THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Real-time analysis for Nanopore sequencing data

Son Hoang Nguyen

B.Sc, M.Sc

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2019

Institute for Molecular Bioscience

Examiner's Copy

Typeset in LaTeX 2_ε.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Son Hoang Nguyen

Abstract

The introduction of third-generation sequencing technologies presents many challenges to the traditional methods in which bioinformatics is applied to genomics data. The MinION thumb-drive sequencing device has gained great attention from researchers worldwide. Despite the relatively high error rates compared to Second Generation Sequencing technologies, the nanopore approach is widely considered to be a turning point due to its ability to decode much longer reads (tens or even hundreds of kilobase pairs) in a real-time fashion.

Prior to the work presented in this thesis, there was no available method that could scaffold and finish assemblies in real-time while the nanopore sequencing run is still in progress. Such a method is desirable because it offers the opportunity to obtain analysis results as soon as sufficient data are generated. With real-time analysis, answers to questions of interests could be obtained *in situ*, in an automated manner that saves considerable amount of time and resources compared to the conventional approach of sending sample to a sequencing centre, waiting for bulk data, and conducting a batch analysis. On top of that, streaming analysis can help to avoid under- and over-sequencing which could result in either the generation of more sequence data than required at greater cost or a low quality assembly if insufficient data are generated.

For the reasons mentioned above, the motivation of this thesis project is to develop methodologies for streaming data analysis of long reads for real-time finishing genome sequences. As the initial result, in Chapter 2, I introduce `npScarf` which can scaffold and complete short read assemblies alongside with the long read sequencing run. This tool operates on an input of contigs, attempting to bridge them together by using long-read data

and reports assembly metrics in real-time so the sequencing run can be terminated once an assembly of sufficient quality is obtained.

It is also desirable to extend the pipeline application for multiple samples at the same time, through a parallel mechanism known as barcoded sequencing. In Chapter 3, `npBarcode`, a tool supporting real-time demultiplexing of nanopore sequencing data, is employed to serve that purpose. Depending on requirements, users can choose to run the dedicated demultiplexer from the command line or using it as part of the `npReader`'s graphical user interface (GUI). The tool provides practitioners a flexible option to monitor a barcoded sequencing run as well as to integrate pooled sequencing into a streaming analysis pipeline. For example, in combination with `npScarf`, we can complete multiple genomes in parallel.

Users can also provide underlying assembly graph structure from short-read assemblers for better quality. This approach is described in Chapter 4. In which, a streaming algorithm is implemented together with GUI in `npGraph`. The benefits of using assembly graph stems from the fact that by traversing the graph of the contigs' building blocks, we can reduce the number of misassemblies and errors in the final sequences.

Chapter 5 discusses another application of nanopore sequencing for decoding small genomes. By employing rolling circle amplification, long reads containing multiple copies of a given viral genome can be obtained. The duplicated patterns are possibly identified by a detection module that can even work with raw signal data. The developed modules, which allow for single-molecule genome assembly of small genomes, can be integrated in a streaming pipeline for real-time analyses as well.

In summary, my thesis project aims to develop and apply in-house tools that aid genome assembly and analysis in real-time in an attempt to facilitate the applications of nanopore sequencing for various use cases, including but not limited to microbial genomics.

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Publications included in this thesis

Involved publications are exclusively listed below with the name(s) of candidate highlighted in **bold** and (co)first author(s) underlined. Reproduction and/or remake of any publication or parts of project in this thesis has been approved by its corresponding authors.

- Cao, M. D., **Nguyen, S. H.**, Ganesamoorthy, D., Elliott, A. G., Cooper, M. A., and Coin, L. J. (2017). Scaffolding and completing genome assemblies in real-time with nanopore sequencing. *Nature Communications*, 8, 14515.

- **Nguyen, S. H.**, Duarte, T. P., Coin, L. J., and Cao, M. D. (2017). Real-time demultiplexing Nanopore barcoded sequencing data with npBarcode. *Bioinformatics*, 33(24), 3988-3990.

Submitted manuscripts included in this thesis

- Pitt, M., **Nguyen, S. H.**, Duarte, T. P., Blaskovich, M., Cooper, M., and Coin, L. (2018). Evaluating the Genome and Resistome of Extensively Drug-Resistant *Klebsiella pneumoniae* using Native DNA and RNA Nanopore Sequencing.

Other publications during candidature

Peer-reviewed papers:

- Jin, M., Lu, J., Chen, Z., **Nguyen, S.H.**, Mao, L., Li, J., Yuan, Z. and Guo, J. (2018). Antidepressant fluoxetine induces multiple antibiotics resistance in *Escherichia coli* via ROS-mediated mutagenesis. *Environment international*, 120, 421-430.

- Lu, J., Wang, Y., Li, J., Mao, L., **Nguyen, S.H.**, Duarte, T., Coin, L., Bond, P., Yuan, Z. and Guo, J. (2018). Triclosan at environmentally relevant concentrations promotes horizontal transfer of multidrug resistance genes within and across bacterial genera. *Environment international*, 121, 1217-1226.

- Lu, J., Jin, M., **Nguyen, S.H.**, Mao, L., Li, J., Coin, L.J., Yuan, Z. and Guo, J. (2018). Non-antibiotic antimicrobial triclosan induces multiple antibiotic resistance through genetic mutation. *Environment international*, 118, 257-265.

Submitted:

- Bialasiewicz, S., Duarte, T.P., **Nguyen, S.H.**, Sukumaran, V., Stewart, A., Appleton, S., Pitt, M.E., Bainomugisa, A., Jennison, A.V., Graham, R. and Coin, L.J., (2018). Rapid Diagnosis of *Capnocytophaga canimorsus* Septic Shock in an Immunocompetent Individual Using Real-Time Nanopore Sequencing. *Preprints*, 2018110395 (doi: 10.20944/preprints201811.0395.v1).

Conferences abstract:

- Poster: *Completing microbial draft genomes with Oxford Nanopore sequencing data.* Has been presented in:
  - The Australian Bioinformatics And Computational Biology Society (**ABACBS**) – Garvin Institute of Medical Research, Sydney 2015.
  - Big Biology and Bioinformatics Symposium ($\mathbf{B^3}$) – Queensland University of Technology, Brisbane 2015.
- Oral presentation: *Scaffolding and completing genome assemblies in real-time with nanopore sequencing.* **COMBINE** Symposium – Queensland University of Technology, Brisbane 2016.

Web resources:

- Software presented in Chapter 2 and Chapter 3 are bundled in Japsa (Java Package for Sequence Analysis) project `https://github.com/mdcao/japsa`.
- Software presented in Chapter 4 and Chapter 5 can be found in `https://github.com/hsnguyen/assembly`.

Contributions by others to the thesis

- A/Prof Lachlan Coin, Dr Minh Duc Cao provided assistance and inputs to the conceptual design of the software tools. Dr Minh Duc Cao implemented fundamental Java modules (Japsa package) which had been used as the codebase for the tools.
- Dr Miranda Pitt and Tania Duarte conducted lab experiments and provided details for Chapter 3 (sample details, library preparation, sequencing methods, discussions).
- A/Prof Andrew Geering and A/Prof Lachlan Coin designed the experiment and supervised the project in Chapter 5 while Ha Ngo Thu and Tania Duarte conducted lab works and generated the data.
- A/Prof Lachlan Coin, Dr Minh Duc Cao, Dr Hoang Nguyen gave feedback and revisions for the thesis.

Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

Research Involving Human or Animal Subjects

No animal or human subjects were involved in this research.

Acknowledgements

It is great honor for me to work in Coin group for my PhD project and for that, my deepest appreciation goes to my supervisors, A/Prof Lachlan Coin and Dr Minh Duc Cao, for giving me such opportunity. Lachlan, with his incredible insight, broad knowledge and kind manners, has made my candidature such a precious experience toward research career. Minh, on the other hand, had been the one who willingly spent his precious time to advised me thoroughly from the very beginning and still offers very helpful mentoring after leaving the group. I have learnt countless from him, both in and out of the office. I would like to thank A/Prof Lutz Krause also for being my next co-supervisor for the last two years. It is privilege for me to be able to work under your guidance.

Special thanks go to my thesis committee including A/Prof Scott Beatson, Dr Cheong-Xin Chan, A/Prof Joselph Powell. My thesis benefits substantially from their advice through three milestone meetings. I would also like to express deep gratitude to Dr Amanda Carozzi for her continuously administered assistance during my candidature. This thesis would not be finished on time without all of their supports.

Undoubtedly, my PhD life cannot become such enjoyable time without my friends and colleagues. I appreciate all the knowledge, games and laughter shared with Arnold, Chenxi, Devika, Miranda and Tania in our lab. I will forever remember every Tuesday soccer games as well as rooftop Friday activities at IMB as they have already become parts of my life. I am obligated to UQ, not only because of the financial support for my studying but also the beautiful jacarandas running along my everyday bikeway to the campus.

Last but not least, I want to say thanks to my family, especially my brother, Dr Hoang Nguyen, who showed incredible efforts in proofreading this thesis while taking care of my beloved newborn niece at the same time.

This thesis is dedicated to my Mother and maternal Grandparents who have always been there in my heart and watching my way.

Financial support

This research was supported by the University of Queensland Centennial Scholarship.

Keywords

bioinformatics, computational biology, genomics, antibiotic resistance, nanopore sequencing, genome assembly, real-time analysis

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080301, Bioinformatics software, 40%

ANZSRC code: 060408, Genomics, 40%

ANZSRC code: 060503, Microbial Genetics, 20%

Fields of Research (FoR) Classification

FoR code: 0803, Computer Software, 40%

FoR code: 0604, Genetics, 40%

FoR code: 0605, Microbiology, 20%

Dedications

This thesis is dedicated to Mother, Grandpa and Nanny – who never stop inspiring me.

Contents

List of Figures

List of Tables

List of Abbreviations

ACF	Auto-correlation Function
AMR	Antimicrobial Resistance
(K/M)bp	(Kilo-/Mega-)base pair(s)
DNA	Deoxyribonucleic Acid
DFS	Depth First Search
DSP	Digital Signal Processing
LPF	Low Pass Filter
FFT	Fast Fourier Transform
NCBI	National Center for Biotechnology Information
NKDF	Normalized Kronecker Delta Function
NSDF	Normalized Square Difference Function
ONT	Oxford Nanopore Technologies
PacBio	Pacific Biosciences
PCR	Polymerase Chain Reaction
PDR	Pan-Drug Resistant
POA	Partial Order Alignment
RCA	Rolling-circle Amplification
RNA	Ribonucleic Acid
SGS	Second-Generation Sequencing
XDR	Extensively Drug Resistant
TGS	Third-Generation Sequencing

1

Introduction

Genes are like the story, and DNA is the language that the story is written in

–Sam Kean

1.1 Reading genome – the book of life

The mission of decoding the genetic information of all organisms has been attracting great attention and effort from the research community over the past three decades. Although genomes are thought to contain nearly all the information necessary to create an organism, sequencing a genome is only the very first step towards understanding the development of an organism. Subsequent to genome sequencing and annotation there remains a huge amount of additional work to interpret the genome and understand how genes interact in pathways in a variety of contexts to sustain life. Nevertheless, the importance of the initial reading is indisputable as it would shape the details and output of any further analysis. In fact, the competition of modern sequencing technologies have never been on hiatus, resulting in remarkably achievements in terms of both quality and quantity. However, unfortunately, we are still far away from having a flawless sequencer to date.

All genetic material of an organism is embraced in the term *genome* mentioned earlier. Basically, the genome includes deoxyribonucleic acid (DNA) molecules – the double helix polymers [1] that virtually define a whole individual life form (except some viruses made by ribonucleic acid or RNA). A DNA molecule is a sequence of nucleotides, which is represented by 4 bases: adenine (A), cytosine (C), guanine (G), or thymine (T). The process of calling each and every single base of this sequence is known as *DNA sequencing*. Sequencing an entire organism's genome is termed Whole Genome Sequencing (WGS). Innovation in sequencing technology has enabled WGS for a large variety of species with increasing genome sizes and complexity over time, such as *Haemophilus influenzae* bacterium [2], *Saccharomyces cerevisiae* [3], *Caenorhabditis elegans* [4] and *Mus musculus* (mouse) [5]. Ultimately, the completion of the Human Genome Project [6] marked the beginning of the genomic era in which rapidly developing technology would make WGS much more efficient and affordable. The cost of sequencing has been dropped drastically over time to the point that scientists are now planning for the Earth BioGenome Project, which plans to sequence 1.5 million different species across multiple continents[7].

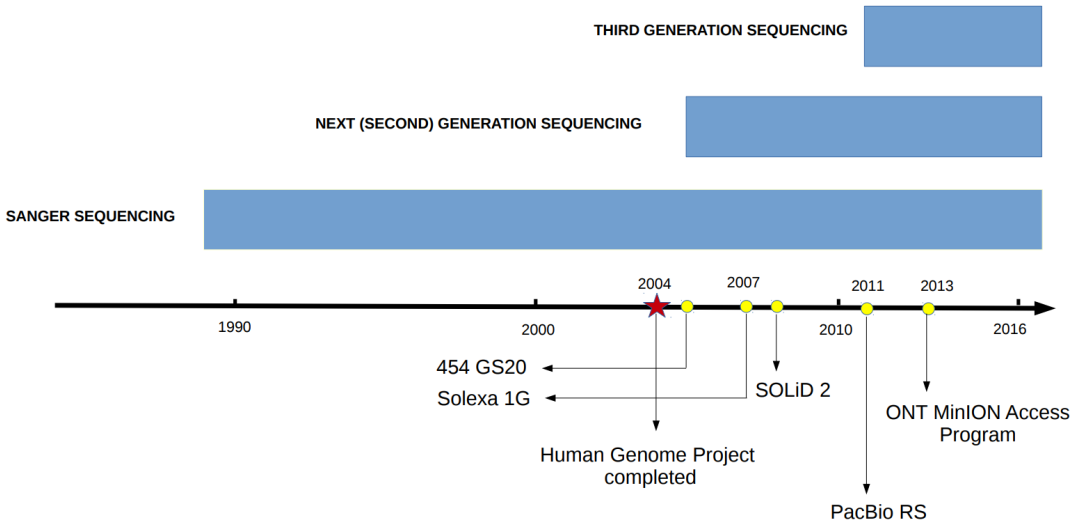


Figure 1.1: Prominent sequencing platforms in the era of genomics.

1.1.1 DNA sequencing technology

The first practical sequencing technology was invented by Frederick Sanger and colleagues [8] and as the consequence, had been named after its founder. Sanger sequencing, can be considered as the *First generation sequencing*, employs the mechanism of selectively terminating the chain elongation by dideoxy nucleotide, which is a nucleotide analogue which lacks a 3‘-hydroxyl group needed for the next incorporation and extension. Improved Sanger platforms nowadays can output reads up to around 1Kbp with high accuracy (99.9%) but the yields and cost are still unreasonable compared with other methods. In fact, Sanger sequencing contributed vastly at the beginning of the sequencing era but became supplanted later by *Second-Generation Sequencing* (SGS) machines (previously known as Next-Generation Sequencing) which are of much larger scale and higher throughput in term of data acquired. For that reason, it is restricted in use today, and used largely only for validation or in combination with other deep-sequencing methods [9]. SGS platforms, on the other hand, can output a large amount of DNA sequence with length up to 1Kbp in form of single, mate-paired or paired-end reads on demand, for a large ranges of studies. Prominent sequencing technologies include pyrosequencing [10, 11] (Roche 454), ion torrent(ThermoFisher), SOLiD sequencing (Applied Biosystems) and sequencing by synthesis (Illumina/Solexa).

TABLE 1.1: Comparison of DNA sequencing methods in general. Each method may have several platforms with varied configurations for different usage. More details in [12, 13].

Gen.	Sequencing methods	Read length (bp)	Accuracy (%)	Reads per run	Time per run
1^{st}	Chain termination (Sanger sequencing)	$400 - 900$	99.9	N/A	$0.3 - 3$ hours
2^{nd}	Pyrosequencing (454)	700	99.9	1 million	24 hours
	Sequencing by synthesis (Illumina/Solexa)	up to 300	> 99	up to 3 billions	$< 1 - 14$ days
	Sequencing by ligation (SOLiD sequencing)	$75 + 35$	99.9	up to 2.8 billions	$1 - 2$ weeks
	Ion semiconductor	200	98	5 millions	2 hours
3^{rd}	Single molecule, real-time sequencing	3000 on average	85	$\approx$ 50K	2 hours
	Nanopore sequencing	extremely long	$85 - 96$	$\approx$ 150K per flow cell	$<$ 48 hours (real-time output)

After years of innovation, Illumina is currently leading the sequencing market. This giant company reportedly has much larger revenues compared to other competitors [14] and its equipment is ubiquitous amongst genomics labs worldwide. In fact, the sequencing productivity, quality and deployment cost-effectiveness have been enhanced significantly with the introduction of new platforms such as MiniSeq, MiSeq series, NextSeq, HiSeq and HiSeq X and Novaseq series. These sequencers provide high-fidelity paired-end reads of length 100-300bp each. In consequence, bacterial DNA sequencing is typically done at average sequencing depth of more than 100-folds. In other words, each of every bases in the whole genome would be covered by around 100 reads using such platforms. In this thesis, data from Illumina MiSeq or HiSeq were used as input for the assembly algorithms.

The race for better sequencing technology is still going on rapidly and intensively, illustrated by the emergence of so-called *Third generation* or long-read sequencing technology [15, 16] with two major representatives *i.e.* Pacific Biosciences of California, Inc. (PacBio)

and Oxford Nanopore Technologies Limited (ONT). As indicated by the name, the defining feature of this category is the ability to measure in real-time the signals of every single molecule and output significantly longer reads than the previous generation sequencers. Table 1.1 presents the performances of above-mentioned sequencing platforms in term of read length, yield, accuracy and running time.

At the time this thesis being written, Illumina has come to an agreement to acquire PacBio for approximately $1.2 billions in an attempt to expand its DNA sequencing capacity. At the same time, ONT is continuing to grow into a essential player in the field of genome sequencing with remarkable success stories. The next section will briefly shed light into TGS focusing on the latter technology.

1.1.2 Third-generation sequencing technology

Single molecule, real-time (SMRT) sequencing The first long-read sequencer available on market was the PacBio RS from Pacific Biosciences of California, Inc. in 2011.

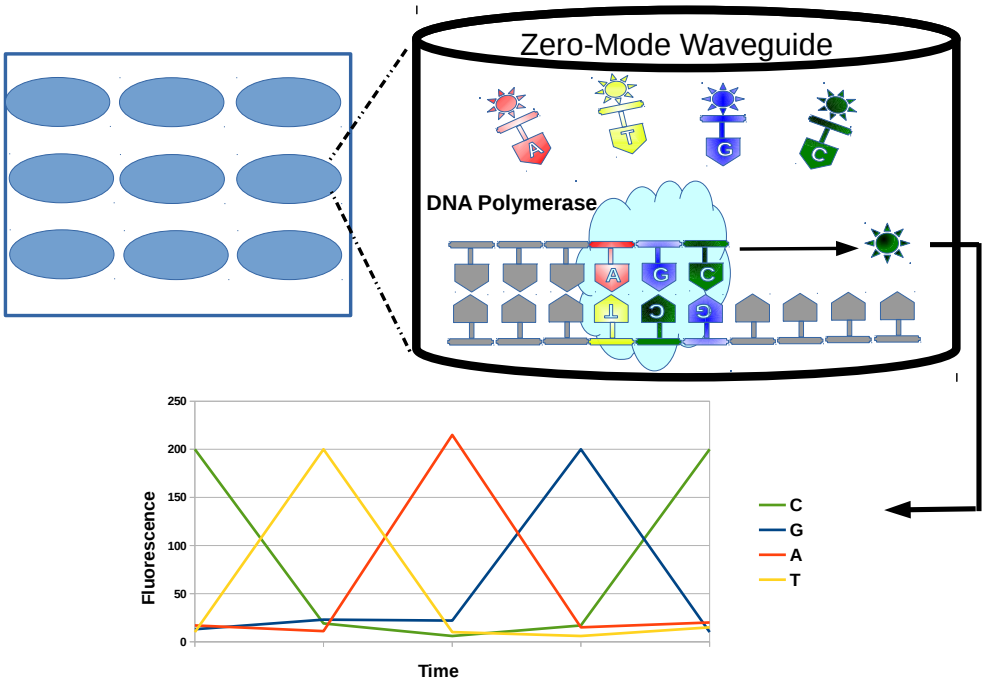


Figure 1.2: Mechanism of SMRT sequencing with Zero-Mode Waveguides.

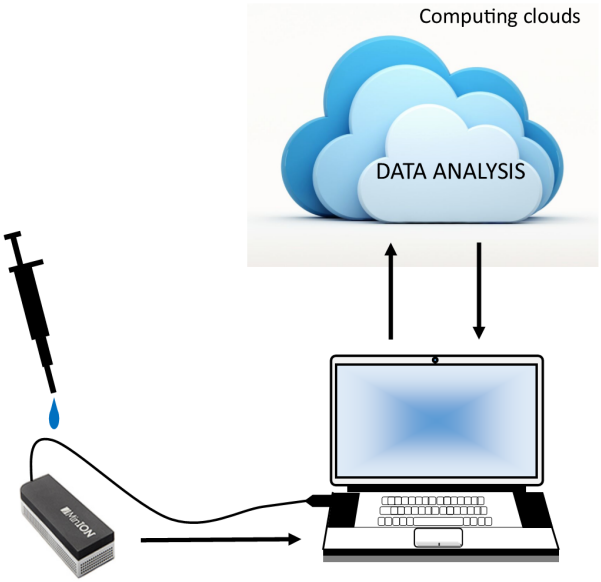
The SMRT technology takes advantage of *zero-mode* waveguides (ZMW) to detect the

activity of DNA polymerase incorporating a single nucleotide at a time [17]. A ZMW structure consists of a circular hole in sub-wavelength scale (20×10^{-21} litre volume) in a metal film [18]. A complex of DNA polymerase and a single stranded template DNA molecule is immobilized at the bottom of the chamber surrounded by fluorescent dyed bases, as shown in Figure 1.2. The DNA synthesis activity cleaves off the dyed tag of each incorporated base which can be detected optically in real-time within this smallest available microscopy structure [19]. A SMRT cell can harbor thousands (RS platform) to hundred thousands (RS II) or even millions of ZMWs (Sequel, 8M chip) to facilitate parallelization and thus improve the throughput per cell.

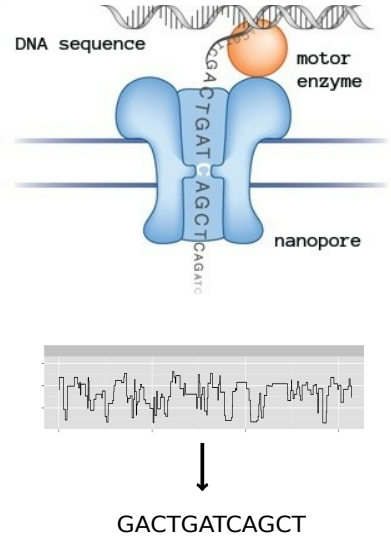
In fact, both the read length and accuracy of the instrument have been continuing to increase significantly. The average accuracy per read has increased from about 82% in the first release to 87% [20, 21] whilst the maximum read length has reached over 50Kbp in recent runs [22]. As the consequence, PacBio DNA reads have gone from only being used in hybrid assembly which required high-fidelity complementary reads [23, 24], to now being able to generate finished *de novo* bacterial genomes on its own [21].

Oxford Nanopore sequencing In late 2013, ONT released the first nanopore sequencer, the *MinION*, in the *MinION Access Program* (MAP). MAP was offering opportunity to researchers worldwide to have access on the third-generation reads. Of the particular advantages using this device is its greater portability and flexibility.

Figure 1.3a presents a general workflow of a sequencing run using MinION. After DNA extraction, the very first step – sample and library preparation, is straightforward but can also be varied for different experimental aims, such as maximizing yield, accuracy or read-length; as well as different experimental conditions, such as amount and quality of starting DNA. The purpose of library preparation is to bring necessary adapters and/or chemistry to the input mixture prior to the sequencing. ONT provides a variety of sequencing kits for different purposes. For example, using the Rapid Sequencing Kit is simple and quick (10 minutes in theory) that would fit experiments with critical time and resource condition *e.g.* remote locations. On the other hand, the Ligation Sequencing Kit family is more time-consuming (about an hour) but gives better control over read length. Other than



(a) Sequencing with MinION



(b) Inside the pore

Figure 1.3: Illustration of nanopore sequencing with MinION. (a) General setup for MinION sequencing. (b) Nanopore sequencing mechanism at molecular level.

these, ONT offers expansion packs for diverse project scenarios such as Barcoding kits for multiplexed sequencing or Low input kits for sequencing as little as 10ng of DNA molecules. In addition, if better quality is desired, users could, in the early stages, consider using a 2D kit (which would sequence both DNA strands in a single read), which has been replaced with an updated $1D^2$ protocol (which with high probability sequences the second strand in the same pore immediately after sequencing the first strand, but as a separate read). After loading the prepared mixture to the device, the sequencing and base-calling phases could be employed automatically and *in situ* thanks to the built-in services. In combination with a MinION, it is only required to have an internet-connected laptop with proper software installed to run the experiment, making it possible to establish sequencing-on-the-field where access to lab equipment is limited.

