaro N. Barbeira, Scott P. Dickinson, Hae Kyung Im, and Jonathan K. Pritchard. Annotation-free quantification of RNA splicing using LeafCutter. *Nature Genetics*, 50(1):151–158, dec 2017.

[73] Y. Liao, G. K. Smyth, and W. Shi. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7):923–930, nov 2013.

[74] LibreTexts libraries. Introductory and General Biology LibreTexts. `https://bio.libretexts.org/Bookshelves/Introductory_and_General_Biology/Book%3A_Introductory_Biology_(CK-12)/3%3A_Genetics/3.8%3A_Human_Genome`, 2016. [Online; accessed 29-December-2018].

[75] Leandro Lima et al. Finding novel alternative splicing events in the presence of a shallow reference transcriptome and deep second-generation sequencing. *In preparation*, 2019.

[76] Leandro Lima, Camille Marchet, Segolene Caboche, Corinne Da Silva, Benjamin Istace, Jean-Marc Aury, Helene Touzet, and Rayan Chikhi. Comparative assessment of long-read error-correction software applied to rna-sequencing data. *bioRxiv*, 2018.

[77] Leandro Lima, Blerina Sinaimeri, Gustavo Sacomoto, Helene Lopez-Maestre, Camille Marchet, Vincent Miele, Marie-France Sagot, and Vincent Lacroix. Playing hide and seek with repeats in local and global de novo transcriptome assembly of short rna-seq reads. *Algorithms Mol Biol*, 12:2–2, Feb 2017.

[78] Nicholas J Loman, Joshua Quick, and Jared T Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature Methods*, 12(8):733–735, jun 2015.

[79] Hélène Lopez-Maestre, Lilia Brinza, Camille Marchet, Janice Kielbassa, Sylvère Bastien, Mathilde Boutigny, David Monnin, Adil El Filali, Claudia Marcia

Carareto, Cristina Vieira, Franck Picard, Natacha Kremer, Fabrice Vavre, Marie-France Sagot, and Vincent Lacroix. SNP calling from RNA-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. *Nucleic Acids Research*, page gkw655, jul 2016.

[80] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12), dec 2014.

[81] Sorina Maciuca, Carlos del Ojo Elias, Gil McVean, and Zamin Iqbal. A natural encoding of genetic variation in a burrows-wheeler transform to enable mapping and genome inference. In *WABI*, volume 9838 of *Lecture Notes in Computer Science*, pages 222–233. Springer, 2016.

[82] Mohammed-Amin Madoui, Stefan Engelen, Corinne Cruaud, Caroline Belser, Laurie Bertrand, Adriana Alberti, Arnaud Lemainque, Patrick Wincker, and Jean-Marc Aury. Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC genomics*, 16(1):327, 4 2015.

[83] Marcel Margulies, Michael Egholm, William E. Altman, Said Attiya, Joel S. Bader, Lisa A. Bemben, Jan Berka, Michael S. Braverman, Yi-Ju Chen, Zhoutao Chen, Scott B. Dewell, Lei Du, Joseph M. Fierro, Xavier V. Gomes, Brian C. Godwin, Wen He, Scott Helgesen, Chun He Ho, Gerard P. Irzyk, Szilveszter C. Jando, Maria L. I. Alenquer, Thomas P. Jarvie, Kshama B. Jirage, Jong-Bum Kim, James R. Knight, Janna R. Lanza, John H. Leamon, Steven M. Lefkowitz, Ming Lei, Jing Li, Kenton L. Lohman, Hong Lu, Vinod B. Makhijani, Keith E. McDade, Michael P. McKenna, Eugene W. Myers, Elizabeth Nickerson, John R. Nobile, Ramona Plant, Bernard P. Puc, Michael T. Ronan, George T. Roth, Gary J. Sarkis, Jan Fredrik Simons, John W. Simpson, Maithreyan Srinivasan, Karrie R. Tartaro, Alexander Tomasz, Kari A. Vogt, Greg A. Volkmer, Shally H. Wang, Yong Wang, Michael P. Weiner, Pengguang Yu, Richard F. Begley, and Jonathan M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, jul 2005.

[84] Jeffrey A. Martin and Zhong Wang. Next-generation transcriptome assembly. *Nature Reviews Genetics*, 12(10):671–682, sep 2011.

[85] M. Mele, P. G. Ferreira, F. Reverter, D. S. DeLuca, J. Monlong, M. Sammeth, T. R. Young, J. M. Goldmann, D. D. Pervouchine, T. J. Sullivan, R. Johnson, A. V. Segre, S. Djebali, A. Niarchou, T. G. Consortium, F. A. Wright, T. Lappalainen, M. Calvo, G. Getz, E. T. Dermitzakis, K. G. Ardlie, and R. Guigo. The human transcriptome across tissues and individuals. *Science*, 348(6235):660–665, may 2015.

[86] E. W. Myers. The fragment assembly string graph. *Bioinformatics*, 21(Suppl 2):ii79–ii85, sep 2005.

[87] Eugene W. Myers, Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo, Michael J. Flanigan, Saul A. Kravitz, Clark M. Mobarry, Knut H. J. Reinert, Karin A. Remington, Eric L. Anson, Randall A. Bolanos, Hui-Hsien Chou, Catherine M. Jordan, Aaron L. Halpern, Stefano Lonardi, Ellen M. Beasley, Rhonda C. Brandon, Lin Chen, Patrick J. Dunn, Zhongwu Lai, Yong Liang, Deborah R. Nusskern, Ming Zhan, Qing Zhang, Xiangqun Zheng, Gerald M. Rubin, Mark D. Adams, and J. Craig Venter. A Whole-Genome Assembly of Drosophila. *Science*, 287(5461):2196–2204, 2000.

[88] Adam M Novak, Glenn Hickey, Erik Garrison, Sean Blum, Abram Connelly, Alexander Dilthey, Jordan Eizenga, M. A. Saleh Elmohamed, Sally Guthrie, André Kahles, Stephen Keenan, Jerome Kelleher, Deniz Kural, Heng Li, Michael F Lin, Karen Miga, Nancy Ouyang, Goran Rakocevic, Maciek Smuga-Otto, Alexander Wait Zaranek, Richard Durbin, Gil McVean, David Haussler, and Benedict Paten. Genome graphs. jan 2017.

[89] Petr Novák, Pavel Neumann, and Jiří Macas. Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. *BMC Bioinformatics*, 11(1):378, 2010.

[90] Sergey Nurk, Dmitry Meleshko, Anton Korobeynikov, and Pavel A. Pevzner. metaSPAdes: a new versatile metagenomic assembler. *Genome Research*, 27(5):824–834, mar 2017.

[91] Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. Detecting superbubbles in assembly graphs. In *Lecture Notes in Computer Science*, pages 338–348. Springer Berlin Heidelberg, 2013.

[92] Fatih Ozsolak and Patrice M. Milos. Single-molecule direct RNA sequencing without cDNA synthesis. *Wiley Interdisciplinary Reviews: RNA*, 2(4):565–570, mar 2011.

[93] Qun Pan, Ofer Shai, Leo J Lee, Brendan J Frey, and Benjamin J Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40(12):1413–1415, nov 2008.

[94] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419, mar 2017.

[95] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature Biotechnology*, 32(5):462–464, apr 2014.

[96] J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley, 1ª edition, 1984.

[97] Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin. Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–i101, jun 2011.

[98] Yu Peng, Henry C. M. Leung, S. M. Yiu, and Francis Y. L. Chin. IDBA – a practical iterative de bruijn graph de novo assembler. In *Lecture Notes in Computer Science*, pages 426–440. Springer Berlin Heidelberg, 2010.

[99] Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature Biotechnology*, 33(3):290–295, feb 2015.

[100] Atif Rahman, Ingileif Hallgrímsdóttir, Michael Eisen, and Lior Pachter. Association mapping from sequencing reads using k-mers. *eLife*, 7:e32920, jun 2018.

[101] Mikko Rautiainen and Tobias Marschall. Aligning sequences to general graphs in o(v + me) time. *bioRxiv*, 2017.

[102] Anthony Rhoads and Kin Fai Au. Pacbio sequencing and its applications. *Genomics, Proteomics & Bioinformatics*, 13(5):278 – 289, 2015. SI: Metagenomics of Marine Environments.

[103] Romeo Rizzi, Gustavo Sacomoto, and Marie-France Sagot. Efficiently listing bounded length st-paths. In *Lecture Notes in Computer Science*, pages 318–329. Springer International Publishing, 2015.

[104] Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature Methods*, 10(1):71–73, nov 2012.

[105] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S Butterfield, Richard Newsome, Simon K Chan, Rong She, Richard Varhol, Baljit Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A Moore, Martin Hirst, Marco A Marra, Steven J M Jones, Pamela A Hoodless, and Inanc Birol. De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7(11):909–912, oct 2010.

[106] Roye Rozov, Gil Goldshlager, Eran Halperin, and Ron Shamir. Faucet: streaming de novo assembly graph construction. *Bioinformatics*, 34(1):147–154, jul 2017.

[107] G. Sacomoto, J. Kielbassa, R. Chikhi, and R. Uricaru *et al.* KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, 13(Suppl 6):S5, 2012.

[108] G. Sacomoto, V. Lacroix, and M.-F. Sagot. A polynomial delay algorithm for the enumeration of bubbles with length constraints in directed graphs and its application to the detection of alternative splicing in rna-seq data. In *WABI*, pages 99–111, 2013.

[109] Gustavo Sacomoto, Blerina Sinaimeri, Camille Marchet, Vincent Miele, Marie-France Sagot, and Vincent Lacroix. Navigating in a sea of repeats in RNA-seq without drowning. In *Lecture Notes in Computer Science*, pages 82–96. Springer Berlin Heidelberg, 2014.