Figure 1.3b illustrates the sequencing mechanism in more details. In fact, the general mechanism of an artificial nanopore has been proposed [25, 26] long before being applied as a real DNA sequencing technique by Hagan Banley and ONT [27]. In which, the unwound molecules are threaded through a nano-scaled pore on an electrically charged membrane

protein known as *the nanopore*. In the whole process, the velocity of the molecule translocating through the pore is controlled by a motor enzyme. The rationale of this technology is that the shift of nucleotide sequence inside of a pore leads to the variation in its electrical resistance, resulting in a change in the current signal which is recorded as squiggles by time. These squiggle signals are feasibly translated back to the format of strings of nucleotide identities. Indeed, a Hidden Markov Model (HMM) [28, 29] in the older version of Albacore, or Recurrent Neural Network (RNN) deep learning algorithm in later versions of Albacore, Chiron [30], Guppy are used to study the transition and predict the sequence of DNA that most likely would generate these signals. Due to the unavoidable noises of the measures in such tiny scale, a certain innate level of error may be expected from the base callers, although ONT has proposed using a variety of different types of pores simultaneously in order to generate independent error profiles.

The nanopores are packed into a *flow cell*, which is the consumable of this technology. Start-up cost of operating MinION is virtually zero, as users only have to pay for running costs, making it suitable for any project scale. The chemistry and computational features are likewise updated continuously to help reduce the noises from output. Recently, R10 flow cell is made available for higher signal accuracy, together with the newest base-calling algorithm from Guppy flip-flop pipeline that supports GPU for heavy computational deep-learning tasks, it has shown positive progress of this sequencing technology to tackle errors. When the demand of resources optimization is concerned, *flongle* (single-use flow cell) can be used for small genomes. On the other hand, washing kits can be applied to reactivate pores from an interrupted flow cell, making it ready for another run.

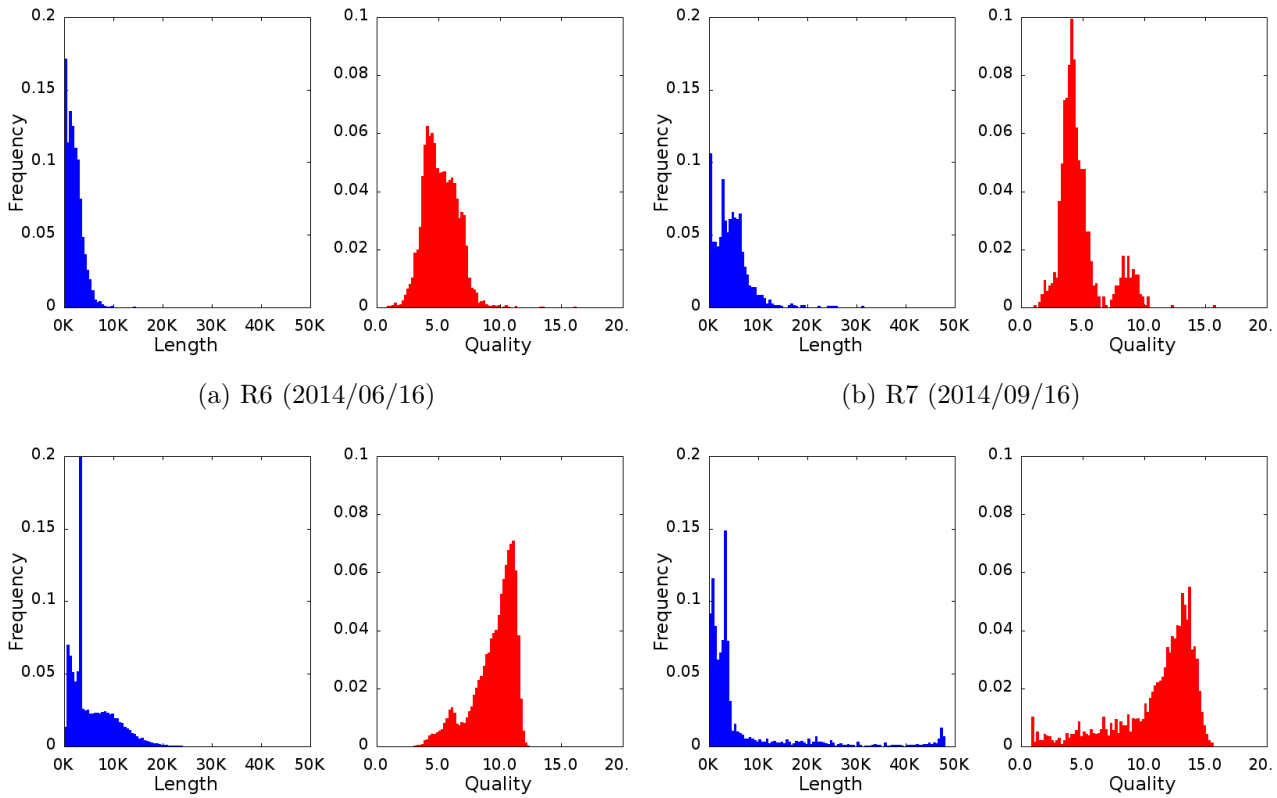
Furthermore, GridION and ProthmethION – the scaled-up version of MinION, are made to public by the company through its access program and has already been deployed for several experiments involving bigger and more complicated genomes. This included human genome [31] and metagenomics mock community [32] where data is available at `https://github.com/LomanLab/mockcommunity`.

In summary, ONT provides promising and fast-growing sequencing technology that could facilitate substantial applications in real-life. The first description of the data generated by this sequencing method at early stage will be presented in the following section.

1.2 Nanopore long-read data at first glance

Together with PacBio RS series, the ONT MinION sequencing device, through its early access program (MAP), has become a prominent long-read data generator for researchers worldwide. An initial interrogation on the Nanopore data generated by MinION over time here would provide first impression on its performance as well as the innovating progress that had been made to improve the output.

1.2.1 Data description



(a) R6 (2014/06/16)

(b) R7 (2014/09/16)

(c) R9.4 (2017/01/18)

(d) R9.5 (2018/09/12)

Figure 1.4: Statistics of nanopore reads using different flow cell versions (with the running date) on the lambda phage sample, shown in the order in which they were released. For each flow cell, the blue normalized histogram is for read length and red one is for quality (Phred score).

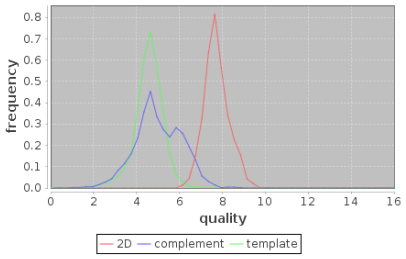
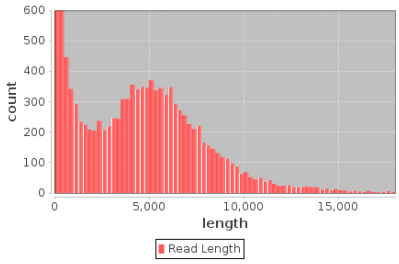
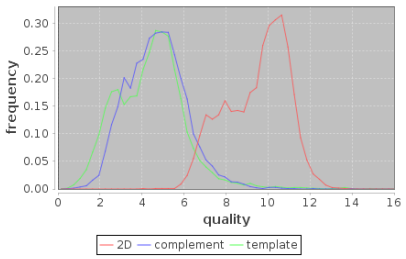
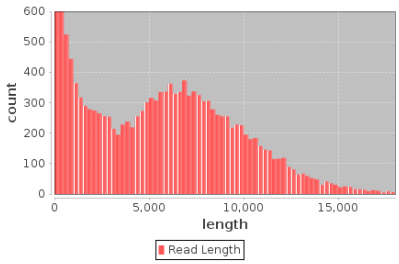
1.2.1.1 Lambda burn-in experiments

Figure 1.4 shows statistics of our lambda experiments acquired by using progressively more update version of MinION flow cells. This type of practice is a standard control 'burn-in' experiment recommended by ONT to help users familiarize themselves with the protocol, the equipment and the result long-read data. The sample is a virus named lambda phage with small genome size of 50Kbp. All the test experiments were run for 6 hours, except R9.5 with a 1-hour run so the sequencing yields are not included in the comparison and the histogram frequencies are normalized by the number of reads for each attempt. Even though the 'burn-in' experiment is not designed to exhaustively study the potential length of nanopore reads due to the limit genome size of the sample, it is easily to see from the figure a clear improvement of read length distribution by applying newer version of flow cells. The early-staged (2014-released) flow cells and chemistry in the kit produced fewer reads longer than 10Kbp, as could be observed from Figure 1.4a and 1.4b. This remarked value has become the mean length of reads generated by R9.4 flow cell which can even possibly reach 20Kbp in extreme cases. Using the newest available flow cell R9.5 - even for just a 1-hour run - produced many more longer reads, including the maximum possible 50Kbp reads that cover the whole genome. Similarly, the fidelity of the long reads has improved with the 2016-released R9 flow cells. The average quality scores (in Phred scale) of the base-called sequences have advanced from 5.0 ($P_{error} \approx 0.3$) in R6 and R7 to 10.0 ($P_{error} \approx 0.1$) in R9.4 and 13.0 ($P_{error} \approx 0.05$) in R9.5. To further enhance the quality, reads from two complementary strands can be sequenced and computationally merged to produce a single double-checked 2D or $1D^2$ read. Even so, noises will inevitably remain in nanopore reads as the result of measurement noise at nano-scale, as compared to high-precision data generated by SGS when the average Phred score is usually 30 or more ($P_{error} \leq 0.001$).

1.2.1.2 The first runs with bacteria data

Amongst our first-hand experiments with real-life samples were the ones with *K. pneumoniae* strains as shown in Table 1.2. Flow cell version R7 and R7.3 with 2D sequencing protocol were used for the runs resulting in 35-fold coverage of long-read data for NDM-1

Table 1.2: Sequence two *K. pneumoniae* strains with the MinION. Figures generated by `npReader` GUI [33].

Strain	Phred Quality Scores	Read Length Distribution	Emp. errors
BAA-2146 (NDM-1 strain) Chemistry R7 Sep 2014 35-X Coverage			Del: 9.5% Ins: 6.3% Mis: 15.3% Unaligned: 13.3%
13883 (type strain) Chemistry R7.3 Dec 2014 15-X coverage			Del: 7.9% Ins: 6.0% Mis: 12.9% Unaligned: 11%

strain and 15-fold coverage for the type strain. From the quality histograms, the red lines represent 2D reads' scores which are clearly better than the ones in 1D reads (green and blue for template and complement sequences respectively). The length distribution shows that MinION sequencing returned significant reads with length around $5K$ for the first sample and $7K$ for the latter. The statistics on the far right of the table show error rate in term of indels, mismatches and hit rate from alignments with reference data. Overall, chemistry R7.3 is more up-to-date than R7 and as expected, gave slightly better results. As a next step, there came a mission to assembly those reads, together with available Illumina data of corresponding samples, in an attempt to have complete genome reconstructions for these two *K. pneumoniae* strains. This task was important at the time as it would depict bigger picture about interactions between genetic elements in the whole genome and help to comprehend the biological pathways supporting this superbug's mechanisms.

1.2.2 Data analysis: challenges and solutions in working with nanopore data

The introduction of long, error-prone nanopore reads has motivated novel developments in bioinformatics, as many existing approaches had been optimized for high-quality short read data. A class of efforts was made to carry out an initial error-correction step before using the dataset [34, 35]. This process usually require either another reference of high quality SGS reads or a large amount of long reads for self corrections. The majority of algorithms following this approach uses alignment-based correction which is normally costly in term of computation.

Another attempt to reduce the error rate from base-called sequences is to utilize the signal-level information. Nanopolish is designed for such task. The underlying technique is to train a hidden Markov model that measures the probability of observing a certain chain of events given a particular DNA sequence. This method has been used for polishing a draft assembly using an abundance of long-read data [36] and has been extended to detect methylation [37]. Nevertheless, this approach is computational expensive and usually requires parallelization which is supported in its modules.

Pairwise sequence alignment is a critical operation for many genomic analyses in general and comparative studies in particular. To adapt to the specific error characteristics of nanopore sequencing data, a number of approaches has been introduced on top of the legacy alignment methods used for short-read SGS data. The ubiquitous practice is still based on the previous seeds-and-extend dynamic programming technique with modifications in term of seed finding, gap extension and chaining algorithms. In addition, there would be justified settings to relax the matching criteria thus allowing more error tolerance and lengthen the alignments. As the results, early long-read alignment tools have been developed or calibrated for PacBio SMRT data, including but not limited to BLASR [38], LASTZ [39], LAST [40], BWA-MEM [41] and DAligner [42]. Recently, several other tools has been designed specifically for comparing long-read data such as GraphMap [43] and especially `minimap` [44] (now `minimap2`) which significantly reduce the running time while maintaining comparative accuracy at the same time. These available solutions have been widely applied in a flexible ways

for different analysis purposes when it comes to long reads with high error rate.

The problem of aligning raw long reads stems from the error profile per base in the targeting sequences, especially indels, that would introduce combinatorial explosion using traditional approaches. The common practice to tackle this problem is to apply *hashing* techniques to reduce the dimensional of the data. A hash function maps a particular string to an index value so that it can be accessed directly without (or minimized) collision. By using these functions, a DNA sequences can be represented by a small set of fingerprints, known as sketch, that will be shared between similar but not necessary exact strings. The prominent sketch types that has been used for nanopore data includes MinHash [45, 46], minimizer [44], HyperLogLog (*Dashing: Fast and Accurate Genomic Distances with HyperLogLog* Daniel N Baker, Benjamin Langmead. **bioRxiv** 501726; doi: `https://doi.org/10.1101/501726`).

For the multiple sequence alignment (MSA) problem, the partial order alignment (POA) graph data structure has been proposed to cope with errors in long reads [47–49]. By applying this method, the authors proved the integrity of information compared to other MSA format and developed POA as an efficient tool for large alignment and consensus calling problem. The latter was then adopted in Racon [50], a commonly used consensus module for long uncorrected reads.

Genome assembly is another major challenge in bioinformatics when the traditional approaches for SGS data cannot be applied directly to TGS data. To better understand the situations and come-up solutions, the general principle of genome assembly and methods for SGS data will be addressed in the next section. Approaches to genome assembly which utilize nanopore reads will come later on top of this foundation knowledge.

1.3 Genome assembly

1.3.1 Definition

Despite the huge improvements of sequencing technology, there still exist the common limitation that in most cases, it is impossible to sequence directly the whole length of a genome. As shown in Figure 1.5, the actual practice is to break the whole genome into smaller pieces

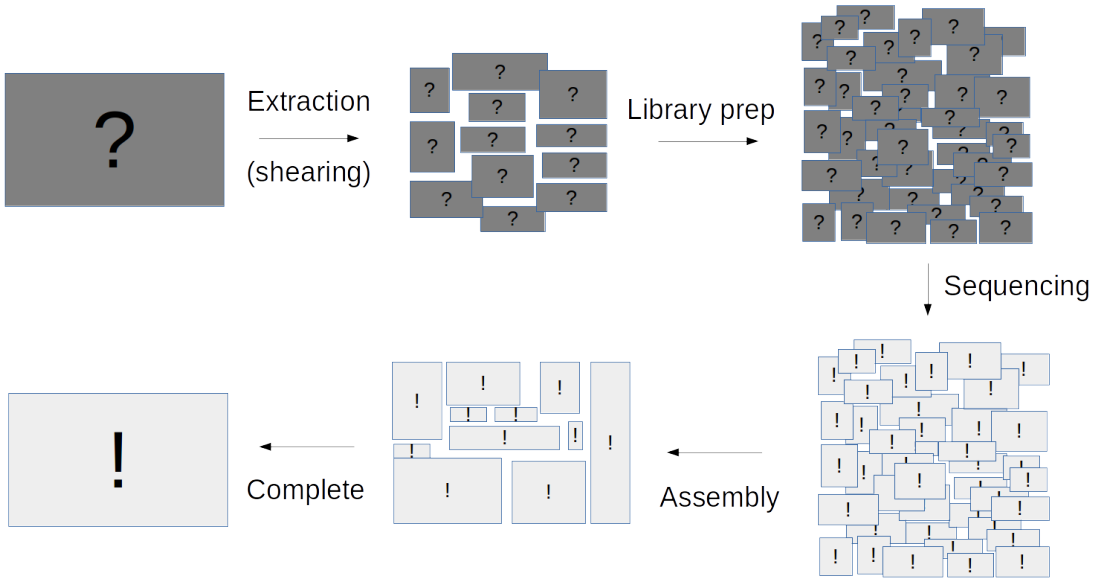


FIGURE 1.5: Basic framework for genome decoding process.

that could be efficiently read by an appropriate instrument. Before sequencing, the extracted DNA sample is normally sheared , *e.g.* by restriction enzymes, into short fragments before being subjected to a cloning step, usually Polymerase Chain Reaction or PCR, that generates a large amount of copies for DNA molecules [51]. The rationale is to have enough coverage to compensate the stochastic errors which are inevitable during library preparation and sequencing process. Those amplified pieces are then glued back together in a process called *assembly* to have a draft genome prior to applications of additional post-processing steps for the final complete reconstruction. These include but not limit to scaffolding, gaps filling, genome polishing and circularizing if applicable. The last two tasks are carried out with computational tools and in many cases, being referred together as in one whole common mission known as *genome assembly*.

Among all the steps involved in WGS, genome assembly undoubtedly plays a critical role in DNA sequencing since it determines how complete and accurate the picture of whole genome is.

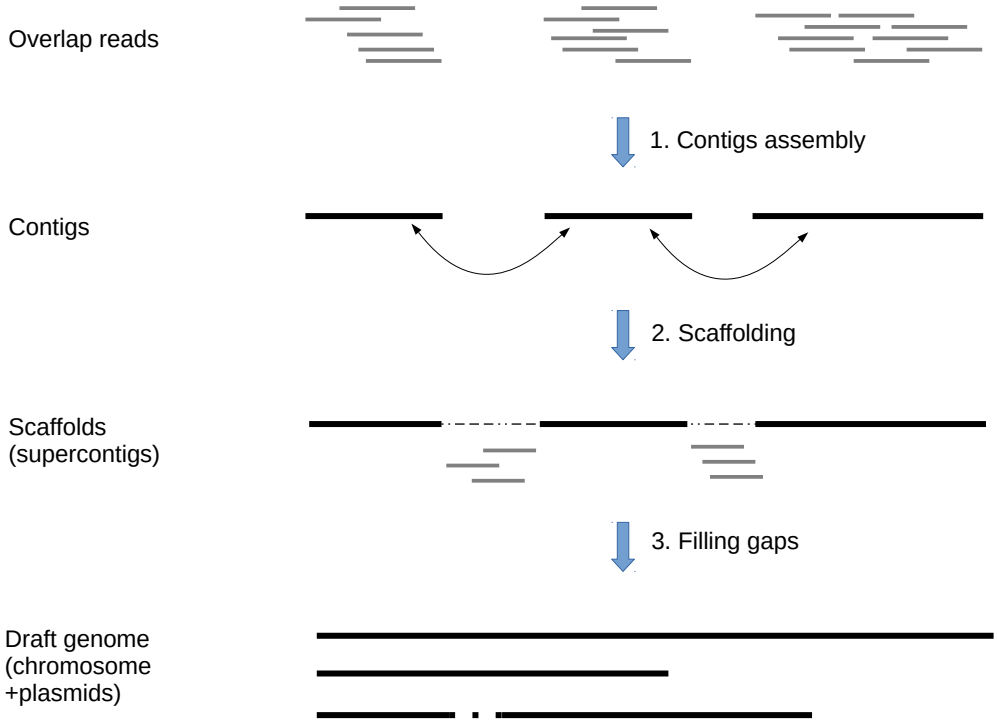


Figure 1.6: General assembly pipeline for short-read data.

1.3.2 General working principle for genome assembly

A typical short-read assembly workflow would include three stages as shown in Figure 1.6. Firstly, the overlapping reads are merged together making up longer, un-gapped sequences named as contigs. Usually this is the most challenging and time-consuming part of the whole process, especially when the the number of reads is extremely large. The next step is to connect the fragmented contigs further by taking advantage of linking information from large insert reads *e.g.* paired-end or mate pair data. The results are structures known as super-contigs or scaffolds that may contain estimated gaps. The final stage is to carefully fill those spaces by appropriate independent reads. The scaffolding and gap filling steps can be invoked repeatedly until no more improvement can be made [52–54]. The final result is a draft genome which consists of the longest possible stretch of sequences that can be induced from the data and algorithm. Unfortunately, it is common to not having the complete genome after this due to the repetitive nature of the DNA sequences. In fact, the fraction of

fully completed genomes available in public databases such as GenBank (National Center for Biotechnology Information or NCBI) or European Bioinformatics Institute (EMBL-EBI), is comparatively small. Most of the assemblies are only near-completed, marked as contig- or scaffold-level, meaning that they are still fragmented and/or gap-bearing and require more work and data to become finished.

1.3.3 Overview of assembly algorithms for SGS data

The demand for robust assemblers became overly compelling when high-throughput sequencing platforms were employed and started generating massive amount of data in the age of SGS. The abundance of randomly distributed short reads from SGS platforms had motivated substantial research in bioinformatics toward solving the genome assembly problem in an efficient way[55]. In general, approaches can be divided into two groups: greedy and graph-based approaches which are reviewed by [56, 57]. Greedy algorithms [58–60] simply assemble all the reads by iteratively joining the two with largest overlap to form the shortest common string, or *supersequence*. It is noteworthy that greedy approaches implicitly use an overlapping graph as the data structure for this purpose. Clearly, the obtained result will be locally optimal and not necessarily the full solution to the problem. However, tools of this category usually produce a relatively good approximation [61].

The graph-based methods utilize data structure of vertices and edges to represent the whole set of reads and all of their potential linkage properties. By traversing through the graph, one can define an assembly as an ordering reads and by searching for the best paths, find candidates for the optimal solution. There are two types of graph in common use, namely Overlap–Layout–Consensus (OLC) [62] and de Bruijin graph (DBG) [63, 64].

Overlap–Layout–Consensus The classical OLC algorithms follow the following three steps: overlap finding, layout forming and consensus calling. Throughout the process, an *overlapping* graph is created which has vertices as reads and edges are represented by overlapped pairs of reads which have been detected based on alignments in the first step. A traversal algorithm is then applied to find the paths of ordered and oriented vertices, also known as the *layout*. The consensus sequences are called and output based on the layout

detected. Among all the steps, finding overlap between all pairs of reads is the most computationally intensive. Although indexing techniques, such as FM index [65], have been applied to reduce the time [66], the first step still remains the major bottleneck in the whole process, especially with the burst of short read data yield introduced by recent Illumina platforms. In that circumstance, the DBG has come to be used as an efficient data structure to deal with this abundance of data.

De Bruijin Graph To construct the graph, reads are broken into consecutive *k-mers* (words of length k) for the set of vertices, whilst edges show adjacencies between *k-mers* with $k-1$ overlap. The data structure is actually storing fixed-length words and their linkages, making its memory requirement dependent only on the genome size but not data size. The graph is then targeted to errors removal based on *k-mers* spectrum, as well as trimming and simplification in which non-branched paths are merged together. At this point, the assembly problem can be reformulated as finding an Eulerian path which defined as a walk visiting each all edges on the simplified graph exactly once. The complexity of this algorithm is dependent on properties of the *k-mers* rather than the number of reads. With a smaller k, the more details of overlaps emerge in the graph, but the trade-offs would be a bigger search space due to increasing number of nodes and edges. The reason for additional connections comes from the raise of repetitive elements not covered by the word length, resulting in more branches to traverse through the graph. Currently, there are several prominent DBG-based assemblers such as Velvet [67], SPAdes [68] and ABySS [69]. All of these work efficiently on Illumina HiSeq or MiSeq output with reasonable resources and running time requirements. In term of small genomes, such as bacterial genomes, which are the focus of this thesis, SPAdes is reported to be the most suitable tool amongst the three [70].

Figure 1.7 gives an illustration of the graph construction by applying either the OLC or DBG approach. A random string containing repetitive elements is given as the reference, from which reads are sampled as output from a short-read sequencer working on this imaginary genome. Using the OLC approach, a graph is built with reads as vertices and edges reflecting overlapping properties between pairs of them. Each time a new read is generated, there is one more vertex being added to the graph and by pairwise alignments to every other

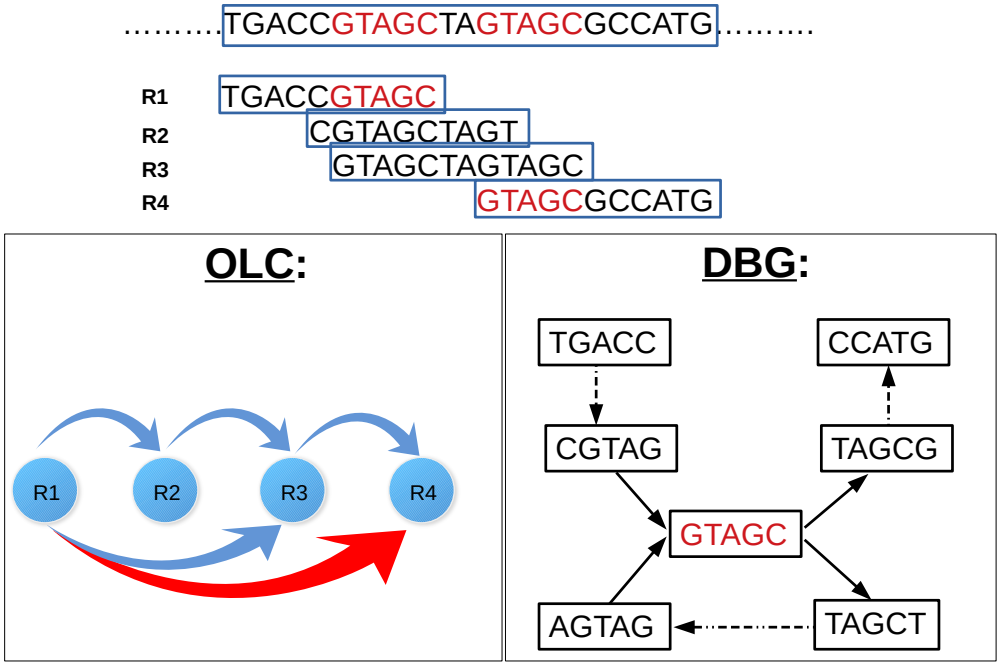


FIGURE 1.7: A simple example about building OLC and DBG graph from a string with repeat (highlighted in red). The repeat introduces ambiguous alignment resulting in a redundant link (highlighted) in the overlap graph. It also de-linearizes the DBG graph by introducing a branched node corresponding to the repetitive *k-mer*.

vertices, new edges can be added properly. To reduce size of the layout graph as the dataset is growing, the algorithm usually tries to remove contained nodes and redundant edges, *e.g.* $R2$ in the example since it is covered by $R1$ and $R3$ together. The repetitive elements cause additional faulty alignment, *e.g.* the red link from $R1$ to $R4$, making it more complicated to traverse the graph in the correct sequence.

On the other hand, DBG approach first breaks reads into set of consecutive *k-mer*s ($k = 5$ in this case). Unlike OLC, no pairwise alignments need to be explicitly carried out. As the consequence, the DBG graph is more efficient in term of construction, however, would be more challenging to solve compared to the reduced OLC graph. The reason stems from the fact that the integrity of reads are usually lost after being broken into smaller pieces as k must be strictly less than the read length. At the same time, the repeat nature of original sequence would split the OLC nodes into extra branches compared to DBG counterparts, making it more fragmented and difficult to traverse. It is worth to mention the importance

of parameter k for the DBG performance. As aforementioned, k should be chosen as large as possible to restraint the loss of information and the complexity of the resulting graph. However the graph would suffer the risk of disjointed components, or dead-ends, if $k-mers$ becomes too long so that one word would less likely overlap with another by $k-1$. From the example, we chose $k = 5$ as the best option since the DBG with $k = 6$ would fail to represent the overlap between R3 and R4.

As mentioned earlier, genome assembly from SGS data usually suffers from the presence of repeats, leading to generation of highly fragmented draft genomes (also known as pre-assemblies). This limits identification of structural variation, or identification of the configuration and location of mobile genetic elements. One theoretical solution is to have reads that exceed the 'golden threshold' of 7000 nucleotides to span over the majority of repeats in prokaryotes [71], thus able to connect fragmented contigs and render continuous assemblies out of them. In fact resolving repeats has remained a bottle-neck until the recent emergence of the so-called Third Generation Sequencing technologies with the introduction of two instruments PacBio RS and Oxford Nanopore MinION that were specially designed for long-read sequencing. Thanks to these sequencers, DNA molecules with continuous length of up to tens of kilo base-pairs or even more could be sequenced in full. As a result, long-read data have been widely exploited to finish genomes [72, 73] or identify mobile genetic elements of interest [74, 75], despite having a higher base-calling error rate. A class of assemblers using these new instrument's output has been developed, which I will describe in the following section.

1.4 Genome assembly with long read data

Nanopore data can be used on its own for *de novo* genome assembly, or in conjunction with high quality SGS data in a process called hybrid assembly. Both approaches have their own pros and cons and should be considered in reference to available dataset and the ultimate purposes of using the final assembly.

1.4.1 Long-read only assemblers

Non-hybrid assembly tools, as stated above, follow either OLC and/or an adjusted DBG approach, described below. Accordingly, most of current OLC algorithms rely on a hierarchical approach [76]. From this perspective, seeds are first chosen top-down as a subset of longest reads that must have good base-called quality and not contain each others. The remaining reads of shorter length are then aligned to the seeds for error-correction before invoking assembly and post-processing as usual.

Hierarchical Genome Assembly Process (HGAP) [77] was one of the first hierarchical OLC tool available for PacBio data but a more well-known assembler in this category is *Canu*, a renovated version of *Celera Assembler* [78] that was designed to work in particular with noisy long sequences. Canu employs a typical OLC approach with an adaptive overlapping strategy and sparse assembly graph construction to be able to build a correct overlap graph out of error-prone data [79]. Canu is able to generate decent final assembly for large eukaryotic genomes but usually requires a lengthy running time. FALCON [80] is another assembler applicable to big genomes, more than that, its string graph is designed to be diploid-aware. Another efficient OLC implementation is to apply a fast overlap finding algorithm first for the assembly skeleton, then use another consensus calling or polishing tool on the draft assembly to improve the quality. This idea has been integrated in SMARTdenovo (`https://github.com/ruanjue/smartdenovo`) or in a pipeline that combines 2 separately developed modules: *miniasm* [44] as the rapid draft constructor and *Racon* [50] as the consensus caller.

Regarding to other class, *Flye* [81, 82] and *wtdbg* (`https://github.com/ruanjue/wtdbg2`) are two representatives of long-read assemblers that used adaptive versions of *de Bruijin* graph. The first method introduces the definition of ABruijin graphs that only take a subset of "solid strings" instead of all fixed length words decomposed from reads. The solid strings are chosen based on the *k-mer* spectrum from which their fidelity are examined. Unlike the traditional DBG methods, a fast dynamic programming algorithm is used to evaluate the overlaps of various lengths between read pairs based on their longest common subpaths. The draft assembly is constructed by adding vertices of those overlapped reads before undergoing a polishing step by aligning it to the original reads. The tool wtdbg (now wtdbg2) on the

other hand, builds a fuzzy de Bruijin graph for genome assembly. This graph is similar to the original DBG but allows mismatches and indels so that approximate *k-mer*s can be collapsed into one node of the graph. Furthermore, the read paths are kept during decomposition and collapsing. After layout construction with the fuzzy DBG, a POA-like consensus step is invoked to generate the final assembly. The tool is reportedly able to quickly finish an assembly but consumes more memory than an original DBG approach due to the new graph structure.

For exclusive long reads assembly, the quality of assembly is in general lesser than SGS counterparts, even after a polishing step. As the consequence, additional polishing steps with Illumina data is desired for higher accuracy in *e.g.* for SNP callings projects. Another approach is to utilize the erroneous long reads as the linker to connect the incomplete SGS assemblies [83, 84]. Assemblers from this class usually require less long read coverage. Moreover, bridging operations are much less resource-consuming than aforementioned approaches. I expand on these hybrid assemblers in the next section.

1.4.2 Hybrid assemblers

Hybrid assembly using long reads is an economical and efficient approach to retrieve more complete genomes from the drafts since it requires less runs of the currently expensive third-generation sequencing. One of the first algorithms that takes advantage of long reads for hybrid assembly is Cerulean [83]. By aligning the contig sequences from short read assembly to the long reads, it tries to build a *skeleton graph* of long contigs connected together as the backbone and later accompany smaller contigs iteratively to the backbone to improve the assembly quality. This tool provides an efficient way of finishing the draft genomes in the sense of running time and memory usage. Likewise, in term of assembling with long-read data, SSPACE-LongRead [84] stands out as a widely-used software for this particular task. Briefly, the method relies on the BLASR [38] alignments between long reads and the pre-assemblies which are normally contigs resulted from short-read assemblers. Based on the linkages from the best alignments, a placement of these contigs into super-scaffolds is established. The scaffolds will undergone a post-processing step of final linearization and

circularization before being used as the assembly output. This approach provided a cost-effective reconstruction of bacterial genomes by using two libraries: one Illumina MiSeq or Roche-454 paired-end reads and one 50-folds long reads data from PacBio. Although long reads from other sources, such as MinION, can also be used by these methods [85], it usually requires extra works on parameter calibration.

In addition, another scaffolding algorithm, LINKS [86], uses a *k-mer* approach to make use of long reads properties rather than an aligner to connect the draft genomes. This method succeeded in improving the contiguity of ABySS genome assemblies and could adapt to a certain scale of eukaryotic genome. However, the performance highly depends on the data quality so that for early-staged chemistry of Nanopore sequencing, only 2D reads screening and/or application of error-correction utilities are recommended beforehand. In ligth of that, beside several dedicated genome scaffolding tools for MinION data as above, users can always utilize the traditional approach of using error-free short reads to correct the nanopore reads at base level. After that, the corrected long reads can be assembled by a conventional OLC algorithm in the next step. Nanocorr [34] and NaS [35] are two representatives for this approach. Normally, these assemblers could provide assemblies of high quality per base but with much more expensive computations in exchange.

Finally, another appropriate software for this purpose is Unicycler [87] which has been developed as the state-of-the-art. This application is in fact a set of tools including a combination of short-read assembly optimization, long-read only assembly, hybrid assembly, consensus calling and other post-processing steps for microbial genome assembly. It is designed to work with either short-read or long-read data but focuses on the hybrid algorithm that take advantage of both type of datasets. Its hybrid assembly module works in a likewise non-interactive (batch) mode on the whole bulk of input data, traverses the input assembly graph and returns a exhaustively polished assembly as the result. This method has been proved to generate high quality, very close-to-complete genome assemblies thanks to exhaustive computational steps but the running time, as a trade-off, is relatively long and is only efficient for microbial genomes. More details about this method will be provided later in Chapter 4 as a relevant content.

1.5 Real-time analysis

1.5.1 Definition

In circumstances of this thesis, the adjective "real-time" indicates a high level of responsiveness of an analysis system in updating its environment status corresponding to inputs fed into the system [88]. The time gap from getting new input to the point of status being updated, or *deadlines* [89], is not necessary to be as close to zero as possible (immediate), but in reality a positive value limited by a reasonable upper-bound (nearly immediate). The upper-bound is normally defined based on human temporal sensation *e.g.* within minutes.

Briefly, real-time data analysis is the operation applied on the data in a prompt time interval to provide near-instantaneous output. In contrast, a data processing system is known to work in "batch mode" if it collects all the input data before any any operation is applied. The result obtained from a batch analysis on the bulk of data is final and static, meaning it cannot be updated using another batch of data without rerunning the whole process. Real-time applications usually involve a specific communications method namely *data stream*. Streaming data is characterized by its continuously migration from a *source* to a *sink*, which are normally the consecutive modules in a real-time pipeline. To implement instant response programs is a challenging task but in return, grants certain amount of advantages over traditional batch counterparts [90]. This will be discussed later throughout the content of this thesis.

1.5.2 Real-time analysis for nanopore sequencing

One of the novel aspects of ONT's sequencing platforms is that the sequence data of each molecule is written to disk as soon as it is generated (with up to 2000 molecules being sequenced in parallel, each molecule progressing through the pore at 450bp per second). This is unlike SGS sequencing-by-synthesis platforms, which sequence billions of short reads in parallel, with each cycle (contributing an extra base) taking several minutes, with the data being provided in batch at the end of a run.

For this task, the raw data of a read must be retrieved and analyzed while sequencing is

still in progress. This offers the opportunity to obtain analysis results as soon as sufficient data are generated, upon which sequencing can be terminated or used for other experiments. As the consequences, answers to the related genomic questions could be obtained *in situ*, in an automated manner that saves considerable amount of time and resources compared to the conventional approaches. Furthermore, streaming analysis can benefit from avoiding under- and over-sequencing which could result in either the generation of more sequence data than is required at greater cost, or a low quality assembly if insufficient data are generated.

Several systems incorporating real-time feature of MinION data have been developed e.g. the cloud based platform Metrichor (Oxford Nanopore), work by Quick *et al.* [91] and MetaPORE [92], focusing on phylogenetic analysis of a sample. Importantly, it is worth noting the selective sequencing protocol, *i.e.*"*Read Until*", which had been proposed and implemented [93] exclusively for nanopore sequencing, motivated by the idea that only DNA molecules of interest should be sequenced. In this proposal, the process could intervene the transition of DNA through the pore by reversing the pore bias to eject the one deemed as non-informative. To achieve this, an examination is carried out for each molecule by reading the real-time squiggle data from current changes caused by the transition and comparing this signal to a reference. This real-time feedback system would be important for target enrichment and background depletion sequencing [94].

On the other hand, as a member one of the very first groups having access to MinION sequencing, I have been developing an in-house tool set to analyze the data for specific studies, focusing in streaming analysis on microbial genomics. The framework had been implemented with utilities ranging from initial analysis to handful of further identification processes such as species typing, strain typing and investigating antibiotic resistance profiles on microbes [95]. In such pipelines, `npReader` [33] continuously scans the folder containing sequencing data in parallel with the MinION sequencing. It picks up base-called sequences as soon as they are generated, and a stream of reads is created to feed the appropriate pipeline for further identification analyses. Different modules of a workflow can communicate to each other via the network sockets or inter-process redirection pipes provided by Unix-like operating systems. In general, each module takes a stream of data of interest (*e.g.* a read, an alignment) as input and carries out its task every time a given amount of data or

waiting interval has passed. As a response, only relevant information is extracted to retain or forward to another module following the analysis. This data processing methodology can return results on-the-go and at the same time, only engages small memory footprint which is a clear benefit when working with large amounts of data, over long period of running experiment. Our tools are proved to be helpful in reducing the turn-around time for the clinical analysis and heading toward rapid diagnostic usage of MinION in real-life applications [96].

However, there still exists a gap for a method that could scaffold and finish assemblies in a real-time fashion. This has become the motivation for this thesis project which will focus on using nanopore data in the scaffolding and annotating feature of `npScarf`, demultiplexing algorithm from `npBarcode`, as well as assembly graph resolving in `npGraph` - all working in real-time.

1.6 Summary

To sum up, for the time being, there are various DNA sequencing platforms available in the market, each of them embraces its own methodology and outputs different reads in term of length, accuracy, throughput and sequencing cost. Hence the choice for a sequencing method depends on the requirement from individual project and sometimes, users need to use several sequencers together on the same sample to obtain adequate data of interest. Amongst the sequencing options, ONT nanopore sequencing with MinION stands out as a remarkably portable and deployable long-read sequencer that offers real-time access to the sequencing output. The continuous upgrades of its flow cell and sequencing kits are indeed improving the quality and quantity of the long-read data, making this sequencing method a very welcomed additional player in the field.

MinION data has been shown to be a useful source of genomic data for various studies thanks to its long-spanning and real-time accessible features. In the midst of those, genome assembly and structural annotation are amongst the most common applications since this technology can bypass the fragmentation caused by repetitive elements that are difficult to resolve using the short-read sequencing. At the same time, the on-time accessibility of the

data opens up the opportunity to establish analyses in real-time, thus offering interactive management on resources during sequencing process.

1.7 Thesis aims

Inspired by the unique functionality of nanopore sequencing technology, this thesis project aims to develop real-time applications using MinION data, focusing on genome assembly pipelines for microbial isolates. The purpose is to have further complete assemblies, as much as possible, that would facilitate downstream annotations.

Goals setting Initially, the project targets to implement a streaming pipeline to rapidly finish short-read assemblies using real-time MinION sequencing. It is aimed to join fragmented contigs with least input data as possible while at the same time, still able to producing high quality and complete sequences. The ability to run and to report completion status in real-time allows users to decide when to stop the sequencing prematurely thus open the possibility of saving time and cost for genome analyses. In addition, hybrid assembly algorithm is expected to work on the assembly graph of the short-read assemblies to further improve the sensitivity and completeness of the final result. It would also provide users appropriate visualization for better interaction with the assembly pipeline in real-time.

On the other hand, it is desired to extend aforementioned pipeline for multiple samples at the same time, through a paralleling mechanism known as barcoded sequencing. This technology is made available from ONT, via Barcoding kits, to allow pooling and sequencing of multiple libraries on the sample flow cell, which further enhances the versatility of the technology. The underlying mechanism is to ligate a unique oligonucleotide sequence, or *barcode*, to the fragments of each DNA sample. Multiple samples can then be pooled together and sequenced in one flow cell. After that, the sequenced reads are demultiplexed into bins by examining the barcode portions on the reads. For this purpose, a streaming demultiplexer for barcode sequencing is developed. The tool brings to MinION practitioners a flexible option to monitor a barcoded sequencing run as well as to integrate pooled sequencing into a streaming analysis pipeline. Together with in-house software developments, their

applications into real-life use cases are taken place. This would give better understanding about software performances as well as specifications on different datasets.

Contributions This thesis resulted in several real-time processing modules that are able to assist genome assembly and annotation, including `npScarf` [97] `npBarcode`[98] and `npGraph`. All of these individual add-on modules were wrapped in a bigger framework that was specifically designed for nanopore data analysis, hosted in `https://github.com/mdcao/japsa`. The main contribution of this thesis regarding genome scaffolding and assembly can be found in a separate repository `https://github.com/hsnguyen/assembly` for the convenience of development and maintenance.

The source code of the whole project is made available and applied in numerous use cases internally as well as from community. Feedback from users and external data sources are treated with care to further improve the performance and functionality of the developing software.

Restrictions The thesis mostly focuses on applications in microbial genomics as this is a clear use case for the applicability of real-time analysis particularly in diagnosis and identification of antibiotic resistance strains. However, the methodologies developed in this thesis can be applied more broadly, and have been demonstrated on yeast genomes. Applications to metagenomics and eukaryotic genomes were attempted but not thoroughly studied thus not included in the scope of in this thesis.

Another limitation is that the developed assemblers are hybrid approaches, meaning an additional SGS run is required before using the tools. An exclusive long-read assembler is difficult to adapt to a real-time pipeline due to its intensive computation for the error-correction stage which usually requires bulk of data and heavy resources consumption.

1.8 Thesis outline

The main research chapters of this thesis are organized as follows.

Chapter 2 : This chapter presents the implementation of the very first pipeline for finishing genomes in real-time using MinION long reads and side applications through `npScarf`.

Chapter 3 : This chapter describes demultiplexing barcoded nanopore sequencing data with `npBarcode` in real-time and its application. In addition, there is another use case of using different assembly strategies to finish four XDR *K. pneumoniae* strains.

Chapter 4 : This chapter shows how to integrate assembly graph into the available pipeline from `npScarf` to improve the assembly quality. This work results in `npGraph` and relate applications.

Chapter 5 : This chapter introduces another computational analysis of MinION sequencing data for small circular genome assembly, such as for virus, bacterial plasmids. Data for two *Caulimovirids* samples are shown to demonstrate the performance of proposed methods.

2

Streaming assembly using Nanopore reads

The speed of decision making is the essence of good governance.

–Piyush Goyal

This chapter describes a unique streaming assembly strategy on MinION data that has been implemented in a tool named `npScarf`. Its methodology allows completing the assembly in real-time together with the nanopore sequencing nature. Several experiments on different datasets are conducted for better illustrations of `npScarf` performance and functionality. Comparisons with other methods indicate better quality from the results while at the same time, consume less resources and running time.

These research findings have been published in a journal manuscript under the digital object identifier (DOI):`10.1038/ncomms14515`. The manuscript will be reproduced with minor changes and reformatting for the whole content of this chapter. The Supplementary materials for this paper is shown in Appendix A.

As the co-first author, I am a primary contributor to the project in term of conception and design, software development, data analysis as well as drafting the first version of the manuscript. Specifically, I have predominantly designed the streaming pipeline and implemented the algorithm as the main developer of `npScarf`. I have also contributed significantly to the analysis and interpretation of the research data. Finally, I am the author of the first draft versions of the manuscript on which the publication is based.

Scaffolding and completing genome assemblies in real-time with nanopore sequencing

Minh Duc Cao[1,*,+], **Son Hoang Nguyen**[1,+], **Devika Ganesamoorthy**[1], **Alysha G. Elliott**[1], **Matthew A. Cooper**[1] and **Lachlan J.M. Coin**[1,*]

[1]Institute for Molecular Bioscience, University of Queensland, St Lucia, Brisbane, QLD 4072 Australia

[*]Correspondence: m.cao1@uq.edu.au and l.coin@imb.uq.edu.au

[+]These authors contributed equally to this work

Received 11 Jul 2016. Accepted 6 Jan 2017. Published 20 Feb 2017

PMID: 28218240 DOI: 10.1038/ncomms14515

Abstract

Third generation sequencing technologies provide the opportunity to improve genome assemblies by generating long reads spanning most repeat sequences. However, current analysis methods require substantial amounts of sequence data and computational resources to overcome the high error rates. Furthermore, they can only perform analysis after sequencing has completed, resulting in either over-sequencing, or in a low quality assembly due to under-sequencing. Here we present `npScarf`, which can scaffold and complete short read assemblies while the long read sequencing run is in progress. It reports assembly metrics in real-time so the sequencing run can be terminated once an assembly of sufficient quality is obtained. In assembling four bacterial and one eukaryotic genomes, we show that `npScarf` can construct more complete and accurate assemblies while requiring less sequencing data and computational resources than existing methods. Our approach offers a time- and resource-effective strategy for completing short read assemblies.

2.1 Introduction

High-throughput sequencing technology has transformed genomics research over the last decade with the ability to sequence the whole genome of virtually any organism on the planet. Most sequencing projects to date employ short read technology and hence cannot unambiguously resolve the repetitive sequences that are present abundantly in most genomes. As a result, assemblies are fragmented into large numbers of contigs and the positions of repeat sequences in the genome cannot be determined. These repeat sequences often play important biological roles; for example, they mediate the lateral transfer of genes between bacterial species via pathogenicity islands and plasmids. Analyzing these regions is thus essential to determine key characteristics such as antimicrobial resistance (AMR) or to identify highly pathogenic variants of many bacterial species [75].

Long read sequencing technologies, for example Pacific Biosciences' (PacBio) and Oxford Nanopore MinION sequencing, allow users to generate reads spanning most repetitive sequences, which can be used to close gaps in fragmented assemblies. A key innovation of the MinION nanopore sequencing device is that it measures the changes in electrical current as a single-stranded molecule of DNA passes through the nanopore and uses the signal to determine the nucleotide sequence of the DNA strand [99–101]. As such, the raw data of a read can be retrieved and analyzed as soon as it is generated, while sequencing of other reads is still in progress. This offers the opportunity to obtain analysis results as soon as sufficient data are generated, upon which sequencing can be terminated or used for other experiments.

Several algorithms have been developed to utilize long reads for genome assembly. *de novo* assemblers such as the hierarchical genome assembly process (HGAP) [77] and nanocorrect/nanopolish [36] can assemble a complete bacterial genome using only long read sequencing data. However, because of the high error rates in these sequencing technologies, this *de novo* approach requires substantial amounts of sequencing data and extensive computational resources, mainly for polishing the genome assembly. Hybrid assemblers, which combine error-prone long reads with highly accurate and cheaper short read sequence data, provide a more economical and efficient alternative for building complete genomes. They

can be classified into three categories. *de novo* methods such as Canu [46] and miniasm/racon pipeline [50] employ fast approximate approaches to assemble a skeleton of the genome using long reads. The skeleton, often as erroneous as the raw reads, is then polished with high quality short reads. On the other hand, tools in the *error-correction* category (*e.g.*, PBcR [23], Nanocorr [34] and NaS [35]) correct long reads with high quality short reads before assembling the genome with the corrected long reads. Finally, the *scaffolding* methods (SPAdes-hybrid [68, 75], SSPACE-LongRead [73, 84] and LINKS [86]) use long reads to scaffold and fill in gaps in the assemblies from short read sequencing.

While these tools are reported to assemble high quality bacterial genomes [102, 103], they have not made use of the real-time sequencing potential of the MinION; assembly of a genome can only be performed after the sequencing is complete. This can lead to over-sequencing, which incurs extra cost and time; or under-sequencing resulting in a low-quality assembly. Here, we present `npScarf`, the first hybrid assembler that can scaffold and complete fragmented short read assemblies with sequence data streaming from the MinION while sequencing is still in progress. In effect, `npScarf` can fully utilize a sequence read within minutes of it being generated. Furthermore, it continuously reports assembly quality during the experiment so that users can terminate the sequencing when an assembly of sufficient quality and completeness is obtained. We show that our method can generate more accurate and more complete genomes than existing tools, while requiring less nanopore sequencing data and computational resources. As such, `npScarf` can be used to efficiently control MinION sequencing in completing existing short read assemblies and in hybrid assembly projects. More importantly, `npScarf` can facilitate the real-time analysis of positioning genomic sequences for time critical applications such as in AMR investigation. We show that the `npScarf` can rapidly and accurately reconstruct genomic islands carrying AMR genes that are fragmented in short read assemblies. It can also identify AMR genes encoded in plasmids. These are among the main analyses to understand the acquisition of AMR in pathogenic bacteria.