[110] Leena Salmela and Eric Rivals. LoRDEC: accurate and efficient long read error correction. *Bioinformatics*, 30(24):3506–3514, 12 2014.

[111] Leena Salmela, Riku Walve, Eric Rivals, and Esko Ukkonen. Accurate self-correction of errors in long reads using de Bruijn graphs. *Bioinformatics*, 33(6):btw321, 6 2016.

[112] Michael Sammeth. Complete alternative splicing events are bubbles in splicing graphs. *Journal of Computational Biology*, 16(8):1117–1140, aug 2009.

[113] M. H. Schulz. *Data Structures and Algorithms for Analysis of Alternative Splicing with RNA-Seq Data*. PhD thesis, Free University of Berlin, 2010.

[114] Marcel H. Schulz, Daniel R. Zerbino, Martin Vingron, and Ewan Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, feb 2012.

[115] Mingfu Shao and Carl Kingsford. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature Biotechnology*, 35(12):1167–1169, nov 2017.

[116] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nat Biotechnol*, 26(10):1135–1145, 2008.

[117] Richard Smith-Unna, Chris Boursnell, Rob Patro, Julian Hibberd, and Steven Kelly. TransRate: reference free quality assessment of de novo transcriptome assemblies. *Genome Research*, pages gr.196469.115+, June 2016.

[118] Lincoln Stein. Genome annotation: from sequence to biology. *Nature Reviews Genetics*, 2:493 EP –, Jul 2001. Review Article.

[119] Wing-Kin Sung, Kunihiko Sadakane, Tetsuo Shibuya, Abha Belorkar, and Iana Pyrogova. An o(m log m)-time algorithm for detecting superbubbles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(4):770–777, jul 2015.

[120] Sonia Tarazona, Pedro Furió-Tarí, David Turrà, Antonio Di Pietro, María José Nueda, Alberto Ferrer, and Ana Conesa. Data quality aware analysis of differential expression in RNA-seq with NOISeq r/bioc package. *Nucleic Acids Research*, page gkv711, jul 2015.

[121] Oxford Nanopore Technologies. World first: continuous DNA sequence of more than a million bases achieved with nanopore sequencing. https://nanoporetech.com/about-us/news/world-first-continuous-dna-sequence-more-million-bases-achieved-nanopore-sequencing, 2017. [Online; accessed 29-December-2018].

[122] Hagen Tilgner, David Knowles, Rory Johnson, Carrie Davis, Sudipto Chakrabortty, Sarah Djebali, João Curado, Michael Snyder, Thomas Gingeras, and Roderic Guigó. Deep sequencing of subcellular RNA fractions shows splicing to be predominantly co-transcriptional in the human genome but inefficient for lncRNAs. *Genome research*, 22:1616–1625, 2012.

[123] German Tischler and Eugene W. Myers. Non Hybrid Long Read Consensus Using Local De Bruijn Graph Assembly. *bioRxiv*, page 106252, 2 2017.

[124] Alexandru I. Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. A novel min-cost flow method for estimating transcript expression with rna-seq. *BMC Bioinformatics*, 14(5):S15, Apr 2013.

[125] Cole Trapnell, David G Hendrickson, Martin Sauvageau, Loyal Goff, John L Rinn, and Lior Pachter. Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nature Biotechnology*, 31(1):46–53, dec 2012.

[126] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, may 2010.

[127] Isaac Turner, Kiran V Garimella, Zamin Iqbal, and Gil McVean. Integrating long-range connectivity information into de bruijn graphs. *Bioinformatics*, 34(15):2556–2565, mar 2018.

[128] Raluca Uricaru, Guillaume Rizk, Vincent Lacroix, Elsa Quillery, Olivier Plantard, Rayan Chikhi, Claire Lemaitre, and Pierre Peterlongo. Reference-free detection of isolated SNPs. *Nucleic Acids Research*, 43(2):e11–e11, nov 2014.

[129] Eric T. Wang, Rickard Sandberg, Shujun Luo, Irina Khrebtukova, Lu Zhang, Christine Mayr, Stephen F. Kingsmore, Gary P. Schroth, and Christopher B. Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470–476, nov 2008.

[130] Thomas D. Wu, Jens Reeder, Michael Lawrence, Gabe Becker, and Matthew J. Brauer. GMAP and GSNAP for genomic sequence alignment: Enhancements to speed, accuracy, and functionality. In *Methods in Molecular Biology*, pages 283–334. Springer New York, 2016.

[131] Chuan-Le Xiao, Ying Chen, Shang-Qian Xie, Kai-Ning Chen, Yan Wang, Yue Han, Feng Luo, and Zhi Xie. MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *Nature Methods*, 14(11):1072–1074, 9 2017.

[132] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, and J. Wang. SOAPdenovo-trans: de novo transcriptome assembly with short RNA-seq reads. *Bioinformatics*, 30(12):1660–1666, feb 2014.