2.2 Results

2.2.1 Algorithm overview

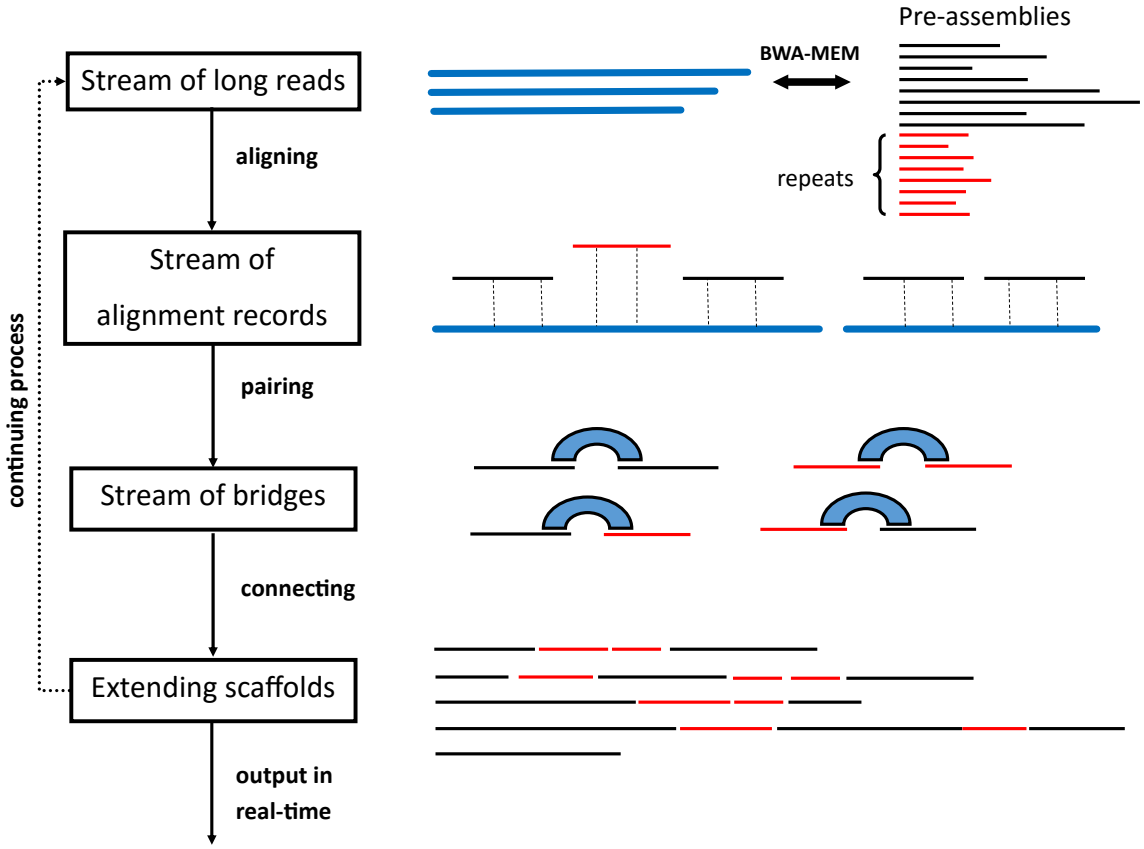


FIGURE 2.1: Workflow of the real-time algorithm. Stream of long reads are aligned to the existing contigs to create alignment records. Bridges connecting contigs are formed, and are used for extending scaffolds. These steps are performed in a streaming fashion.

The genomes of most organisms contain an abundance of repeat sequences that are longer than the read length limit (300 bps) of Illumina sequencing platforms [104]. In assembling a genome using this technology, these repeat sequences cannot be distinguished and hence are often collapsed into contigs, leaving gaps in the genome assembly. To complete the assembly, `npScarf` first determines the multiplicity of each contig, thereby identifying contigs

representing non-repetitive sequences (called unique contigs). It then scaffolds and fills in gaps in the assembly in a streaming fashion (Figure 2.1). Upon receiving a long read from the MinION, `npScarf` immediately aligns it to the unique contigs. Reads aligned to two unique contigs form a bridge connecting the two contigs. Gradually, the unique contigs are joined to form the scaffold of the genome, while the repetitive contigs are used to fill in the gaps in the scaffold. The details of the algorithms are presented in Methods.

2.2.2 Completing bacterial assemblies

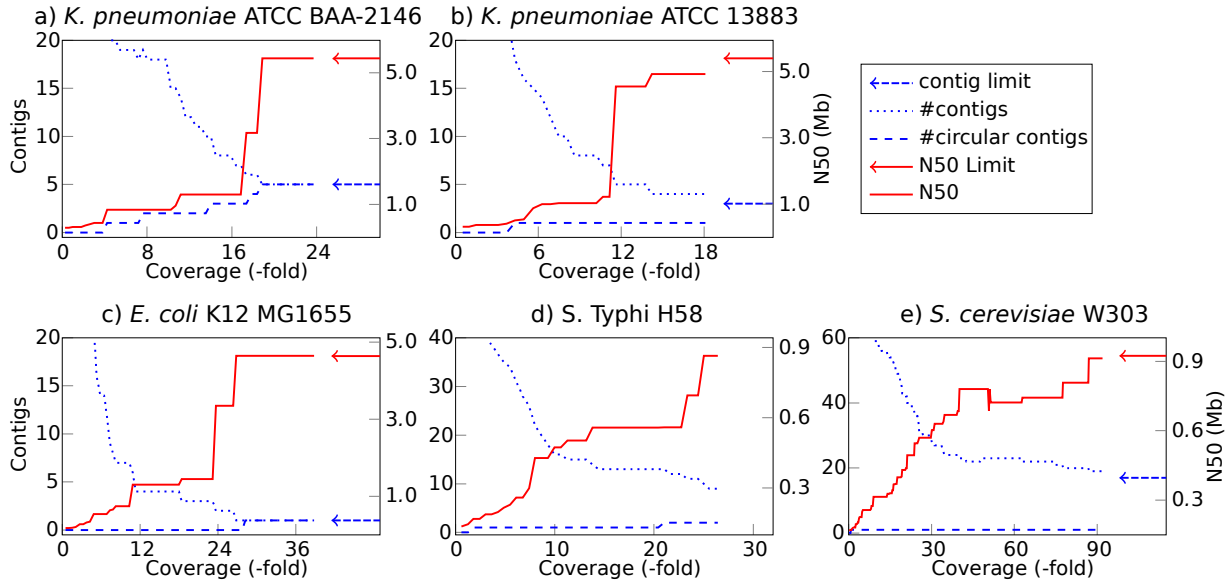


Figure 2.2: Assembly statistics during real-time scaffolding. The plots show N50 statistics, number of contigs, and number of circular contigs against the amount of nanopore sequencing data.

We assessed the performance of our algorithm for its ability to scaffold and complete the Illumina assemblies of two bacterial *Klebsiella pneumoniae* strains, ATCC BAA-2146 (New Delhi metallo-beta-lactamase (NDM-1) positive) and ATCC 13883 (type strain). We first sequenced the genomes of these strains with the Illumina MiSeq platform to 250-fold coverage and assembled them with SPAdes [68] (See Methods). This resulted in assemblies of 90 and 69 contigs that were 500bp or longer, respectively. The N50 statistics of the two assemblies were 288Kbp and 302Kbp, respectively. We then sequenced the two strains

TABLE 2.1: Comparison between `npScarf`'s assemblies and the reference genomes of two *K. pneumoniae* strains

`npScarf` assemblies			Reference sequences		
Name	**Size (bp)**	**Plasmid *ORI***	**Accession**	**Size (bp)**	**Plasmid *ORI***
K. pneumoniae ATCC BAA-2146					
Contig 1*	5,437,518	-	CP006659.1*	5,435,369	-
Contig 2*	141,026	IncA/C2	CP006661.1*	140,825	IncA/C2
Contig 3*	118,278	IncFIB(K); IncFII(K)	CP006663.1*	117,755	IncFIB(K); IncFII(K)
Contig 4*	85,233	IncR; IncFIA(HII)	CP006662.1*	85,164	IncR; IncFIA(HII)
Contig 5*	2,015	ColRNAI	CP006660.1*	2,014	ColRNAI
K. pneumoniae ATCC 13883					
Contig 1	4,923,970	-	KN046818.1	5,284,261	-
Contig 2	372,214	-			
Contig 3	139,480	IncFIA(HII); IncFIB(K)	KN046820.1	95,930	IncFIA(HII); IncFIB(K)
			KN046821.1	42,420	-
Contig 4*	119,388	ColRNAI; IncFII(pCoo); pSM22	KN046819.1	106,842	IncFII(pCoo); pSM22
			KN046822.1	16,331	-

*Circular sequences.

with Oxford Nanopore MinION using chemistry R7. For ATCC BAA-2146, we obtained 185Mbp of sequencing data (∼33-fold coverage of the genome), of which 27Mbp were two-directional (2D) reads. The run for strain ATCC 13883 yielded only 13.5Mbp of sequencing data (∼2.4-fold coverage). We re-sequenced this strain with the improved chemistry R7.3. By combining sequencing data from both experiments for this strain, we obtained a total of 100Mbp (∼18-fold coverage) data, including 22.5Mbp of 2D reads. The quality of the data, described in [95], was broadly similar to that reported by other MinION users [75, 105, 106].

As the pipeline described here was developed after we performed the MinION sequencing runs, we tested our streaming analysis by re-running the base-calling using the Metrichor service. Sequence reads in fast5 format were written to disk, and were instantaneously picked up and streamed to the pipeline by npReader [33]. In essence, the scaffolding pipeline received sequence data in fastq format in a streaming fashion as if a MinION run was in progress. During analysis, the pipeline continuously reported the assemblies' statistics (the numbers of contigs and the N50 statistic), allowing us to track the completeness of the assembly, as well as the number of circular sequences in the genome. This is especially important for the

analysis of bacterial genomes where chromosomes and plasmids are usually circular. To validate the resulting assemblies, we compared them with the reference genomes of these strains obtained from NCBI (GenBank Accessions GCA_000364385.2 and GCA_000742135.1). We also ascertained the predicted plasmids in these assemblies by looking for the existence of plasmid origins of replication sequences from the PlasmidFinder database [107].

Figure 2.2a) and 2.2b) present the progress of assembly completion against the coverage of MinION data during scaffolding. As expected, N50 statistics increased and the number of contigs decreased with more MinION data. For *K. pneumoniae* ATCC BAA-2146, we found that our algorithm required only 20-fold coverage of sequence data ($<$ 120Mbp) to complete the genome, reducing the assembly to the limit of five contigs (one chromosome and four plasmids). Those five contigs were circularised, indicating completeness. We found these five contigs to be in total agreement with the complete genome assembly of the strain, previously sequenced with PacBio and Illumina [74] (See Table 2.1 and Supplementary Fig. 1).

With 18-fold coverage of the MinION data for *K. pneumoniae* ATCC 13883, the assembly was improved to four contigs, in which one was reported to be circular (Contig 4). These contigs were aligned to the reference genome for this strain, which contained 16 contigs in five scaffolds. We found Contig 1 and Contig 2 from the `npScarf`'s assembly were aligned to the reference scaffold KN046818.1, while Contig 3 and Contig 4 were aligned to two reference scaffolds (See Table 2.1 and Supplementary Fig. 2). The alignments contained forward and reverse matches. The breakpoints of these matches corresponded to the contig joints in the reference scaffolds, indicating the incorrect orientation of contigs in the reference scaffolds. The reference scaffold KN046818.1 was 5.2Mbp in size, suggesting this scaffold was the chromosome and was fragmented into two contigs in the `npScarf`'s assembly. In examining this chromosomal sequence, we found the two contigs to be separated by an rRNA operon of 7Kbp in length. BLAST search revealed the structure of this operon with rRNA 5S, 23S and 16S as the main components. This rRNA operon sequence was also found to be present at five other loci in the genome, which were all resolved. However, no long MinION read was found to align to this particular position, possibly because of the low yield of this dataset, which caused the chromosome sequence to be fragmented. We anticipate this

could be resolved with more nanopore sequencing data. Contig 3 (139Kbp) and Contig 4 (119Kbp) contained several origin of replication sequences (See Table 2.1), suggesting they were plasmid sequences; Contig 4 was also reported to be a circular sequence. In Contig 4, we noticed an extra plasmid origin of replication sequence (ColRNAI) that was not found in the reference genomes (see Table 2.1). In examining the position of ColRNAI, we found it was in one of the gaps in the reference scaffold, hence not reported in the reference assembly.

2.2.3 Real-time analysis for positional information

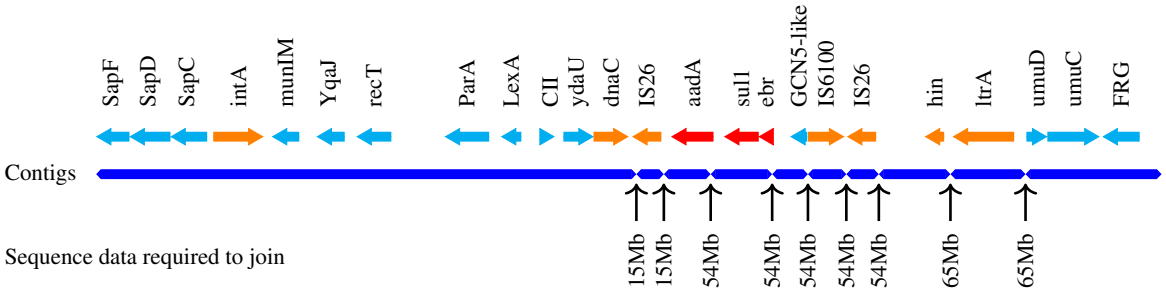


FIGURE 2.3: Structure of a pathogenic island from *K. pneumoniae* ATCC BAA-2146. The island harbours three antibiotic resistance genes *strep*, *sul1* and *ebr*, flanked by mobility genes integrase (*int*), inverstase (*hin*), DNA replication (*dnaC*), and insertion sequences (IS26 and IS6100). The island was fragmented into 10 contigs in the Illumina assembly, and was completely resolved with 65Mbp out of the total of 185Mbp of nanopore sequence data.

The ability to complete genome assemblies in streaming fashion also enables real-time analyses that rely on positional information. Such analyses include identifying genes encoded in bacterial genomic islands and plasmids. These functional regions in the bacterial genomes can be horizontally transferred between organisms, which is one of the main mechanisms for acquiring AMR in pathogenic bacteria. Here we demonstrate these analyses on the multi-drug resistant *K. pneumoniae* ATCC BAA-2146 strain.

Prior to scaffolding the Illumina assembly of the sample, we annotated the assembly using Prokka [108] to identify the positions of genes and insertion sequences in the assembly. Bacterial genomic islands are genomic regions longer than 8Kbp, containing certain

Table 2.2: Timeline of determining plasmid-encoded antibiotic resistance genes

Data required	Gene ID	NCBI ref	Antibiotic resistance	Plasmid evidence
10Mbp	blaTEM-1B	JF910132	penicillins, some cephalosporins	IncR;IncFIA(HI1)
	strB	M96392	streptomycin	IncR;IncFIA(HI1)
	strA	AF321551	streptomycin	IncR;IncFIA(HI1)
	sul2	GQ421466	sulfonamides	IncR;IncFIA(HI1)
14Mbp	aac6Ib	M21682	tobramycin, amikacin, netilmicin, sisomicin	IncR;IncFIA(HI1)
21Mbp	mphA	D16251	erythromycin	IncFIB(K);IncFII(K)
	tetA	AJ517790	tetracyclines	IncFIB(K);IncFII(K)
	QnrB7	EU043311	quinolones	IncR;IncFIA(HI1)
29Mbp	dfrA14	DQ388123	trimethoprim	IncR;IncFIA(HI1)
46Mbp	blaNDM-1	FN396876	penicillins, cephalosporins, carbapenems	IncA/C2
51Mbp	rmtC	AB194779	aminoglycosides (include gentamicin, kanamycin)	IncA/C2
78Mbp	sul1	AY224185	sulphonamide	IncA/C2
	aac6Ib_1	M21682	tobramycin, amikacin, netilmicin, sisomicin	IncA/C2
	blaCMY-6	AJ011293	penicillins, some cephalosporins	IncA/C2
83Mbp	blaSHV-11	GQ407109	penicillins, some cephalosporins	IncR;IncFIA(HI1)
	aac6Ib	M21682	tobramycin, amikacin, netilmicin, sisomicin	IncR;IncFIA(HI1)
	blaOXA-1	J02967	penicillins	IncR;IncFIA(HI1)
	aac3-IIa	X51534	gentamicin, tobramycin, netilmicin, sisomicin	IncR;IncFIA(HI1)

classes of genes such as AMR genes. In addition, they often carry mobility genes such as transposase, integrase and insertion sequences (IS) [109]. These sequences generally appear multiple times in the genomes (repetitive sequences), causing genomic islands fragmented in the short read assembly. We ran Islander [110] and PHAST [111] on the Illumina assembly, which together detected six genomic islands. In the annotation, we also found 28 insertion sequences; 14 of these were within 3Kbp of the contig ends, suggesting that any genomic islands flanked by these insertion sequences were fragmented. During scaffolding of the assembly with nanopore sequencing data, `npScarf` constructed four additional genomic islands, which were not previously reported by Islander and PHAST (data not shown). All 10 genomic islands were precisely in agreement with the analysis of the PacBio assembly by [74]. Figure 2.3 presents the structure of such a genomic island, namely Kpn23SapB, and the timeline of its reconstruction. The genomic island harboured three AMR genes,

namely *aadA* (mediates resistance to streptomycin and spectinomycin), *sul1* (sulfonamides) and *ebr* (ethidium bromide and quaternary ammonium). The genomic island also carried two copies of the insertion sequence IS26, which flanked the AMR genes, and a copy of the insertion sequence IS6100. The presence of these repetitive sequences caused the island to be fragmented into 10 contigs in the Illumina assembly; the three resistance genes were in two different contigs. `npScarf` required 64.59Mbp of data (14-fold coverage of the genome) to report the full structure of the island (Figure 2.3).

For real-time detection of plasmid-encoded genes, we identified plasmid origin of replication sequences from the Illumina assembly using the PlasmidFinder database [107]. Contigs containing a plasmid origin of replication sequence were considered to be part of a plasmid. Essentially, only 166 genes contained within these contigs could be ascertained as plasmid-encoded genes from the Illumina sequencing of the *K. pneumoniae* ATCC BAA-2146 strain. During scaffolding of the Illumina assembly, once a contig was added to a plasmid, `npScarf` reported genes in the contig as plasmid-encode genes. The amount of long-read sequence data required to assign each gene to a plasmid is presented in the Supplementary Spreadsheet.

With the Illumina assembly, we identified 27 AMR genes, but none was in a contig containing a plasmid origin replication sequence. As such, whether any of these genes were carried by a plasmid could only be ascertained with long reads. Table 2.2 presents the time-line of such determination. In particular, we confirmed 18 AMR genes as plasmid-encoded with 83Mbp (∼14-fold coverage) of nanopore sequencing data. In addition, as all four plasmids were circularized and complete with 103Mbp (∼18-fold coverage) of data, we could confidently conclude that only these 18 AMR genes were plasmid-encoded, even before the completion of the full genome assembly and the sequencing run.

2.2.4 Comparison with other methods

Table 2.3: Comparison of assemblies produced by `npScarf` and the comparative methods

Method	Assembly size (Mbp)	#Contigs (≥ 500bp)	N50 (Kp)	Mis-assemblies	Error (per 100Kbp)	Run times (CPU hrs)	
K. pneumoniae ATCC BAA-2146. Nanopore data: 33X coverage							
SPAdes	5.70	90	288	0	4.72	15.63	
SPAdes-Hybrid	5.75	17	3,076	1	6.61	16.07	
SPAdes+SSPACE	5.74	53	400	4	12.73	15.63 +	2.3
SPAdes+LINK	5.74	31	554	5	16.05	15.63 +	4.03
SPAdes+`npScarf` (rt)	5.78	5	5,438	0	20.00	15.63 +	1.6
SPAdes+`npScarf` (b)	5.78	5	5,438	0	22.76	15.63 +	0.84
NaS+CA	5.89	29	345	15	18.89	324.35 +	3.49
Nanocorr+CA	5.68	68	139	8	141.32	312.64 +	1.37
Canu+Pilon	0	-	-	-	-	-	-
Miniasm+Pilon	0	-	-	-	-	-	-
K. pneumoniae ATCC 13883. Nanopore data: 18X coverage							
SPAdes	5.51	69	302	5	6.22	16.95	
SPAdes-Hybrid	5.54	15	729	19	8.02	16.97	
SPAdes+SSPACE	5.55	36	685	13	12.39	16.95 +	1.48
SPAdes+LINK	5.55	17	1,527	18	16.12	16.95 +	1.12
SPAdes+`npScarf` (rt)	5.55	4	4,924	21	10.84	16.95 +	0.52
SPAdes+`npScarf` (b)	5.55	4	4,924	21	10.26	16.95 +	0.45
NaS+CA	5.46	38	394	36	10.24	192.78 +	6.92
Nanocorr+CA	5.02	60	148	16	118.34	161.33 +	2.6
Canu+Pilon	0.04	4	12	4	10.40	0.53 +	0.46
Miniasm+Pilon	0.03	3	13	1	14.12	0.00 +	0.26
E. coli K12 MG1655. Nanopore data: 67X coverage							
SPAdes	4.61	114	176	0	3.51	4.38	
SPAdes-Hybrid	4.67	42	4,643	2	1.21	4.76	
SPAdes+SSPACE	4.66	59	3,155	1	29.26	4.38 +	3.42
SPAdes+LINK	4.66	50	3,318	2	36.19	4.38 +	4.03
SPAdes+`npScarf` (rt)	4.64	1	4,644	2	13.08	4.38 +	2.43
SPAdes+`npScarf` (b)	4.64	1	4,646	2	11.72	4.38 +	1.91
NaS+CA	4.87	21	874	19	10.60	807.19 +	6.77

Continued on next page

Table 2.3 – continued from previous page

Method	Assembly size (Mbp)	#Contigs (≥ 500bp)	N50 (Kp)	Mis-assemblies	Error (per 100Kbp)	Run times (CPU hrs)
Nanocorr+CA	4.66	2	4,650	6	10.41	213.68 + 8.49
Canu+Pilon	0.11	9	14	0	13.90	0.79 + 0.28
Miniasm+Pilon	1.91	85	23	1	595.61	0.04 + 1.24
S. Typhi H58. Nanopore data: 26X coverage						
SPAdes	4.84	89	107	7	39.05	1.86
SPAdes-Hybrid	4.88	27	443	12	55.46	2.06
SPAdes+SSPACE	4.88	34	358	10	59.39	1.86 + 1.55
SPAdes+LINK	4.86	20	473	13	66.65	1.86 + 1.28
SPAdes+`npScarf` (rt)	4.87	9	864	18	53.86	1.86 + 0.93
SPAdes+`npScarf` (b)	4.86	8	864	16	52.01	1.86 + 0.47
NaS+CA	4.97	54	212	17	58.87	248.32 + 7.21
Nanocorr+CA	2.98	95	37	9	973.63	199.85 + 0.94
Canu+Pilon	0	-	-	-	-	- -
Miniasm+Pilon	0.02	2	14	0	10.96	0.01 + 0.26
S. cerevisiae W303. Nanopore data: 196X coverage						
SPAdes	11.82	364	155	29	124.10	20.54
SPAdes-Hybrid	12.06	240	346	68	158.13	67.81
SPAdes+SSPACE	13.39	263	392	89	136.66	20.54 + 31.54
SPAdes+LINK	12.09	161	580	83	143.04	20.54 + 26.97
SPAdes+`npScarf` (rt)	12.00	19	913	82	141.93	20.54 + 21.28
SPAdes+`npScarf` (b)	11.90	17	924	79	141.01	20.54 + 18.84
NaS+CA	12.76	121	155	123	140.08	9811.88 + 140.69
Nanocorr+CA	13.48	108	600	133	197.00	7208.08 + 272.86
Canu+Pilon	12.31	43	497	81	229.08	599.36 + 58.5
Miniasm+Pilon	11.79	51	391	41	1400.82	0.27 + 30.27

We compared the performance of our algorithm against existing methods that were reported to build assemblies with nanopore sequencing. In addition to the two samples presented above, we sourced three other samples reported in the literature including *i.*) an *Escherichia coli* K12 MG1655 strain sequenced to 67-fold coverage with a nanopore R7.3 flowcell and

standard library preparation [112]; *ii.*) a *Salmonella enterica* serovar Typhi (S. Typhi) haplotype, H58 [75] sequenced to 27-fold and *iii.*) a *Saccharomyces cerevisiae* W303 genome (196-fold) [34]. Note that the coverage reported here was from all base-called data (including both 1D and 2D reads). Of the methods selected for comparison, SPAdes-hybrid [68], SSPACE-LongRead [84], LINKS [86] and `npScarf` were scaffolders, whereas Nanocorr [34] and NaS [35] belonged to the error correction category. We assembled the Illumina data of these samples using SPAdes [68] before running the scaffolding methods with nanopore data. SPAdes-hybrid was run by incorporating nanopore data into the assembly (with –nanopore option). The two error correction tools Nanocorr and NaS were run on the nanopore sequencing data using about 50-fold coverage of Illumina data, as suggested by authors of the respective publications. The corrected reads were then assembled using Celera Assembler [78]. We observed that the quality of the assemblies produced by Celera Assembler were highly sensitive to the parameters specified in the specification file. We therefore ran Celera Assembler for each dataset on three specification files provided by the authors of NaS and Nanocorr, and report here the most complete assembly obtained. We also ran two popular *de novo* assembly methods, Canu [46] and Miniasm [44] on these datasets. These methods necessitated a polishing step using Pilon [113].

We evaluated the assemblies in terms both of completeness and accuracy. The completeness of an assembly was assessed by N50 statistics and the number of contigs that were longer than 500 bp. To examine the accuracy of an assembly, we compared it with the closest reference genome of the samples in NCBI (See Methods) to obtain the number of misassemblies, mismatches and short indels. During the test, we recorded the CPU times required by these pipelines to produce the assemblies. Run times for the scaffolder methods included times for running SPAdes and for scaffolding, while those for NaS and Nanocorr included correction time and Celera Assembler time. The times reported for the *de novo* methods included that for polishing using Pilon. Table 2.3 presents the comparison metrics of all assemblies as reported by Quast [114], as well as their run times.

We ran `npScarf` in real-time mode, in which nanopore sequencing data are streamed to the pipeline in the exact order they were generated. This allowed us to assess the completeness of the assemblies against the amount of data generated. Figure 2.2 shows the progress of

completing the assemblies for all five samples. As mentioned previously, `npScarf` produced complete and near-complete assemblies for the two *K. pneumoniae* samples (Figures 2.2a and 2.2b) with under 20-fold coverage of nanopore data. For the *E. coli* K12 MG1655 sample, `npScarf` required less than 30-fold coverage of nanopore data to complete the genome assembly with one circular contig. `npScarf` also reduced the *S. Typhi* assembly to only nine contigs (N50=864Kbp), which was significantly better than the assembly reported by [75] from the same data (34 contigs, N50=319Kbp).

As for the *S. cerevisiae* W303 genome, which contains 16 nuclear chromosomes and one mitochondrial chromosome, `npScarf` generated an assembly of 19 contigs (N50=913Kbp); substantially fewer than the 108 contigs (N50=600Kbp) generated by the next best method (Nanocorr, see Table 2.3). We noticed a drop in N50 statistics at the point where about 50-fold coverage of nanopore data were received (Figure 2.2e). This was because `npScarf` encountered contradicting bridges and hence broke the assembly at the lowest scoring bridge in lieu of a higher scoring one. The N50 was then improved to reach the N50 of 913Kbp with 90-fold coverage of nanopore sequencing; the assembly did not change with more data (90-fold to 196-fold). We examined the assembly by comparing that with the reference genome of *S. cerevisiae* strain S288C. One of the contigs (Contig 17, length = 81Kbp) was reported to be circular, which was completely aligned to the mitochondrial chromosome of the reference genome. Ten chromosomes (II, IV, V, VII, IX, X, XI, XIII, XV and XVI) were completely assembled into individual contigs, and three chromosomes (I, III and VIII) were assembled into two contigs per chromosome (See Supplementary Figure 3). We found a misassembly that joined chromosome IV and the start of chromosome XIV into Contig 10. The end of chromosome XIV was also joined with chromosome XII into Contig 2. These misassemblies essentially fused these three chromosomes into two contigs. We found these misassemblies were due to the presence of interspersed repeat elements which are known for being problematic in assembly analysis [104]. The assemblies produced by Canu and Miniasm also presented several misassemblies fusing different chromosomes together, emphasizing the challenges posed by interspersed repeats in assembling complex genomes (See Supplementary Figures 4 and 5).

We reran `npScarf` on the datasets in batch mode, in which the scaffolding was performed

with the complete dataset. We found that all five assemblies were more complete than in real-time mode. In particular, the *S. cerevisiae* W303 assembly was further reduced to 17 contigs as chromosomes I and VIII were resolved into individual contigs (data not shown). In this assembly, 12 out of 17 chromosomes were completely recovered to one contig, one chromosome (XIII) was fragmented into two contigs and three chromosomes were fused into two contigs due to misassemblies

In all datasets, `npScarf` consistently produced the most complete assemblies, while its accuracy was among the best. It was the only method that was able to completely resolve the *K. pneumoniae* ATCC BAA-2146 genome (five contigs, N50 of 5.4Mbp) with no misassembly, requiring only 20-fold coverage of nanopore data; the second most completed assembly (produced by SPAdes Hybrid) contained 17 contigs and had the N50 of only 3.1Mbp despite using 33-fold coverage of nanopore sequence data. On the well studied *E. coli* K12 MG1655 strain sample where LINK, NaS and Nanocorr were reported to resolve the whole genome with a larger dataset (147-fold coverage) [86], none of these methods could produce the same result on the 67-fold coverage data set we tested. On the other hand, `npScarf` was able to reconstruct the genome into one circular contig with as little as 30-fold coverage of the data. On the S. Typhi dataset, `npScarf` produced assemblies with nine contigs in real-time mode, and with eight contigs in batch mode (N50=864Kbp), significantly better than assemblies from other methods (over 20 contigs). We observed that the *de novo* methods, Canu and Miniasm failed to construct a skeleton for the these bacterial genomes (either no output or only a few small sequences produced), possibly due to the low coverage of these datasets.

The *S. cerevisiae* W303 assembly produced by `npScarf` was near complete and N50 statistics reached the theoretical limit of 924Kbp. Note that `npScarf` obtained these results from only less than half of the dataset (95-fold coverage). On the whole dataset (196-fold coverage), Canu and Miniasm produced assemblies of 43 contigs (N50=496Kbp) and 51 contigs (N50=391Kbp), respectively. These assemblies contained more than twice as many contigs as the results from `npScarf`. The second most complete assembly in terms of N50 statistic was produced by Nanocorr (N50=600Kbp) which was significantly lower than that from `npScarf`.

We observed that the scaffolding methods and the *de novo* methods were much faster

than the error correction counterparts. Both NaS and Nanocorr required the alignment of the short reads to the long reads, which were computationally expensive. On the other hand, the scaffolding pipelines required 20 CPU hours or less to build an assembly from short reads, and from a few hours to around 30 hours to scaffold the assembly with long reads. As Canu and Miniasm did not produce a decent assembly for the bacterial data sets, we only include them in the comparison for the *S. cerevisiae* dataset. Miniasm was the fastest on this dataset, requiring only 0.27 CPU-hours to assemble and over 30 CPU-hours to polish the genome. Apart from SPAdes-Hybrid, which performed scaffolding as part of short read assembly, `npScarf` was the fastest among other scaffolders and consistently required less scaffolding time. Note that the times reported in Table 2.3 were for processing the entire nanopore dataset, whereas `npScarf` could be terminated early once a desirable assembly was obtained. We observed that `npScarf` required only 2GB of memory for scaffolding the bacterial datasets, and 4GB for the *S. cerevisiae* dataset, which can be easily installed on a laptop computer. A summary of memory usage of other tools was presented in Supplementary Table 1.

2.3 Discussion

The development of high-throughput long read sequencing technologies such as PacBio and nanopore has opened up opportunities for resolving repetitive sequences to assemble complete genomes and to improve existing genome assemblies. However, the relatively high error rates of these technologies pose a challenge to the accurate assembly of genome sequences. An obvious solution is to combine long and erroneous reads with more accurate and cheaper short read data for assembling genomes [23, 72]. One such approach is to perform *de novo* assembly of long reads to generate a skeleton of the genome, and error correct the skeleton with accurate short reads [44, 46]. Alternatively, erroneous long reads are corrected [23, 34, 35, 72] before being assembled with classical assemblers designed for long and accurate reads such as Celera Assembler [78]. These approaches usually require large amounts of long read data. Hybrid assemblers in the scaffolding class harness long spanning reads to guide the extension of contigs in the draft genome assemblies. For example, SSPACE-LongRead [84]

and Cerulean [83] rely on the alignment of long reads to the assembly graph to determine the adjacent contigs. LINKS [86] uses a k-mer approach, which further improves the running time with a small sacrifice of accuracy. Overall, hybrid-assembly methods, especially those in the scaffolding category, provide economical genome finishing pipelines that can produce high quality genome assemblies from small amounts of long read data on modest computing equipment. `npScarf` is similar to these mentioned scaffolders in the sense that it aligns the long reads to the contigs to build a scaffold of the genome. However, our method estimates the copy number of each contig in the genome and constructs the scaffold from non-repetitive contigs, while the repetitive contigs are used to fill the gaps in the scaffold. Consequently, we demonstrated that `npScarf` is capable of generating more complete and accurate assemblies than the competitors, while requiring much less data.

To date, there is no prominent assembler that takes advantage of the real-time feature from nanopore sequencing. Nanopore technology allows one to terminate a run and wash the flowcell for subsequent runs without compromising sequencing yield and quality. The ability to analyse data on the fly and to stop a sequencing run when sufficient data are generated plays a critical role to control resources necessary for a single experiment.

One of the main contributions of our algorithm is that it can process data streaming from the sequencer and report the current status of the analysis in real-time. Our pipeline still relies on a base-caller, and the introduction of fast real-time base-callers such as Nanocall [115] and DeepNano [116] helps to reduce the latency. The current pipeline processes a sequence read within minutes of it finishing traversing the pore, rather than as the read is actually passing through the pore, and as such is real-time at the temporal resolution of minutes, but not at the millisecond level required to update with the addition of each base. However, this temporal resolution is sufficient to allow our pipeline to answer the biological problems at hand at the earliest possible time, and while sequencing is still in progress. Investigators can also assess the progress of the analysis, and terminate the sequencing once an assembly of sufficient quality and completeness is obtained. This enables the generation of sufficient data necessary for the analysis to guarantee the experimental outcomes and, at the same time, avoids costly over-sequencing. While our pipeline still requires short read data which cannot be generated in real-time with current technology, it offers a strategy to minimise

the generation of the more expensive long read data.

The real-time function to complete genomic sequences opens the possibility of *in situ* biological analyses [95]. Certain biological markers of interest may be identified from short read assemblies, but their positions in the genome could only be determined by completing the genome assembly with long reads. We have shown that `npScarf` can facilitate such analyses in real-time by demonstrating the identification of AMR genes encoded in plasmids and pathogenicity islands.

2.4 Methods

2.4.1 Determining unique contigs

Before scaffolding a fragmented short read genome assembly, `npScarf` determines the multiplicity of each contig in the assembly by comparing short read sequencing coverage of the contig to that of the whole genome. Coverage information is often included in the sequences assembled by most tools, such as SPAdes [68] and Velvet [67], or can otherwise be obtained from the mapping of short reads to the assembly. An reasonable estimate for depth coverage of the genome is that of the largest contig. `npScarf` however leverages this to the *normalized average coverage* of the largest contigs so long as their depth coverage does not deviate from the estimated genome depth coverage. More formally, let $depth_i$ and len_i respectively represent the sequencing depth (coverage) and the length of contig i, where contigs are sorted in decreasing order in length. Let $depth_g$ represent the estimated coverage of the whole genome. `npScarf` first initializes $depth_g$ to that of the largest contig:

$$depth_g^1 = depth_1 \tag{2.1}$$

It then iteratively updates the estimate

$$depth_g^i = \frac{\sum_i depth_i \times len_i}{\sum_i len_i} \tag{2.2}$$

and terminates the process when the depth coverage of the next contig greater than a threshold

$$\frac{depth_i}{depth_g^{i-1}} > \theta \tag{2.3}$$

npScarf set the threshold θ to 1.5. In our experience, the statistic is stable with up to 20 of the largest contigs longer than 20Kbp, which are most likely unique contigs in bacterial genomes [71]. We hence also add these into the condition for termination. The multiplicity of contig i (mul_i) is determined by

$$mul_i = \frac{depth_i}{depth_g} \tag{2.4}$$

npScarf considers a contig unique if its multiplicity is less than θ.

2.4.2 Bridging unique contigs and filling gaps with repetitive contigs

npScarf next builds the backbone of the genome from the unique contigs. It identifies the long reads that are aligned to two unique contigs, thereby establishing the relative position (*i.e.*, distance and orientation) of these contigs. To minimize the effect of false positives that can arise from aligning noisy long reads, npScarf groups reads that consistently support a particular relative position into a bridge and assigns the bridge a score based on the number of supporting reads and the alignment quality of these reads. When two unique contigs are connected by a bridge, they are merged into one larger unique contig. npScarf uses a greedy strategy based on Kruskal's algorithm [117], which merges contigs from the highest scoring bridges. In the newly created contig, the gap is temporarily filled with the consensus sequence of the reads forming the bridge. npScarf then identifies repetitive contigs that are aligned to this consensus sequence, and uses these contigs to fill in the gap.

2.4.3 Real-time processing

To support real-time analysis of nanopore sequencing, the previously described algorithm can be augmented to process long read data directly from a stream (See Figure 2.1). In this mode, npScarf employs a mapping method that supports streaming processing such as BWA-MEM [41] to align a small number of long reads to the existing assembly as they arrive. This block-wise processing allows npScarf to make use of information from a small batch of reads sequenced within a short period of time (within minutes). If a read is aligned

to two unique contigs, it is added to the bridge connecting the two contigs. Once the bridge reaches a pre-defined scoring threshold, the two contigs are merged and the gap is filled as above. In case this merging contradicts with the existing assembly (for example, if the relative distance and/or orientation implied by the bridge is inconsistent with those of previously used bridges) `npScarf` revisits the previous bridges, breaks the smallest scoring contradicting bridge and uses the current bridge instead. The algorithm hence gradually improves the completeness and the quality of the assembly as more data are received.

2.4.4 Bacterial cultures and DNA extraction

Bacterial strains *K. pneumoniae* ATCC BAA-2146 ((NDM-1 positive) and ATCC 13883 (type strain) were obtained from American Type Culture Collection (ATCC, USA). Bacterial cultures were grown overnight from a single colony at 37 °C with shaking (180 rpm). Whole cell DNA was extracted from the cultures using the DNeasy Blood and Tissue Kit (QIAGEN©, Cat #69504) according to the bacterial DNA extraction protocol with modified enzymatic lysis pre-treatment.

2.4.5 Illumina sequencing and assembly

Library preparation was performed using the NexteraXT DNA Sample preparation kit (Illumina), as recommended by the manufacturer. Libraries were sequenced on the MiSeq instrument (Illumina) with 300 bp paired end sequencing, to a coverage of over 250-fold.

2.4.6 MinION sequencing

Library preparation was performed using the Genomic DNA Sequencing kit (Oxford Nanopore), according to the manufacturer's instructions. For the R7 MinION Flow Cells SQK-MAP-002 sequencing kit was used and for R7.3 MinION Flow Cells SQK-MAP-003 were used, according to the manufacturer's instructions. A new MinION Flow Cell (R7 or R7.3) was used for each sequencing run. The library was loaded onto the MinION Flow Cell and the Genomic DNA 48-hour sequencing protocol was initiated using MinKNOW software.

2.4.7 Data collection

MinION data for the *E. coli* K12 MG1655 sample [105] were downloaded from the European Nucleotide Archive (ENA) with accession number ERP007108. We used the data from the chemistry R7.3 run (67-fold coverage of the genome from run accession ERR637419) rather than the chemistry R7 reported in work by [34, 35, 86]. Illumina MiSeq sequencing data for the sample were also obtained from ENA (assession number ERR654977). Data from both Illumina and MinION sequencing of the S. Typhi strain [75] were collected from ENA accession number ERP008615. The *S. cerevisiae* W303 sequencing data were provided by [34] from the website `http://schatzlab.cshl.edu/data/nanocorr/`.

2.4.8 Data processing

Read data from Illumina sequencing were trimmed with *trimmomatic* V0.32 [118] and subsequently assembled using SPAdes V3.5 [68]. SPAdes was run with the recommended parameters (-k 21,33,55,77,99,127 –careful). SPAdes-Hybrid was run with the inclusion of the –nanopore option. SSPACE and LINKS were run on the original SPAdes' assemblies. For SSPACE, we used the parameters reported to work with MinION reads in [73] (-i 70 -a 1500 -g -5000). In the case of LINKS, a script was adapted from the example run for *E. coli* K12 MG1655 sample to allow 30 iterations of the algorithms being executed for each dataset. NaS and Nanocorr were applied to correct nanopore data from the maximum of 50-fold coverage of Illumina data. The corrected long reads were assembled using Celera Assembler version 8.3 with the configuration files provided by the respective publication. Canu was run with the recommended parameter for nanopore data (-nanopore-raw). Miniasm were run with its default parameter. The short reads were aligned to Canu's and Miniasm's assemblies by BWA-MEM, and Pilon was run on the alignments to polish the assembly sequences.

The Illumina assembly of *K. pneumoniae* ATCC BAA-2146 was annotated using Prokka (version 1.12-beta) with the recommended parameters for a *K. pneumoniae* strain. AMR genes from the assembly were identified using the ResFinder database [119]. Plasmid origin of replication sequences in both *K. pneumoniae* assemblies were identified by uploading the assembly to the PlasmidFinder database [107].

In real-time analysis, `npScarf` aligned incoming long reads using BWA-MEM [41] with the parameters –k11 –W20 –r10 –A1 –B1 –O1 –E1 –L0 –a –Y –K10000 . The –K10000 parameter allowed alignments to be streamed to the scaffolding algorithm after several reads were aligned.

2.4.9 Comparative metrics

The assemblies produced by the mentioned methods were evaluated using Quast (V3.2) to compare with the respective reference sequences. The number of contigs, N50 statistics and the number of misassemblies were as per Quast reports. The error rates were computed from sum of the number of mismatches and the indel length. The CPU time for each pipeline was measured using the Linux time command (/usr/bin/time -v); the sum of user time and system time was reported. When a pipeline was distributed across a computing cluster, its CPU time was the sum of that across all jobs.

2.4.10 Data availability

Sequencing data for the two *K. pneumoniae* samples were deposited to the European Nucleotide Archive (ENA). The accession numbers for the MinION sequencing data are ERR1474979 and ERR1474981, and that for the MiSeq sequencing data are ERR1474547 and ERR1474549. The software presented in this article and its documentation is publicly available at GitHub `https://github.com/mdcao/npScarf`.

3

Multi-samples analyses with barcode sequencing

The more you read, the better you get at it.

–James Patterson

The first part of this chapter presents a demultiplex algorithm that can be applied together with a barcode Nanopore sequencing in order to analyze multiple samples simultaneously in real-time. This practice is favorable for microbial genomics when the genome size is relatively small and a single flow cell can normally host more than one isolate for a run. As the result, a combination of barcode sequencing and streaming analysis can further reduce the overuse of resources.

The concept for such real-time demultiplexer has been implemented in `npBarcode` and published in the *Bioinformatics* journal article, namely *"Real-time demultiplexing Nanopore barcoded sequencing data with* `npBarcode`*"* as shown below. Its content will be adopted with additional details for the first section of this chapter. As the first author of this paper, I am the primary designer and software developer of the project. I contributed significantly to data generation, interpretation as well as writing the first draft of the manuscript.

Later in this chapter, an use case will be addressed as a highlighted example of genome assembly practices to resolve an actual problem in research. `npScarf` with other state-of-the-art assembly methods are employed for such task. The data being presented in this section is part of another study, namely *"Evaluating the Genome and Resistome of Extensively Drug-Resistant Klebsiella pneumoniae using Native DNA and RNA Nanopore Sequencing"* which has myself as a co-author. In general, my contributions fell mostly in the bioinformatics aspects of the research, especially in data processing and genome assembly. I also helped in plotting and interpreting the final results.

The full manuscript with supplementary materials can be accessed via the preprint version from the link `https://doi.org/10.1101/482661`. The study has been submitted to *Nature Microbiology* journal and is currently under review.

Real-time demultiplexing Nanopore barcoded sequencing data with npBarcode

Son Hoang Nguyen[1,*], **Tania Duarte**[1], **Lachlan J.M. Coin**[1] and **Minh Duc Cao**[1,*]

[1]Institute for Molecular Bioscience, University of Queensland, St Lucia, Brisbane, QLD 4072 Australia
[*]Correspondence: s.nguyen@uq.edu.au and m.cao1@uq.edu.au.

Received 20 Jun 2017. Accepted 23 Aug 2017. Published 24 Aug 2017

PMID: 28961965 DOI: 10.1093/bioinformatics/btx537

Abstract

Motivation: The recently introduced barcoding protocol to Oxford Nanopore sequencing has increased the versatility of the technology. Several bioinformatics tools have been developed to demultiplex the barcoded reads, but none of them support the streaming analysis. This limits the use of pooled sequencing in real-time applications, which is one of the main advantages of the technology.

Results: We introduced `npBarcode`, an open source and cross platform tool for barcode demultiplex in streaming fashion. npBarcode can be seamlessly integrated into a streaming analysis pipeline. The tool also provides a friendly graphical user interface through `npReader`, allowing the real-time visual monitoring of the sequencing progress of barcoded samples. We show that `npBarcode` achieves comparable accuracy to the other alternatives.

Availability: `npBarcode` is bundled in Japsa - a Java tools kit for genome analysis, and is freely available at https://github.com/hsnguyen/npBarcode.

3.1 Demultiplex barcode sequencing with MinION

3.1.1 Introduction

Oxford Nanopore Technologies (ONT) sequencing has already become an established technology for its portability and its potential for high yield data generation. In particular, it offers the ability to sequence genomes in real-time where practitioners can analyze data streaming directly from the device, and can terminate a sequencing run once the satisfactory results are obtained. Recently, ONT has introduced barcode protocols to allow pooling and sequencing multiple libraries on the sample flow cell, which further enhances the versatility of the technology. The underlying mechanism is to ligate a unique oligonucleotide sequence, or *barcode*, to the fragments of each DNA sample. Multiple samples can then be pooled together and sequenced in one flow cell. The sequenced reads can then be demultiplexed into bins by examining the barcode portions on the reads.

Several outstanding tools for demultiplexing Nanopore barcoded sequences such as poreFUME [120], Porechop `https://github.com/rrwick/Porechop` and Metrichor built-in demultiplexer have been developed. Of these tools, only the latter supports real-time analysis of a sequencing run, but it is only available as a cloud service. This limits the use of this technology in time-critical applications or when the users wish to perform sequencing only until sufficient data are obtained.

Being able to exploit the streaming property of nanopore sequencing is important for time-critical applications, especially when its data generating yield and rate is escalating drastically with new technologies and platforms already or soon become available such as GridION, PromethION. Hence, algorithms that can work with streaming data and be able to integrated into a pipeline should be taken into consideration in earnest for this sequencing technology. Furthermore, a more friendlier user interface to inexperienced users is highly on demand as many more researchers worldwide are enrolled in MinION multi-sample sequencing. In addition, the visualization can provide more efficient statistical reports in real-time thus allowing users to have better control over the whole sequencing process. For that reason, we have developed another tool for ONT barcode sequencing demultiplex analysis.

Here, we present `npBarcode`, a tool for demultiplexing barcoded MinION sequencing data in real-time. `npBarcode` provides the traditional command line interface and a graphical user interface. The command line interface offers a flexible environment to be integrated in with other real-time downstream analyses, *e.g.* real-time finishing genome sequence (`npScarf` [97]) and real-time species identification (npAnalysis[95]). The demultiplexer is also integrated into `npReader`'s [33], our previously developed platform for real-time analysis and visualization of nanopore sequencing. From this mode, beside the utilities provided by `npReader`, one can visually monitor on the sequencing progress of each barcoded sample.

3.1.2 Results

3.1.2.1 Algorithm overview

`npBarcode` relies on local pairwise alignment to detect the existence of a barcode sequence in each nanopore read. We apply the Smith-Waterman algorithm with Gotoh's optimization [121] for the alignment. Basically, the aligner attempts to align the barcode sequences within a window on both ends of a nanopore read. The read is assigned to the group of the barcode with the highest alignment score, provided that the score is greater than a threshold, and is greater than the second best alignment score by a safety distance.

`npBarcode` is implemented within the Japsa toolkit as a program named `jsa.np.barcode`. The program consumes the base-called data in a streaming fashion and demultiplexes the data into different channels containing reads that belong to the same bin. These output streams can be piped to other downstream real-time analyses. This design allows practitioners to integrate the tool into a streaming analysis pipeline of interests.

3.1.2.2 Comparison with other methods

We barcoded and sequenced 8 different bacterial strains on a single MinION R9.4 flow cell using the Oxford Nanopore Technology's 2D native barcoding kit (SQK-LSK208 + EXP-NDB002). Another run with PCR barcoding kit for 3 libraries is not discussed here as part of a different study, but its result is included in Appendix Figure B.1.

The pool consisted of one gram positive isolate (*Staphylococcus aureus*) and seven gram

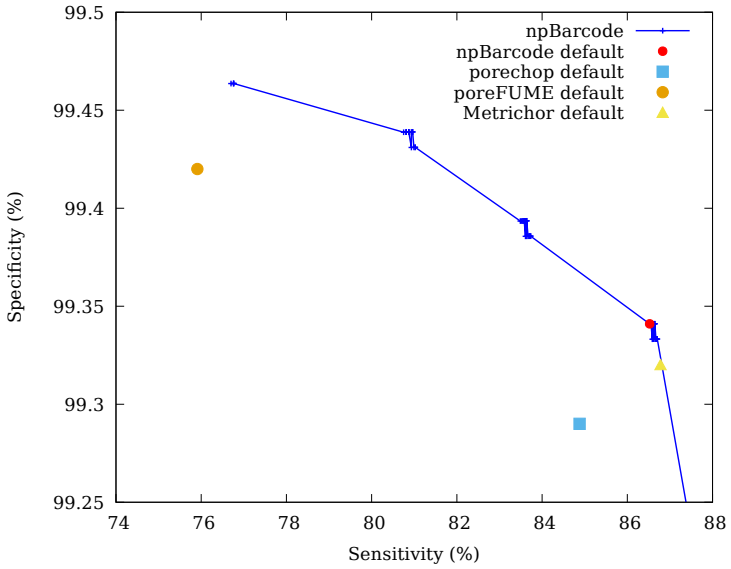
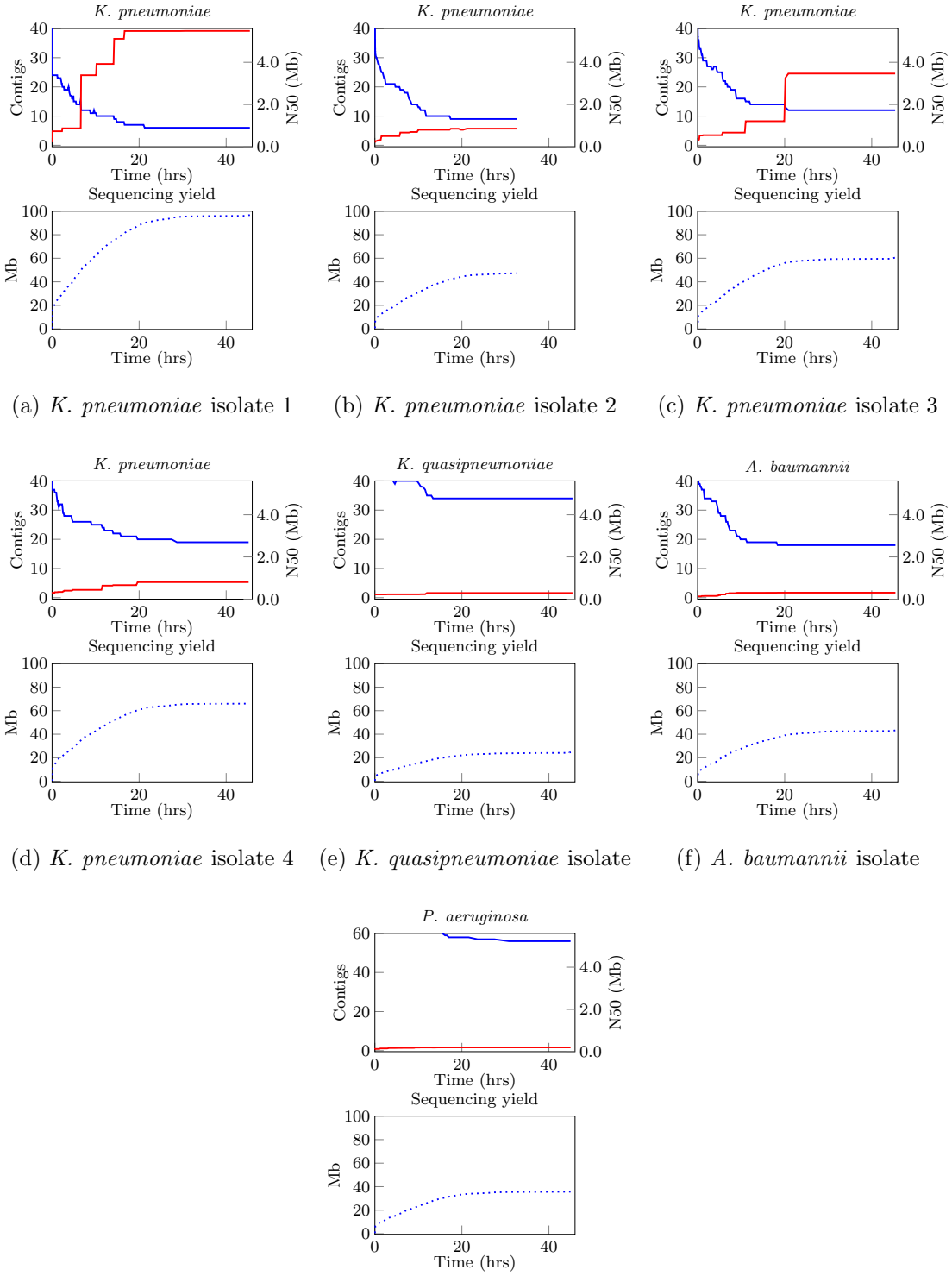


FIGURE 3.1: Plot of sensitivity versus specificity of `npBarcode` compared with existing tools.

negative isolates (four *Klebsiella pneumoniae*, one *Klebsiella quasipneumoniae*, one *Pseudomonas aeruginosa* and one *Acinetobacter baumannii*). In order to establish a ground truth benchmark for comparison of different de-multiplexing tools, we aligned the sequence reads of the eight samples to their respective assemblies to identify the original source of the reads. Due to the high level of similarity among the gram negative isolates, we could only obtain the confident assignment of reads to the *S. aureus*, a gram positive strain, versus the other gram negative strains. Hence, we set up the comparison framework as the accuracy of the tool detecting the barcode associated with the *S. aureus* sample. For an exhaustive benchmark for each of every strains included, ones can assess Appendix Figure B.2.

Figure 3.1 presents the sensitivity and specificity of `npBarcode` with differing parameters. We also obtained the results from Metrichor, porechop and poreFume with their default parameter settings. Statistics of all tools with their automatic settings is depicted in details in Appendix Table B.3. We found that `npBarcode` performed comparably with Metrichor and was marginally more accurate than the competitive porechop and poreFUME.



(a) *K. pneumoniae* isolate 1 (b) *K. pneumoniae* isolate 2 (c) *K. pneumoniae* isolate 3

(d) *K. pneumoniae* isolate 4 (e) *K. quasipneumoniae* isolate (f) *A. baumannii* isolate

(g) *P. aeruginosa* isolate

Figure 3.2: Result for running a combining pipeline of `npReader`, `npBarcode` and `npScarf` with ONT Native Barcoding Kit in order to scaffold genomes of 7 gram-negative samples in real-time. For each sample, the upper plot shows number of contigs (blue) and N50 (red) while the lower graph presents data yield (bases) for that particular demultiplexed sample over time.

3.1.2.3 Real-time scaffolding pipeline for multiple samples

As part of the experiment, we also integrated `npBarcode` into a streaming analysis pipeline where we scaffolded genome assemblies in real-time. Prior to the MinION run, we sequenced the seven gram negative samples with Illumina technology and used `SPAdes` [68] to assemble them into more than 100 contigs each. The *S. aureus* sample was used as a control sample during demultiplexing, and was not used for scaffolding. As soon as a sequence read was generated, it was base-called, demultiplexed and streamed into the appropriate instance of `npScarf` [97] for scaffolding. This pipeline allowed us to simultaneously scaffold all seven bacterial samples while the sequencing was still in progress (Figure 3.2). We were able to complete the genome of one *K. pneumoniae* isolate after 16 hours of sequencing and with about 80Mbp of nanopore sequencing data. While the genomes of the other six isolates were not completed due to their low proportions in the pooled sample, and the less than ideal yield of the run, they were significantly improved over time with nanopore long reads.

3.1.2.4 Graphical user interface

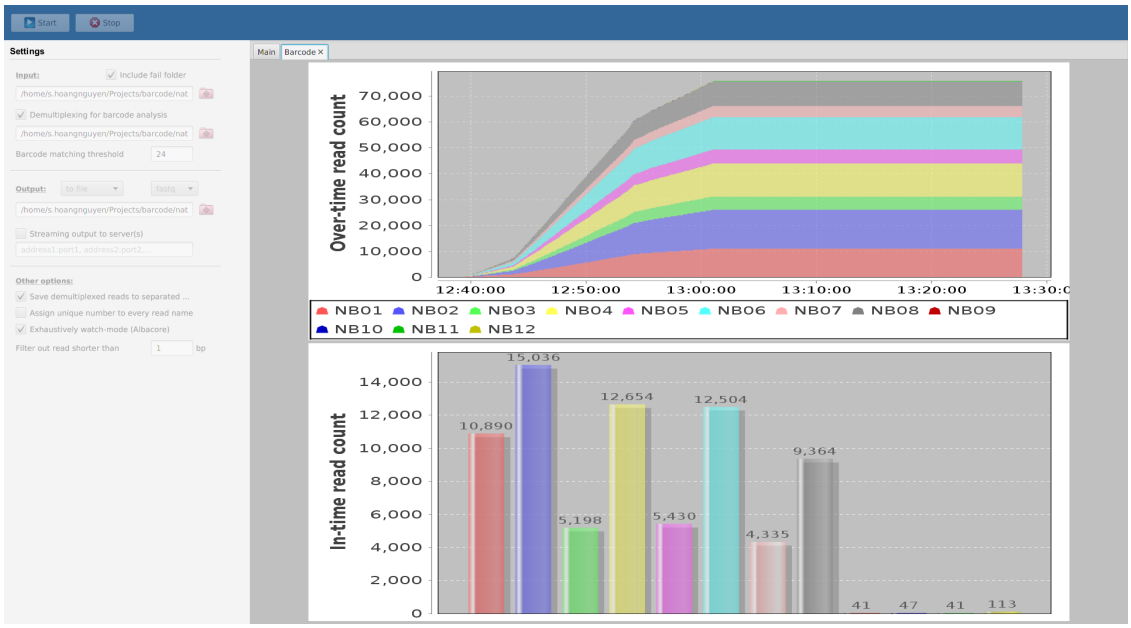


FIGURE 3.3: Graphical User Interface of npBarcode integrated in npReader. The result shown is for the a MinION run using Native barcoding kit on 8 libraries.

`npBarcode` is also integrated into `npReader`'s graphical user interface. This allows users

to monitor the amount of sequencing data for each pooled sample in real-time. When required, `npReader` provides a view showing read-count per bin in a real-time fashion. This view provides two plots that can depict the progress in an over-time and in-time manner, respectively. The top graph shows a more general progress viewing by showing the read count of the whole process while the lower graph figures the exact number of binned reads at a particular time point. An example of the visualization is shown in Figure 3.3. Overall it reflects the demultiplex functioning appropriately with the majority of reads falling in the bins corresponding to the samples used for the barcode sequencing. Only an negligible amount of reads are mis-classified into unrelated bins.

3.1.3 Methods

3.1.3.1 Bacterial cultures and DNA extraction

Bacterial strains include *Streptococcus pneumoniae* from the American Type Culture Collection (ATCC 700677) and clinical isolates from Hygeia General Hospital, Athens, Greece or Instituto Dante Pazzanese de Cardiologia, Brazil [122]. Clinical isolates comprise of four *Klebsiella pneumoniae* (3_GR_13, 5_GR_13, 11_BR_13, 22_GR_12), *Klebsiella quasipneumoniae* (21_GR_13), *Acinetobacter baumannii* (source: Hygeia General Hospital, 2013) and *Pseudomonas aeruginosa* (source: Hygeia General Hospital, 2013). It is worth noting that the sample identifiers are re-assigned for convenient as shown in Table B.1 which emphasize on the gram stain of the bacteria: GP for gram positive and GN for gram negative. Bacterial cultures were supplied as stabs/slants or on agar, grown in nutrient broth or brain heart infusion broth, glycerol stocks were made to 20% (v/v) glycerol and stored at 80^{o}C.

For DNA extractions, glycerol stocks were struck out on either nutrient agar or tryptic soy agar with 5% defibrinated sheep blood to isolate single colonies. Strains were grown overnight at 37^o shaking at 220 rpm and DNA subsequently extracted from this inoculum. DNA was isolated using the DNeasy Blood and Tissue Kit (Qiagen) with the additional enzymatic lysis buffer pre-treatment as per the manufacturers instructions. DNA was quantified with Qubit3.0 (ThermoFisher Scientific).

3.1.3.2 Illumina sequencing and assembly

One nanogram of DNA was used for the Nextera XT (Illumina) library preparation as per the manufacturers instructions and quality was evaluated via a 2100 Bioanalyzer (Agilent Technologies). Libraries were sequenced on an Illumina MiSeq (300 bp paired-end reads) with >100X coverage per sample.

The raw reads from each dataset were processed by `Trimmomatic` [118] version 0.36 to remove adapter sequences and attain paired reads only. After that, `SPAdes` 3.10.1 was used to generate short-read assembly consisting of high quality contigs.

3.1.3.3 MinION barcode sequencing

The same DNA extract from the Illumina sequencing run was used for Nanopore sequencing. The library preparation was done following the 2D native barcoding kit (SQK-LSK208+EXP-NDB002) with the input between 900 to 1500 nanograms of DNA for each sample as in Table B.1. The samples were pooled in equimolar concentrations and 15.75ng was loaded into a R9.4 FLOMIN106 flow cell. After 20 hours the flow cell was topped up with 6μL of library.

The sequencing run was stopped after 43 hours and generated 65,029 reads and 331,329,280 events. The reads were basecalled with settings: 2D Basecalling plus Barcoding for FLO-MIN106 250bps - v1.125.

3.1.3.4 Comparative metrics

We conduct a sensitivity/specificity test on the ability to identify the only gram positive bacterial *S. aureus* of the interested binning algorithms. The rationale stems from the fact that it has the most distinctive genome compare to the others (as shown in Appendix Table B.2), therefore it would account for the least probability of having ambiguous alignments. Statistics are generated as: the number of indeed aligned reads from the bin *S. aureus* (true positive); number of unaligned reads in the bin (false positive); number of aligned (false negative) and unaligned (true negative) reads from outside the bin. Sensitivity and specificity are then calculated by following equations: $\texttt{sensitivity} = \frac{\texttt{TP}}{\texttt{TP+FN}}$; $\texttt{specificity} = \frac{\texttt{TN}}{\texttt{TN+FP}}$ where

`TP`, `FN`, `TN` and `FP` stand for true positive, false negative, true negative and false positive respectively.

3.1.3.5 Software UI design

We provide both command line and graphical user interface for nanopore barcode sequencing. While the former is flexible and convenient to those who want to integrate downstream analysis directly from demutiplexed results, the latter offers a simpler maneuver and better visualization.

From terminal, one can invoke either `jsa.np.npreader` with barcode input sequences specified, or dedicated `jsa.np.barcode` module which requires an executable script for downstream analysis.

On the other hand, users can type `jsa.np.npreader -gui` to have a user-friendly demultiplex running, especially for streaming-mode as the sequencing is still ongoing. Beside the traditional `npReader`'s window, whenever having a demultiplex phase involved, there is an additional tab view showing binned read-count in a real-time fashion. This view provides 2 plots that can depict the progress in an over-time and in-time manner respectively. The top graph would offer more general progress viewing by showing the read count of the whole process while the lower graph figures the number of binned reads in more detail at a particular time point.

3.1.3.6 Real-time analysis setup

Users can implement a streaming pipeline that consumes demultiplexed output from `npBarcode` in two ways. The first method is simply watching the content of the appending output files and explicitly pipe them to appropriate downstream processes, *e.g.* by using `tail -f`. In order to do this, ones initially need to tell `npBarcode` to output the demultiplexed sequences into different files to read from.

Another way, which has been used in this study, is to provide a script for downstream scenario to `npBarcode` and let it handle implicitly. This approach will reduce the I/O operations and disk space required for the analysis. For example, in order to invoke `npScarf`

to scaffold multiple samples at the same time using barcoded sequences, we stream the base-called sequence to `npBarcode` command line via the inter-process communication (Linux pipe) or network channels (Java sockets) protocol [123] and provide it a script (*script.sh*) to invoke `npScarf`. An example of such bash script is given in Appendix List B.1.

< `data stream` > | `jsa.np.barcode -seq` − `-bc` *barcode.fasta* `-sc` *script.sh*

where < `data stream` > is the input stream and *barcode.fasta* is the file containing the actual barcode sequences that have been used. From an appending base-called output file, the stream can be achieved by a monitoring command, *e.g.* `tail -f` (for a local mounted file system) or by using socket communication utility, *e.g.* `jsa.util.streamServer` and `jsa.util.streamClient` from Japsa package. Refer to Japsa documentation for more details.

3.1.4 Conclusion

We have described `npBarcode`, a tool supporting real-time demultiplex of nanopore sequencing data. Depending on requirements, users can choose to run the dedicated demultiplexer from command line or using it as part of `npReader`'s graphical user interface. The tool provides practitioners a flexible option to monitor a barcoded sequencing run as well as to integrate pooled sequencing into a streaming analysis pipeline.

Shortly after the manifestation of open-sourced demultiplexers such as `npBarcode`, Albacore and new (commercial) version of Metrichor have been offered to the community that integrate direct FASTQ output as well as demultiplex option. The overlapping functionality lead to the retirement of `npReader` and `npBarcode` maintenance, however, they are still applicable for the use cases that demands flexible control and customization from end users. The code is made available and free via Japsa package.

In the next section, we adopt a combination of in-house tools and other open-source software for another assembly use case in order to comprehend the resistance gene profile of several Extensively Drug-Resistant (XDR) *K. pneumoniae* strains using MinION sequencing.

3.2 Assembly of multiple XDR strains for *Klebsiella pneumoniae*

The following detail is extracted from a submitted publication with permission from its corresponding authors. As part of this thesis, I only report my major contributions in the publication, *i.e.* genome assembly and associated computational analysis on the DNA barcode sequencing data.

In this section, a combined application of `npScarf` with other assemblers is implemented to generate the final assembly of interests. Such practice can demonstrate in depth the performances of each method and at the same time, provide a consensus approach for a *de novo* reference-free assembly. This use case also highlights the ability of long-read assembly in discerning the location of acquired resistance in the whole genomes of XDR bacterial strains, especially on the identified plasmid(s).

3.2.1 Introduction

The bacterial samples exclusively subjected to this research conduct were *Klebsiella pneumoniae*, a species known as one of the prominent pathogens found in health care facilities. As a matter of fact, *K. pneumoniae* is vastly responsible for the risks of *nosocomial* infections (caused by transmission of germs within hospital environment) with mortality rate has been observed as high as 50% [124–128]. More importantly, the treatment options are becoming limited as the bacteria is developing its resistance to the available antibiotics, including carbapenems, fosfomycin, tigecycline and polymyxins [129]. As shown by current evidence, plasmids are the primary source of resistance genes [130] of the *superbugs* and are irrepressibly disseminating them to other strains, accounting for the rapid global dissemination of resistance [124, 131]. More severely, pandrug-resistant (PDR) *K. pneumoniae* strains have been reported as resistant to all commercially available antibiotics [132, 133] at the moment.

The shortcoming for a robust detection methodology to accurately assess bacterial infections, the resistance profile in particular, has been considered as one factor accounting for the exhibition of antibiotic resistance [134]. In light of antimicrobial therapy, this drawback

gave rise to the unnecessary applications of antibiotics for viral infections and ineffective antibiotics being administered for resistant infections. Rapid sequencing has been proposed as a way to determine PDR profiles, including approaches which utilize high accuracy short reads, as well as those which utilize real-time single molecule sequencing. In particular, MinION is a portable single-molecule sequencer which can sequence long fragments of DNA and stream the sequence data for further data processing in real-time to detect the presence of bacterial species and acquired resistance genes [91, 95, 135–137].

Moreover, the long reads coupled with the ability to multiplex samples has immensely aided with the assembly of bacterial genomes [98, 138–140]. This capability allows for the rapid determination of whether resistance is residing on the chromosome or plasmid(s). Of particular interest are high levels of resistance encoded on plasmids, as these genes can rapidly be transferred throughout the bacterial population via horizontal gene transfer.

3.2.2 Data description

MinION sequencing and associated computational analysis were conducted to interrogate the resistance profile in the whole genomes of four XDR clinical *K. pneumoniae* strains. These isolates had been previously sequenced by Illumina platform and undergone antimicrobial susceptibility testing. Three of them, namely 1_GR_13, 2_GR_12 and 16_GR_13, exhibited resistance to all 24 classes or combinations of antibiotics tested, including the 'last resort' polymyxin. Accordingly, there were high abundance of antibiotic resistance genes found in these samples' genome (≥ 26) [122]. Another polymyxin-susceptible XDR isolate, 20_GR_12, was included for comparative studying purpose.

The WGS sequencing yield and length statistics for each sample are shown in Table 3.1. Illumina paired-end data was assembled by `SPAdes` v3.10.1 [68]. The fact of having more than 60-folds coverage for each sample assures the high quality of the short-read assembly. The Rapid Barcoding Sequencing was used for a mixture of 1_GR_13, 16_GR_13 and 20_GR_12 and the base-called data were demultiplexed by `npBarcode`. Isolate 2_GR_12, due to potential carbonhydrate contamination, has been prepared separately and subjected to another MinION run using Rapid Sequencing Kit. As shown in the table, this sample has

Table 3.1: Sequencing data statistics for each *K. pneumoniae* strains.

Strain	Lineage	Illumina		Nanopore		
		Coverage(X)	N50[a]	Coverage(X)	N50[b]	Time(mins)
1_GR_13	ST147	63	242,838	215	8,712	1,279
2_GR_12	ST258	158	196,706	67	5,251	2,468
16_GR_13	ST11	123	203,529	101	5,012	1,277
20_GR_12	ST258	262	256,217	115	10,151	1,277

[a] of the Illumina assembly contigs.

[b] of the raw Nanopore sequences.

the least coverage of Nanopore data despite of the longest sequencing time. The low N50 values of 2_GR_12 and 16_GR_13 ($\simeq$ 5Kbp) may introduce difficulties in completing their genomes using hybrid approaches, however, the abundance of long-read data from the latter would increase the chance of having appropriate bridges over the repetitive elements.

3.2.3 *De novo* assembly with multiple approaches

In order to have the assembly of high confidence without any reference genomes, multiple prominent approaches were applied to the dataset. The real-time assembly is not reported here as exhaustive operations of MinION flow cells were established in expectations to obtain as much data as possible, not to mention `npScarf` is the only tool supporting the streaming mode. Instead, only the final assembly were of interest. After having results from different assemblers, an alignment-based comparative interrogation was conducted to investigate the completeness and quality of each contigs in relations with its corresponding counterparts. By doing so, the consensus sequence can be determined as the best candidate among all.

Table 3.2: Contigs in the genome assembly results of 4 *K. pneumoniae* isolates using multiple approaches. Output identifies circular sequences in **bold** and underlined if chosen for the final assembly.

Isolate	Assembly method			
	npScarf	**Unicycler**	**Canu**	**minasm+Racon**
1_GR_13	**5,289,533**; **193,063**; **168,873**; 61,039; **53,489**; 55,020; **6,457**	**5,181,675**; **192,771**; 159,172; **108,879**; **55,018**; **53,495**;	**5,073,727**; **226,704**; **184,820**; **136,154**; 100,186; **82,924** ; 54,495	**5,167,584**; **192,237**; **168,292**; **108,582**; **53,325**
2_GR_12	**5,466,424**; 457,649; 60,365; 42,041; 32,458; 21,300; **13,841**; 12,226	3,743,268; 1,694,231; 175,636; 152,644; 95,481; **43,380**; 28,913; 26,127;16,315; 16,781; **13,841**	5,440,093; 392,198; 122,463; 57,534; 26,891	3,769,033; 1,696,038; 204,124; 179,356; 158,561; **43,085**; 13,400; 2,250
16_GR_13	**5,426,917**; **186,908**; **154,971**; **63,588**; 37,608; 35,578; **5,225**; 4,426; **3,703**	**5,426,765**; **187,670**; **155,161**; **63,589**; **5,234**; **4,940**; 2,156	**5,400,611**; **199,904**; **180,542**; **83,467**; 14,853; 11.623; 9,308; 9,423; 7,568; 7,089	**5,410,256**; **186,950**; **154,635**; **63,299**; 5,100; 4,900
20_GR_12	**5,391,578**; **163,468**; **50,940**; 50,856; 12,578	**5,395,894**; **170,467**; **50,979**; **43,380**; **13,841**; 4,645	5,342,491; **192,947**; 74,844; 72,588; 25,624	**5,380,057**; **169,880**; **50,636**; **43,157**; 13,600

In particular, the enforced hybrid assemblers included `npScarf` and `Unicycler` v0.3.1 [87]. Assemblers using only ONT reads were `Canu` v1.5 [79] (excluding read shorter than 500bp) and a combination pipeline of `miniasm` v0.2-r-168-dirty, `minimap2` v2.1-r311; `Racon` (git commit 834442) were used in both cases to polish the assemblies afterward [50]. Consensus sequences were determined among all obtained assemblies manually using visualization from `Mauve` [141] snapshot_2015-02-13. From which, potential bridging differences between candidates were further investigated by alignments with Illumina paired-end data on `IGV` 2.3 [142, 143].

The output from each assembly method is reported in Table 3.2. Overall, `Unicycler` returned most accurate results and as the consequence, a majority of the final assembly were composed from its output contigs. The high quality achieved thanks to the application of the optimized assembly graph in `Unicycler`'s hybrid assembly module, as well as comprehensive post-processing steps carried out on the draft sequences subsequently [87]. `npScarf`, on the other hand, can generate longer contigs but the results are more susceptible to misassemblies. The algorithm, which is independent on the assembly graph, can bridge two nodes that lack connections in the graph, but at the same time, is more prone to erroneous alignments. The two remaining methods, even though with polishing step, produced inferior results but would give additional information for the ultimate backbones of the assembly.

Amongst four datasets, 2_GR_12 shows most fragmented assembly as the shorter reads could not resolve all repeats. For this sample, only `npScarf` and `Canu` can successfully assemble the chromosomal sequence as the longest contig of length $\simeq$ 5.4Mbp. Two other circular sequences were reports as complete plasmid DNA from `Unicycler`. In addition, there were 3 unfinished (linear) sequences in the final assembly for this sample, corresponding to 5 fragmented contigs from `Unicycler`. Another longest contig found by `npScarf` belonged to 16_GR_13 dataset (5,426,917bp). `Unicycler` reported very close circular sequence (5,426,765bp) with equivalent quality but hosting less genes than the former thus was not selected in this case.

3.2.4 AMR analysis on the final assembly

TABLE 3.3: Final assembly of XDR *K. pneumoniae* isolates and location of antibiotic resistance genes.

Isolate	Contig[a]	Length[b] (bp)	Abundance (X)	Resistance Genes[c]
1_GR_13	C	**5,181,675**	1	blaSHV-11, fosA, oqxA, oqxB
	P: IncA/C2	**192,771**	1.95	aadA1, ant(2")-Ia, aph(6)-Id, ARR-2, blaOXA-10, blaTEM-1B, blaVEB-1, cmlA1, dfrA14, dfrA23, rmtB, strA, sul1, sul2, tet(A), tet(G)
	P: IncFIB$_{pKpn3}$, IncFII$_{pKP91}$	**168,873**	2	aadA24, aph(3')-Ia, aph(6)-Id, dfrA1, dfrA14, strA
	P: IncFIB$_{pKPHS1}$	**108,879**	1.53	-
	-	**55,018**	14.10	-
	P: IncR, IncN	**53,495**	2.36	aadA24, aph(3')-Ia, aph(6)-Id, blaVIM-27, dfrA1,mph(A), strA, sul1
2_GR_12	C	**5,466,424**	1	blaSHV-11, fosA, oqxA, oqxB
	P: IncFIB$_{pKpn3}$, IncFIIK	197,872	1.3	aadA2, aph(3')-Ia, catA1, dfrA12, mph(A), sul1
	P: IncA/C2	175,636	1.49	aadA1, ant(2")-Ia, aph(3")-Ib, aph(6)-Id, ARR-2, blaOXA-10, blaTEM-1A, blaVEB-1, cmlA1, dfrA14, dfrA23, rmtB, sul1, sul2, tet(A), tet(G)
	P: IncFIB$_{pQil}$	95,481	1.61	blaKPC-2, blaOXA-9, blaTEM-1A
	P: IncX3	**43,380**	1.91	blaSHV-12
	P: ColRNAI	**13,841**	4	aac(6')-Ib, aac(6')Ib-cr
16_GR_13	C	**5,426,917**	1	blaSHV-11, fosA, oqxA, oqxB
	P: IncFIB$_{pKpn3}$, IncFIIK	**187,670**	0.88	aac(3)-IIa, aac(6')Ib-cr, aadA2, aph(3')-Ia, blaCTX-M-15, blaOXA-1, catB4, dfrA12, mph(A), sul1
	P: IncA/ C2	**155,161**	0.99	aadA1, ant(2")-Ia, aph(3")-Ib, aph(6)-Id, ARR-2, blaOXA-10, blaTEM-1B, blaVEB-1, cmlA1, rmtB, sul1, sul2, tet(A), tet(G)
	P: IncL/ MpOXA-48	**63,589**	1.49	blaOXA-48
	-	**5,234**	188.49	-
	P: ColRNAI	**4,940**	97.77	-
20_GR_12	C	**5,395,894**	1	blaSHV-11, fosA, oqxA, oqxB
	P: IncFIB$_{pKpn3}$, IncFIIK	**170,467**	1.77	aph(3')-Ia, blaKPC-2, blaOXA-9, blaTEM-1A
	P: IncN	**50,979**	1.42	aph(3")-Ib, aph(6)-Id, blaTEM-1A, dfrA14, sul2, tet(A)
	P: IncX3	**43,380**	1.78	blaSHV-12
	P: ColRNAI	**13,841**	10.82	aac(6')-Ib, aac(6')Ib-cr

[a] Contig identity indicating chromosome (C) or plasmid (P: replicon determined by PlasmidFinder 1.3).

[b] **Bold** indicates circular sequences.

[c] Resistance genes determined by ResFinder 3.0 and displayed in alphabetical order.

Table 3.3 shows the consensus assembly of four XDR *K. pneumoniae* strains. Genome annotations had been accomplished by using `Prokka` v1.12 [108]. Plasmids were identified among the output contigs by looking for the *origin of replication* sequences from the collection of `PlasmidFinder` 1.3 [144]. The location of acquired antibiotic resistance genes were determined based on alignments with `ResFinder` 3.0 [119] database.

Overall, the assembly contigs were circular as ones should be for fully resolved microbial genomes, with the exception of 2_GR_12 as aforementioned. The chromosomal sequence for each isolate was located on top of the contig list from Table 3.3 with the genome size varying between $5.1 - 5.5$Mbp. They were shown to carry the same set of resistance genes including blaSHV-11, fosA and oqxAB.

According to Table 3.3, the majority of resistance ($\geq 75\%$) were hosted by plasmids. In all samples, there was at least one megaplasmid, defined as a plasmid larger than 100Kbp, being assembled. From which, the replicon sequence IncA/C2 or InFIB and IncFIIK were commonly found. The IncA/C2 plasmid appeared in all samples except 20_GR_12. This plasmid contained up to 16 resistance genes which conferred resistance towards aminoglycosides, β-lactams, phenicols, rifampicin, sulphonamides, tetracyclines and trimethoprim, with the exception of 16_GR_13. Isolate 16_GR_13 lacked trimethoprim resistance on its IncA/C2 plasmid. The plasmids containing both replicons IncFIB and IncFIIK differed vastly between all four replicates. All contained IncFIB$_{pKpn3}$ and IncFIIK, however, 1_GR_13 differed with IncFII$_{pKP91}$. Additionally, a differing IncFIB replicon was detected on a separate contig in 1_GR_13 (pKPHS1) and 2_GR_12 (pQil). The only instance where another dual replicon was identified was in 1_GR_13 which harboured both IncR and IncN. This plasmid contained aminoglycoside, β-lactam, trimethoprim, macrolide and sulphonamide resistance. Assembly for 1_GR_13 also contained a 5.5Kbp circular contig which was annotated as a phage genome.

The ColRNAI plasmid was shown in every sample but 1_GR_13 which encoded aminoglycoside and quinolone resistance (aac(6')-Ib, aac(6')-Ib-cr) (Table 3.3). The ColRNAI plasmid in 2_GR_12 and 20_GR_12 was $13,841$ bp in size and shared 75% similarity between the two isolates. This plasmid differed in 16_GR_13 (35% the size) with no resistance genes being found. The same IncX3 plasmid ($43,380$ bp) was apparent in isolates 2_GR_12 and

20_GR_12. Unique to 16_GR_13 was the IncL/ MpOXA-48 plasmid containing blaOXA-48 and the 50,979 bp IncN plasmid in 20_GR_12 with resistance against 5 classes (aminoglycoside (aph(3")-Ib, aph(6)-Id), β-lactam (blaTEM-1A), sulphonamide (sul2), tetracycline (tet(A)), trimethoprim (dfrA14)) of antibiotics.

Multiple copies of acquired resistance genes were located across plasmids in several isolates. For 1_GR_13, up to three copies were observed for genes aadA24, aph(3')-Ia, aph(6)-Id, dfrA1, dfrA14, strA and sul1. In 2_GR_12, the copy number of sul1 and blaTEM-1A were two and for 16_GR_13, sul1 was spotted with the same frequency.

3.2.5 Data availability

Whole genome sequencing of 4 clinical isolates and their final assemblies have been uploaded to NCBI under BioProject PRJNA307517. MinION DNA sequencing data has been deposited within project ID SRP133040. Accession numbers are as followed: SRR6747887 for 1_GR_13; SRR6747886 for 2_GR_12; SRR6747885 for 16_GR_13 and SRR6747884 for 20_GR_12.

3.2.6 Discussion

The ability for ONT to sequence long fragments of DNA in parallel has present an efficient method to finish and resolve the assembly of bacterial genomes and plasmids [97, 138], particularly *superbugs* that host high levels of resistance.

In this use case, we have identified multiple megaplasmids ($\geq$ 100Kbp), which were previously unresolved via Illumina sequencing [122]. These harboured replicons IncA/C2 or a dual replicon, IncFIIK and IncFIB. The IncA/C, IncF and IncN plasmids have been commonly associated with multidrug resistance [145]. Although several plasmids in this study revealed similarity to previously reported isolates via NCBI, various sequences deviated. In particular, the IncA/C2 plasmid exhibited multiple regions unique to these isolates. Several IncA/C2 megaplasmids have been previously described which harbour various resistance genes, however, the extent of resistance in this study has yet to be unveiled [146, 147]. Prior studies have shown the IncFIIK and IncFIB replicons to localize on the same plasmid and

also megaplasmids with multidrug resistance [131]. The IncFIB$_{pQil}$ plasmid in this study contained various β-lactam resistance genes (blaKPC-2, blaOXA-9, blaTEM-1A) which has been identified previously [148]. Similarly, blaOXA-48 segregated with the IncL/M replicon [149, 150], however, deviations in this plasmid were identified.

The final assembly has been generated by using multiple methods available, in which `Unicycler` returned the most accurate contigs. `npScarf` can produce assembly of highly continuity with decent quality, but susceptible to misassemblies in some cases due to its greedy approach. Since these mistakes can affect the fidelity of the results, *e.g.* when studying the highly variant XDR plasmids, improvement in term of accuracy is needed for `npScarf`. Methods to adopt assembly graph will be investigated on top of the original algorithm in the next chapter to serve this purpose.

4

Integration of assembly graph into scaffolding pipeline

Timing and accuracy is really what matters at the end of the day.

–Carson Wentz

Actually the idea of using assembly graph for scaffolding in `npScarf` has been considered from the very beginning of the software design. However, the computational challenges hinder its real-time process ability thus being neglected from the first versions of `npScarf`. In the first section of this chapter, I will introduce the concept and model of the SPAdes assembly graph which would be used as the input. The next section will describe the first attempt to apply this information into `npScarf` for a better gap-filling. Finally, a real-time assembly graph bridging algorithm will be shown in a new tool, namely `npGraph` together with its performance in numerous use cases.

4.1 Assembly graph

`npScarf` is a hybrid assembler that essentially conducts scaffolding on the SGS contigs and then filling the gaps, without using information from the short-read assembly process about the formation of the contigs themselves. This is illustrated in Figure 1.6 where the algorithm enhances step 2 and 3 by using nanopore long reads in an attempt to give more complete genome assemblies. However, SGS assemblers usually provide a richer source of information than just the contig sequences themselves, namely *assembly graph*. For the graph-based approaches, either OLC or DBG, assembly graph stores the ultimate string content of DNA sequences and their links in its components: edges and vertices. This is usually resulting from running a simplification algorithm on the original graph built from input data, *e.g.* tips clipping, bubble removing, error removal, and/or scaffolding using paired-end or mate-pair links [67, 68] (Section 1.3).

Assembly graph is normally implemented by using a directed graph data structure. In which, each component (vertex or edge, depends on particular implementation) storing a string has a corresponding counterpart for that string in reversed complement order [67]. To have a more compact representation of the graph, we employ the bidirected graph data structure instead.

For a bidirected graph $G = \{V, E\}$, each edge has two directions associated with two nodes forming it. The direction here refers to a binary value that represent either incoming or outgoing state. Unlike an edge in a directed graph, a bidirected edge can be traversed in

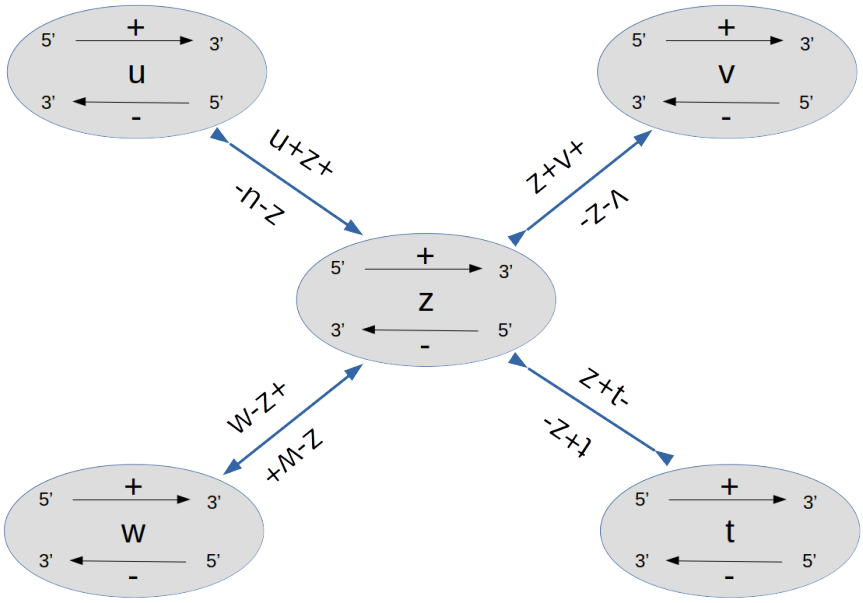


Figure 4.1: An example of bidirected graph model. Each node contains a DNA sequence that can be spelled as template(+) or reverse complement(-). There are four possible types of a bidirected edge as shown as ones from node z. Each edge has two possible spelling depends on the starting node, resulting in two sequences of template and reversed complement. According to the rule of bidirected graph traversal, we have the following valid paths spelled as: $(u+, z+, v+)$, $(u+, z+, t-)$, $(w-, z+, v+)$, $(w-, z+, t-)$ and in the opposite direction respectively $(v-, z-, u-)$, $(t+, z-, u-)$, $(v-, z-, w+)$, $(t+, z-, w+)$.

both ways, similar to the property of an undirected edge except the content of each traversed vertex is not unique, but direction-wised. This means that if there exist a path from node A to B (or B is reachable from A) then the reversed path is legal as well (A is reachable from B), with a reverse-complemented spelling manner. Thus another equivalent interpretation for bidirected graph is an undirected graph of *directed vertices*, which can be related directly to the purpose of modeling connections of DNA sequences as nodes in a graph structure, regarding they are double-stranded (sense and anti-sense). The edge-based modeling, in short, is more intuitive to work with; however in terms of comprehensive understandings and implementations, the directed-node ideology is additionally helpful.

Edge-based modeling From this point of view, any edge $e \in E$ connecting an ordered pair of node $(u, v) \in V^2$ would fall into one out of four categories: ►►, ►◄, ◄► or ◄◄. By modeling edge directions as such, the graph traversal is able to cover the double-stranded

property of the DNA sequences (template/reverse complement) in vertices through edge-walking. The rule is that if we follow the arrow, its associated vertex is spelled as it is (+) and if we traverse against the arrow direction, the corresponding vertex is spelled as reverse complement sequence (-). As mentioned before, there are 2 ways of traversing through an edge, *e.g.*from u to v and from v to u, as illustrated in Figure 4.1.

Vertex-based modeling From this angle, it is convenient to firstly introduce the definition of directed node, $\overrightarrow{v} = (v, d_v) \in \overrightarrow{V}$, as a tuple of vertex v and its associated direction $d_v \in D = \{in, out\}$. In this case, a bidirected edge is almost similar to an undirected edge connecting two (directed) vertices, *e.g.*an unique edge that connect $\overrightarrow{u}$ and $\overrightarrow{v}$ can be equally written as $(\overrightarrow{u}, \overrightarrow{v})$ or $(\overrightarrow{v}, \overrightarrow{u})$. These two forward and reverse edges would refers to the same bidirected edge, except the content will be different depending on the order of spelled directed nodes.

The two definitions are equivalent and their applications are interchangeable and cohesive in our implementation. In summary, the meaning and spelling for a bidirected graph components is as follow:

- u ▶▶ v equivalent to $(u - out, v - in)$: spelling $u + v+$ or $v - u-$
- u ▶◀ v equivalent to $(u - out, v - out)$: spelling $u + v+$ or $v - u-$
- u ◀▶ v equivalent to $(u - in, v - in)$: spelling $u - v+$ or $v - u+$
- u ◀◀ v equivalent to $(u - in, v - out)$: spelling $u - v-$ or $v + u+$

To create a valid path when traversing a bidirected graph, two adjacent edges must obey the direction rule similar to a conventional directed graph, that is a path which has an edge entering an intermediate node (not 2 ending nodes) must leave that node on the next edge. In addition, due to the allowance of traversing against the arrows in bidirected graph, it is possible to follow edges against their direction but the spelling will be reverse-complemented.

4.2 Application of the assembly graph in npScarf_wag

The first attempt of utilizing assembly graph is to integrate this pieces of information into the scaffolding algorithm for the gap-filling step thus increasing the accuracy of the final assembly. This is due to the fact that the edges from assembly graph come from Illumina data with much higher accuracy (99.9%) compared to nanopore data (≈90%), which usually require high abundance to generate a consensus read with comparative quality.

As shown in Figure 4.2, for the first approach of gap filling, which is implemented in the very first version of npScarf, segments from nanopore reads corresponding to a gap are first extracted based on the alignments to the flanking regions. A set of these pileup segments are then undergone a consensus calling step by multiple sequence aligner *e.g.* *kalign* [151] or *poaV2* [47–49], and the result sequence can later be used to fill in the gap.

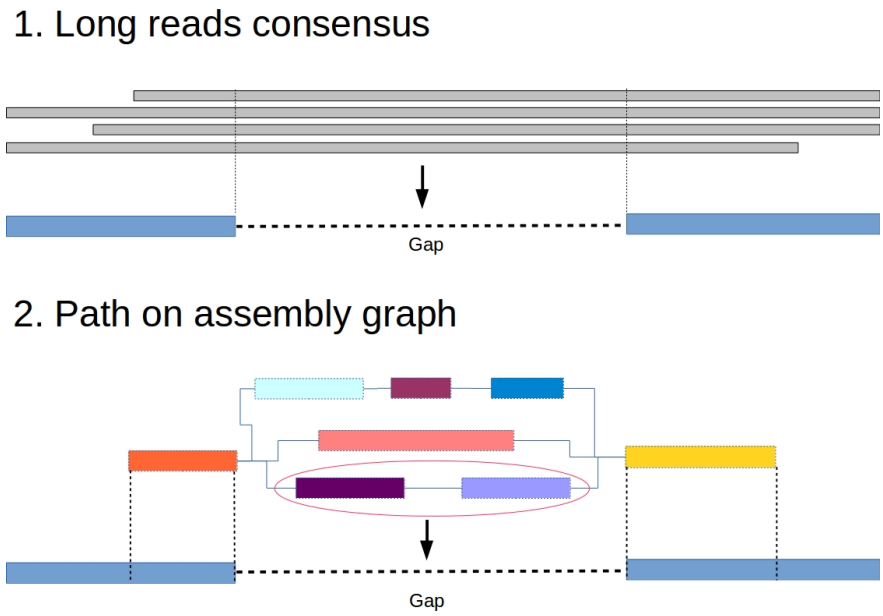


Figure 4.2: Two approaches for gap filling: 1. using consensus sequences from long reads in npScarf and 2. using corresponding path in the assembly graph in npScarf_wag .

Another npScarf version, namely npScarf_wag, utilizes assembly graph information for the gap filling step. In which, assembly graph from SPAdes is also learned together with contigs used as pre-assemblies in the workflow shown in Figure 2.1. It is worth mentioning that each of these contigs is made of a path by traversing the assembly graph and normally short-read assemblers allow backtracking this piece of information, *e.g.* via "*contigs.paths*"

file output from `SPAdes`.

Mathematically, of this thesis' scope, a (bidirected) path $p \in P$ in (bidirected) graph $G\{V, E\}$ is defined as a walk through a series of alternative vertices and edges that are connected together, given that the bidirected transition rule holds true for every intermediate vertices. For example, from the edge-based modelling point of view, a path p of size k can be written as $p = \{v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k\}$ where e_i is the bidirected edge connects v_{i-1} and v_{i+1} with every $1 \leq i < k$ and will determine the their directions in the walk. A path of size 0 contains only one vertex and no edge. Importantly, the transition rule requires $dir(e_i, v_i) \neq dir(e_{i+1}, v_i)$, $\forall 1 \leq i < k$ where $dir : (E, V) \rightarrow D$ is the function that returns direction (from vertex-based modelling) of a vertex on a given edge.

Using this decomposition, the corresponding components of the assembly graph (consecutive nodes) that make up the corresponding flanking tips of the gap are identified. Then by applying a searching algorithm, *e.g.* Depth-First Search (DFS), on the graph, a set of candidate paths connecting two tipping nodes are determined. By selecting the path that maximize the likelihood of the long reads, the gap can be filled up by the sequence spelled out from it as shown in Algorithm 1.

Algorithm 1: Pseudo-code for filling gap with assembly graph in `npScarf2`.

Data: Assembly graph $G\{V, E\}$

Input: Bridge $B = (contig_1, contig_2)$ with gap

Output: Sequence to fill in the gap of B

```
1 begin
2     p_1 := decompose(G, contig_1)      // decompose first contig into graph components
3     p_2 := decompose(G, contig_2)      // decompose second contig into graph components
4     v_1^→ := p_1.peek()                // get the last directed vertex of p_1
5     v_2^→ := p_2.getRoot()             // get the first directed vertex of p_2
6     p := DFS(G, v_1^→, v_2^→)          // find a path connect these 2 tips
7     return p.spell()
```

Theoretically, this approach can increase the accuracy of gap-filling step, however, it offers limited capacity in reducing the misassemblies. The reason is that the ending components of a contig can be repetitive, make it impossible to confirm a genuine bridge with the existence

of a path connect two tips as above. To deal with this issue, the assembly graph should be applied completely to the scaffolding algorithm by using the graph components as the building blocks, in replacement of the independent contigs as in `npScarf` and `npScarf_wag`. This idea lead to the development of `npGraph` - a graph-based tool that can finish genomes in real-time.

4.3 Resolve assembly graph in real-time by long reads with npGraph

4.3.1 Introduction

Streaming assembly methods have been proven to be useful in saving time and resources compared to the traditional batch algorithms with examples included *e.g.*`Faucet` [152] and `npScarf` [97]. The first method allows the assembly graph to be constructed incrementally as long as reads are retrieved and processed. This practice is helpful dealing with huge short-read dataset because it can significantly reduce the local storage for the reads, as well as save time for a DBG construction while waiting for the data being retrieved.

`npScarf`, on the other hand, works on the available short-read assembly to scaffold the contigs using nanopore sequencing which is well-known by the real-time property. The completion of genome assembly along with the sequencing run provides explicit benefits in term of resource control and turn-around time for analysis [97]. However, due to the greedy approach of a streaming algorithm, as well as being an alignment-based-only scaffolding mechanism, running the tool with default settings suffers from misassemblies [87, 153]. In many case, the gap filling step has to rely on the low quality nanopore reads thus the accuracy of the final assembly is affected as well. To tackle the quality issue while maintaining its streaming execution, an assembly graph processing system is investigated as it would provide an additional high-quality source of linking information for the assembly operations.

After the construction of an assembly graph, the next step is to traverse the graph, resolve the repeats and identify the longest possible un-branched paths that would represents contigs for the final assembly. Hybrid assembler using nanopore data to resolve the graph has been

implemented in `hybridSPAdes` [154] or `Unicycler` [87]. In general, the available tools employ batch-mode algorithms on the whole long-read dataset to generate the final genome assembly. In which, the `SPAdes` hybrid assembly module, from its first step, exhaustively looks for the most likely paths (with minimum edit distance) on the graph for each of the long read given but only ones supported by at least two reads are attained. In the next step, these paths will be subjected to a decision-rule algorithm, namely `exSPAnder` [155], for repeat resolution by step-by-step expansion, before output the final assembly. On the other hand, `Unicycler`'s hybrid assembler will initially generate a consensus long read for each of the bridge from the batch data. The higher quality consensus reads are used to align with the assembly graph to find the best paths bridging pairs of anchored contigs. While the later approach employs the completeness of the dataset from the very beginning for a consensus step, the former only iterates over the batch of possible paths and relies on a scoring system for the final decision of graph traversal. For that reason, the first direction is more suitable for a real-time pipeline.

However, the challenges to adapt this approach into a real-time mechanism are obvious and mainly come from the heavy path-finding task and the complication of self-improvement step which is critical to a streaming algorithm. A modified DFS algorithm and a voting system with accumulating scores calculation has been implemented to overcome these issues. This results in `npGraph`, an user-friendly tool with GUI that can traverse the assembly graph and scaffold its components in real-time as long as the nanopore sequencing process is running and continuously generating long reads.

4.3.2 Methods

4.3.2.1 Overview

The input consists of Illumina assembly graph resulted from running assembler, e.g. `SPAdes` [68], `Velvet` [67], `AbySS` [69] on Illumina short reads, together with long reads from third generation sequencing technology (Oxford Nanopore Technology, Pacbio). The long reads will be aligned with the contigs in the assembly graph to indicate longer paths that should be traversed. These local paths, given sufficient data, are expected to untangle the complicated

graph and guide to the global Eulerian paths (or cycles if possible) that represent the entire genomic sequences.

Basically the work flow of `npGraph` can be divided into 3 main phases: abundance binning, graph resolving and assembly output. The first step is to bin the contigs into different groups of *population* based on their coverage. Each population would include contigs of the similar abundance in the final assembly sequences, *e.g.* chromosome , plasmids, or even particular species genome in a metagenomics community. The binning phase would assist to differentiate between repetitive contigs and unique ones as well. By using this information, in combination with paths inducing from long reads, the assembly graph is then traversed and resolved in real-time. Finally, the graph is subjected to the last attempt of resolving and cleaning, as well as output the final results. The whole process can be managed by using either command-line interface or GUI.

4.3.2.2 Pre-processing: identify repeat and unique contigs

The purpose of this step is to investigate the sequencing properties of contigs, *i.e.* length and *k-mer* count, in combination with the graph topology, for an initial binning algorithm to determine multiplicity for each of them.

Initial binning contigs into groups of abundances Each contig is represented as a node in the assembly graph and an edge connecting two nodes indicates their overlap (link) properties. This step is to cluster the significant nodes (longer than 10Kbp) into different sets, namely *core* groups, based on their degree and coverage values.

DBSCAN clustering algorithm [156] is applied for this task. The idea is to consider a coverage value of a significant contig (which can be split into more than 10,000 *k-mers*) to be a sampled mean of a Poisson distribution (of *k-mers* count). The metric is a distance function based on Kullback-Leibner divergence [157], or relative entropy, of two Poisson distributions.

Assume there are 2 Poisson distribution P_1 and P_2 with density functions

$$p_1(x, \lambda_1) = \frac{e^{-\lambda_1}\lambda_1^x}{\Gamma(x+1)} \tag{4.1}$$

and

$$p_2(x, \lambda_2) = \frac{e^{-\lambda_2}\lambda_2^x}{\Gamma(x+1)} \tag{4.2}$$

The Kullback-Leibner divergence from P_2 to P_1 is defined as:

$$D_{KL}(P_1||P_2) = \int_{-\infty}^{\infty} p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \tag{4.3}$$

or in other words, it is the expectation of the logarithmic difference between the probabilities P_1 and P_2, where the expectation is taken with regard to P_1.

The log ratio of the density functions is

$$\log \frac{p_1(x)}{p_2(x)} = x \log \frac{\lambda_1}{\lambda_2} + \lambda_2 - \lambda_1 \tag{4.4}$$

take expectation of this expression with regard to P_1 with mean λ_1 we have

$$D_{KL}(P_1||P_2) = \lambda_1 \log \frac{\lambda_1}{\lambda_2} + \lambda_2 - \lambda_1 \tag{4.5}$$

The metric we used is the distance defined as

$$D(P_1, P_2) = \frac{D_{KL}(P_1||P_2) + D_{KL}(P_2||P_1)}{2} = \frac{1}{2}(\lambda_1 - \lambda_2)(\log \lambda_1 - \log \lambda_2) \tag{4.6}$$

Coverage re-estimation Due to the possible divergence of sequencing coverage relative to the real abundance of sequences, especially the short ones, an optimization step is implemented to alleviate this issue. The coverage measure of nodes (which represent contigs) are spread throughout the graph via edges that connect them for comparison and calibration. The assignment of coverage to edges is also helpful to identify multiplicity in later step.

The re-estimation is basically carried out by following two steps.

1. From nodes coverage, estimate edges' value by quadratic unconstrained optimization of the least-square function:

$$\frac{1}{2}\sum_i l_i((\sum e_i^+ - c_i)^2 + (\sum e_i^- - c_i)^2 \tag{4.7}$$

where l_i and c_i is the length and coverage of a node i in the graph;

$\sum e_i^+$ and $\sum e_i^-$ indicates sum of the values of incoming and outgoing edges from i respectively. The above function and be rewritten as:

$$f(x) = \frac{1}{2}x^T Q x + b^T x + r \tag{4.8}$$

and then being minimized by using Newton or gradient method.

2. Update nodes' coverage based on itself and its neighbor edges' measures.

The calibration is iterative until no further improvements are made or a threshold loop count is reached.

Multiplicity estimation Based on the coverage values of all the edges and the graph's topology, we induce the copy numbers of every significant nodes (long contigs) in the final paths. For each node, this could be done by investigating its adjacent edges and answering the questions of how many times it should be visited, from which abundance groups. Multiplicities of insignificant nodes (of sequences with length less than $1,000$ bp) can be estimated in the same way but usually with less confident due to more complicated connections and greater variances of coverage values. For that reason, they are only used as augmented information to calculate candidate paths' score in the next step.

4.3.2.3 Untangling assembly graph by stream of nanopore data.

Building bridges in real-time Bridge is the data structure designed for tracking the possible connections between two unique contigs (anchor nodes in the assembly graph). This approach was implemented in `npScarf` gap-filling phase with assembly graph as in Figure 4.2. Here we take advantage of the graph topology and nodes' multiplicity information to employ a dynamic bridging mechanism. This procedure considers the dynamic changing of multiplicity property for each node, meaning that a n-times repetitive node can become a unique node at certain time point when its $(n-1)$ occurrences are identified and assigned in appropriate unique paths.

Other than `npScarf`, a bridge in `npGraph` has several completion levels. A bridge is only created with at least one unique contig as an end, if so it has level 1 of completion. If both ends are identified, the level is 2. The number is greater than that only if paths connecting two ends are found. A bridge is known as fully complete (level 4) if there is only one unique linking path left. Given a bridge with 2 ends, a path finding algorithm (described in next section) is invoked to find all candidate paths. Each of these paths is given a score of

alignment-based likelihood which are updated immediately as long as there is an appropriate long read being generated by the sequencer. As more nanopore data arrives, the divergence between candidates' score becomes greater and only the best ones are voted for the next round.

Whenever a bridge becomes complete thanks to the voting system, the assembly graph is *transformed* or *reduced* by replacing its unique path by an composite edge and removing any unique edges (edges coming from unique nodes) along the path. The assembly graph would have at least one edge less than the original after the reduction. The nodes located on the reduced path, other than 2 ends, also have their multiplicities subtracted by one and the bridge is marked as finally resolved without any further modifications.

Path finding algorithm In `npScarf` with assembly graph, the path finding algorithm is the original DFS (depth first search) which becomes computationally expensive when the traversing depth increases. For `npGraph`, we implement a modified stack-based version utilizing Dijkstra's shortest path finding algorithm [158] to reduce the search space.

Algorithm 2 demonstrates the path finding module in general. In which, function

`shortestTree`$(\overrightarrow{vertex}, distance) : (V, Z) \rightarrow V^n$

from line 3 of the algorithm's pseudo code builds a shortest tree rooted from $\overrightarrow{v}$, following its direction until a distance of approximately d (with a tolerance regarding nanopore read error rate) is reached. This task is implemented based on Dijkstra algorithm. This tree is used on line 4 and in function $includedIn()$ on line 19 to filter out any node or edge with ending nodes that do not belong to the tree.

Basically, the algorithm keeps track of a stack that contains sets of candidate edges to discover. During the traversal, a variable d is updated as an estimation for the distance to the target. A hit is reported if the target node is reached with a reasonable distance *i.e.* close to zero, within a given tolerance (line 21). A threshold for the traversing depth is set (150) to ignore too complicated and time-consuming path searching.

It is worth to mention that the $length()$ functions for node and edge are totally different. While the former returns the length of the sequence represented by the node, *i.e.* contig from short-read assembly, the latter is usually negative because an edge models a link between

Algorithm 2: Pseudo-code for finding paths connecting a bridge with 2 ends.

Data: Assembly graph $G\{V, E\}$

Input: Bridge $B = (\overrightarrow{v_1}, \overrightarrow{v_2})$ with two ending unique bidirected nodes $\overrightarrow{v_1}, \overrightarrow{v_2}$

Output: Set of candidate paths P connecting B

```
1  begin
2      d:=B.length()      // length of the bridge or the distance between 2 ending nodes
3      M:=shortestTree(v2→, d)              // build shortest tree from v2→ with range d
4      if M.contain(v1→) then
5          S:=new Stack()                          // stack of sets of edges to traverse
6          edgesSet:=getEdges(v1→)            // get all bidirected edges going from v1→
7          S.push(edgesSet)
8          p:=new Path(v1→)                          // init a path that has v1→ as root
9          while true do
10             edgesSet:=S.peek()
11             if edgesSet.isEmpty() then
12                 if p.size() ≤ 1 then
13                     break      // stop the loop when there is no more edge to discover
14                 S.pop()
15                 d+=p.peekNode.length() + p.popEdge().length()
16             else
17                 curEdge := edgesSet.remove()
18                 v→:=curEdge.getOpposite(p.peekNode())
19                 S.push(getEdges(v→).includedIn(M))
20                 p.add(curEdge)
21                 if reach v2→ with reasonable d then
22                     P.add(p)
23                 d-=v→.length() + curEdge.length()
24     return P
```

two nodes, which is normally an overlap (except for composite edges). For example, in a *k-mers* DBG-derived assembly graph, the value of an edge is $-k$.

4.3.2.4 Result extraction and output

`npGraph` reports assembly result in real-time by decomposing the assembly graph into a set of longest straight paths (LSP), each of the LSP will present a contig for the result. A path $p = \{v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k\}$ of size k is considered as straight if every edge along the path, $e_i, \forall i = 1, \ldots, k$, must be the only option to traverse from either v_{i-1} or v_i following the transition rule. To decompose the graph, we can just simply mask out all incoming/outgoing edges rooted from any node with in/out degree greater than 1 as demonstrated in Figure 4.3. These edges are defined as branching edges which stop straight paths from further extending.

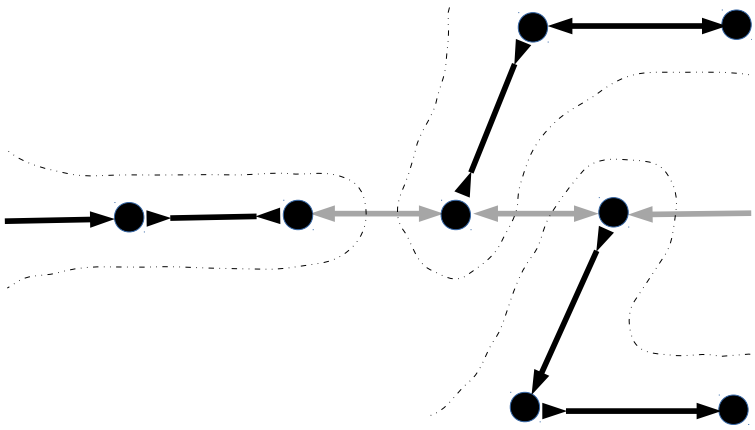


FIGURE 4.3: Example of graph decomposition into longest straight paths. Branching edges are masked out (shaded) leaving only straight paths (bold colored) to report. There would be 3 contigs extracted by traversing along the straight paths here.

The decomposed graph is only used to report the contigs that can be extracted from an assembly graph at certain time point. For that reason, the branching edges are only masked but not removed from the original graph as they would be used for further bridging.

The final assembly output contains files in both FASTA and GFA v1 format (`https://github.com/GFA-spec/GFA-spec`). While the former only retains the actual genome sequences from the final decomposed graph, the latter output file can store almost every properties of the ultimate un-masked graph such as nodes, links and potential paths between them.

4.3.2.5 User interface

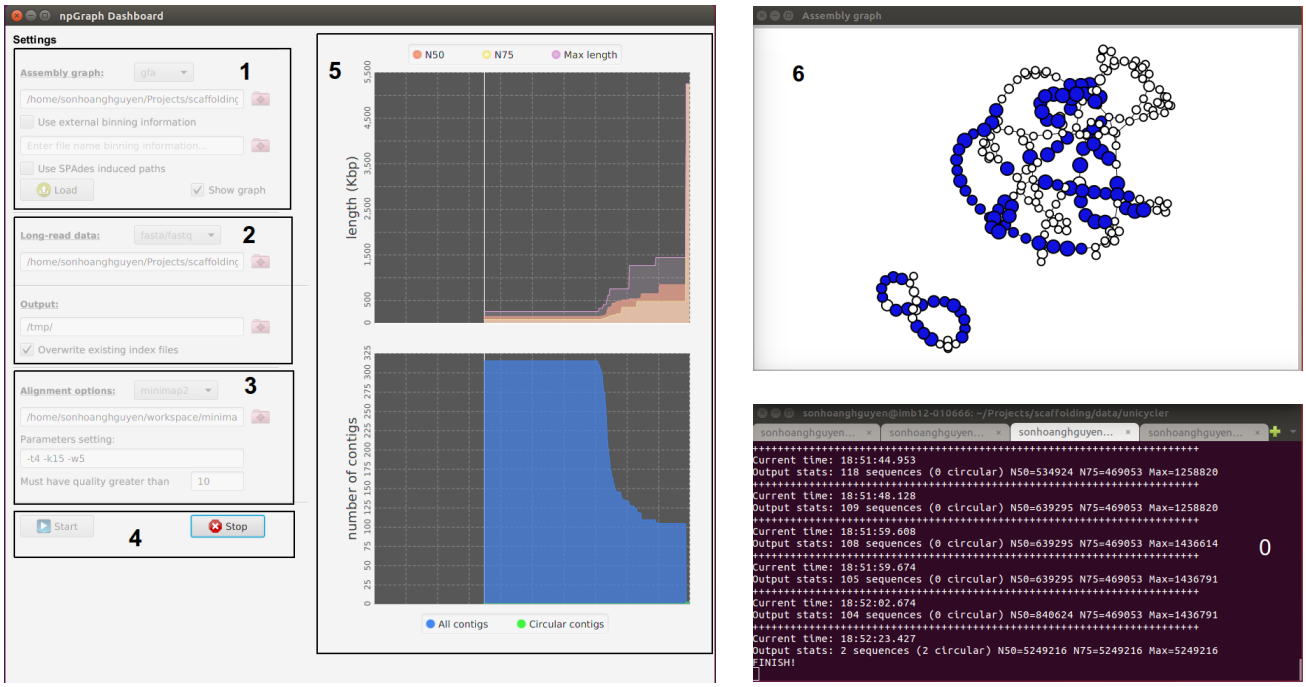


Figure 4.4: `npGraph` user interface including Console (**0**) and GUI components (**1-6**). The GUI consists of the Dashboard (**1-5**) and the Graph View (**6**). From the Dashboard there are 5 components as follow: **1** the assembly graph input field; **2** the long reads input field; **3** the aligner settings field; **4** control buttons (start/stop) to monitor the real-time scaffolding process; **5** the statistics plots for the assembly result.

`npGraph` can be invoked and fully function from the command-line interface. In addition, in order to aid the visualization of the assembly process, a GUI has been developed as well.

The GUI includes the dashboard for control the settings of the program and another pop-up window for a simple visualization of the assembly graph in real-time (Figure 4.4). In this interface, the assembly graph loading stage is separated from the actual assembly process so that users can check for the graph quality first before carry out any further tasks. The box numbered **1** on Figure 4.4 is designed for this task. Only after an assembly graph is loaded successfully, users can move to box **2** to specify the nanopore input data. Settings for an aligner (`BWA-MEM` or `minimap2`) in box **3** is required if the input is the raw sequences in FASTA/FASTQ format. Another option is to run the alignment independently and provide SAM/BAM input for the next stage of bridging and assembly. This stage is controlled

by buttons in box **4**: the START button ignites the process while the STOP button can prematurely terminate it and output the assembly result till that moment. The plots from the right panel (**5**) depicts real-time statistics of the assembly contigs inferred from the graph. From the second window (**6**), the colored vertices imply unique contigs while the white ones involve either unspecified or repetitive elements. The number of different colors (other than white) indicates the amount of abundant groups being detected as population bins (*e.g.* chromosome versus different plasmids, or different bins in metagenomics).

A proper combination of command line and GUI can provide an useful streaming pipeline that copes well with MinION output data. The practice is similar to the previous developed pipelines [33, 97, 98] that allow the analysis to take place abreast to a nanopore sequencing run.

4.3.3 Results

4.3.3.1 Hybrid assembly for synthetic datasets

To evaluate the performance of the method, `npGraph` was benchmarked against `SPAdes` with its hybrid assembly module [154], `npScarf` with/without assembly graph integrated, and Unicycler version 0.4.6 on the latter's testing data [87] . This dataset were simulations of Illumina and MinION raw data, generated *in silico* based on random and available microbial references. Specifically, the synthetic Illumina data was generated by using a wrapper script of ART [159] that allows uniform coverage for circular genome sequencing with justifiable depths. PBSIM [160], on the other hand, was used to simulate the long reads.

There were three settings for each of the synthetic raw data (*good, medium, bad*) corresponding to the quality, yield and length of reads being generated. In the SGS simulation, the *bad* data consist of 100bp paired-end reads with low and uneven coverage (40X) leading to many dead ends in the assembly graph due to missing regions in the genome. The read length was 125bp in the *medium* setting with better depth distributions that could cover the genome better. Finally, the *good* option provided best datasets with 150bp of read length and 100X coverage. While the quality of SGS reads would determine the assembly graph nature, the nanopore data plays critical role in graph resolving. The three settings leveled up

the depth (8X, 16X, 32X) and at the same time, average length (5Kbp, 10Kbp, 20Kbp) and maximum identities (90% ,95% ,98%) respectively. We only considered the *good* configurations of both platforms for the sequence data being tested. Also, since the other comparative tools do not support streaming assembly, there were only batch-mode runs being carried out and the reciprocal results were examined by QUAST 5.0.2 [161].

Table 4.1: Comparison of assemblies produced in batch-mode using npGraph and the comparative methods on 5 synthetic datasets taken from https://cloudstor.aarnet.edu.au/plus/index.php/s/dzRCaxLjpGpfKYW

Method	Assembly size (Mb)	#Contigs	N50 (Kp)	Mis-assemblies	Error (per 100 Kb)	Run times (CPU hrs)
random sequence with repeats						
SPAdes	3.928	226	40.5	0	0.00	0.95
SPAdes-Hybrid	4.109	3	4,000.0	0	0.85	1.196
Unicycler	4.110	3	4,000.0	0	0.47	6.783
npScarf	4.251	9	3,952.2	27	8.74	0.95 + 0.39
npScarf_wag	4.554	9	3,999.6	37	6.16	0.95 + 0.45
npGraph (bwa)	4.110	3	4,000.0	0	0.47	0.95 + 0.33
ngGraph (minimap2)	4.110	3	4,000.0	0	0.47	0.95 + 0.02
Mycobacterium tuberculosis H37Rv						
SPAdes	4.371	114	125.5	1	1.51	1.55
SPAdes-Hybrid	4.411	1	4,411.2	0	1.73	1.68
Unicycler	4.412	1	4,411.5	0	2.56	6.36
npScarf	4.446	4	4,389.9	12	11.41	1.55 + 0.78
npScarf_wag	4.408	1	4,407.6	2	7.01	1.55 + 0.79
npGraph (bwa)	4.411	1	4,411.6	0	7.28	1.55 + 0.63
ngGraph (minimap2)	4.411	1	4,411.4	0	7.01	1.55 + 0.02

Continued on next page

Table 4.1 – continued from previous page

Method	Assembly size (Mb)	#Contigs	N50 (Kp)	Mis-assemblies	Error (per 100 Kb)	Run times (CPU hrs)
E. coli O25b H4-ST131						
SPAdes	5.173	159	191.0	1	1.69	1.26
SPAdes-Hybrid	5.249	7	5,109.6	0	2.65	1.40
Unicycler	5.249	3	5,109.8	0	4.29	4.70
npScarf	5.354	7	5,087.5	14	29.12	1.26 + 0.78
npScarf_wag	5.413	7	5,108.1	6	30.29	1.26 + 0.78
npGraph (bwa)	5.252	3	5,112.3	0	16.37	1.26 + 0.66
ngGraph (minimap2)	5.250	3	5,111.1	0	14.61	1.26 + 0.03
Streptococcus suis BM407						
SPAdes	2.119	81	131.0	0	3.84	0.59
SPAdes-Hybrid	2.147	48	1,438.0	0	0.98	0.65
Unicycler	2.171	2	2,146.2	0	2.99	2.58
npScarf	2.220	4	2,120.0	9	97.20	0.59 + 0.31
npScarf_wag	2.245	4	2,128.3	3	89.64	0.59 + 0.31
npGraph (bwa)	2.167	6	2,146.7	0	26.77	0.59 + 0.21
ngGraph (minimap2)	2.167	6	2,146.2	0	22.53	0.59 + 0.01
Acinetobacter AB30						
SPAdes	4.134	265	42.5	0	3.23	0.95
SPAdes-Hybrid	4.287	49	3,308.0	0	5.04	1.84
Unicycler	4.333	1	4,333.0	1	6.95	5.27
npScarf	4.595	11	4,299.7	1	120.99	0.95 + 0.45
npScarf_wag	-	-	-	-	-	-
npGraph (bwa)	4.317	6	2,766.9	1	39.82	0.95 + 0.41
ngGraph (minimap2)	4.337	1	4,336.8	0	24.71	0.95 + 0.03

Table 4.1 shows evaluation results for 5 synthetic datasets, the output of the full run can be found in Table C.1. In the first column of applied methods, beside `Unicycler` and `hybridSPAdes`, `npScarf` is included as the original scaffolder (described in Chapter 2) and `npScarf_wag` is the modified version with assembly graph integrated (this Chapter, Section 4.2). On the other hand, `npGraph` can use 2 different aligners, `BWA-MEM` and `minimap2`, for its bridging phase thus both practices were included in this comparison.

In general, the graph integrated version of npScarf improved the assembly results in terms of misassemblies and error reduction while virtually consuming similar resources compared to the original version. The only exception where the number of misassemblies being increased was the simulation of a random sequence with many repeats. There were 10 more mistakes detected using the later version npScarf_wag. However, with additional investigations, we found that the number of misassemblies on the true positive circular sequences (3 from the reference) has been significant reduced by applying assembly graph for npScarf. The errors mostly came from redundant sequences output from the software due to the failure in estimation of contig multiplicity. Even though using assembly graph for gap fillings, there were no changes in the way to determine if a contig is unique or not from npScarf_wag. It still relied on the length and coverage statistics, *i.e. Astats* [78] to find anchors that were critical for the backbone construction of the assembly. Redundant path findings for false positive replicons consequently returned additional wrong translocations in the final contigs which were reported by QUAST. Other than that, the method had successfully produced better assemblies than the original. Regarding *Mycobacterium tuberculosis* H37Rv, the number of misassembled breakpoints had been trimmed down from 12 to 2, while the number of final contigs had reduced from 4 to only one as in the reference. Results in cases of *E. coli* O25b H4ST131 and *Streptococcus suis* BM407 also showed enhancements in terms of those categories as well as N50 statistics. There was improvements considering the nucleotide errors (mismatches and indels) as well from aforementioned datasets, except for the *E. coli* when slightly more mismatches had been detected. However, these errors can be corrected by running polishing tools with the raw Illumina data afterward.

For *Acinetobacter* AB30 synthetic data, it was an deficiency for npScarf_wag in traversing the graph to find candidate paths for a bridge of long distance due to particular large search space. The exhaustive, naive DFS implementation for this version of npScarf required a lot of memory to traverse a complex assembly graph that usually exceed a normal desktop's capacity. This issue has been fixed in npGraph when Algorithm 2 was used on the definitive graph. This resulted in completed runs of the assembly process for all datasets with the similar number of misassemblies compared to the best figures in this category. As shown in Table 4.1 and C.1, the assembly graph based methods offered significant improvements when

compared to `npScarf`. Not only because of the clear drops with respect to misassemblies and errors, but it was also reflected by the number of final contigs and their N50 as well.

To align the long reads to the assembly graph components, either `BWA-MEM` [41] or `minimap2` [44] was invoked in `npGraph`. The former option was inherited from `npScarf` pipeline with the intact parameters while the latter was used with the recommended settings (`-k15 -w5`) for the best sensitivity working on MinION data. Even though, `BWA-MEM` normally reported more hits than `minimap2` but at the same time, was responsible for more false positive alignments. For instance, regarding the last dataset from Table 4.1, the assembly of `npGraph` using `BWA-MEM` was suffered from the ambiguous alignments thus more fragmented than the other counterparts. On the other hand, referring to more complicated graphs from *Acinetobacter* A1 and the yeast *Saccharomyces cerevisiae* S288c from Table C.1, the number of misassemblies from using `minimap2` were increased due to the lacks of appropriate alignments to support accurate bridging process. However, under almost circumstances, using either aligners would result in final assemblies with similar qualities. Furthermore, in terms of running time and resources required, `minimap2` proved to be the best option. The total CPU hours had been trimmed down drastically with the new aligner, making `npGraph` the fastest hybrid assembler available. This feature is certainly more favoured to a real-time assembly as well. As the consequence, as long as `minimap2` is expected to replace `BWA-MEM` for long-read sequencing data alignment, it would likewise become the main aligner for `npGraph` pipeline in the future.

It is noteworthy to discuss in more details the errors addressed in the above assembly methods. This figure measured the total mismatches and indels per 100kpb from the assembly sequences when mapping to the reference. As expected from hybrid assemblies where Illumina sequencing data were used as the main building blocks, the figures were hardly bigger than 100 (equivalent to 0.1% error rate) for almost every case. In addition, the indels errors, which mainly caused by TGS data, were found relatively low in the final contigs (Table C.1). The majority of the differences accounted for the mismatched nucleotides caused by the alternative paths connecting the unique anchors from the backbone of the assembly. This phenomenon may root from homologous repeats or sequencing errors of the genome. From all the hybrid assemblers, `hybridSPAdes` reported results with highest fidelity. This

meant that the performance its decision-rule algorithm exSPAnder [155] was the most accurate amongst all path finding methods. As the trade-off, there were fewer connections satisfying its quality threshold, resulting in the fragmented assemblies in cases of *Streptococcus suis* or *Acinetobacter* samples (Table 4.1 and C.1). Unicycler, which employs an algorithm based on semi-global (or glocal) alignments [162] with the consensus long reads, returned the second best reliable and at the same time, closest-to-complete results overall. npScarf, on the other hand, exploited the long reads for the gap filling thus inherited the high error rates from them. By integrating the assembly graph for the task, the errors were reduced in general (random sequences, *M. tuberculosis*, *S.suis* from Table 4.1) but not completely since the mis-placed contigs were still not resolved in other circumstances. npGraph significantly reduced the errors compared to npScarf, however the figures were still higher than the those of the best counterparts. This implied a more robust decision making system is needed in npGraph's real-time path finding module for even better output's accuracy.

4.3.3.2 Hybrid assembly for real datasets

Several sequencing datasets of actual bacterial samples [140] were used in this scenario. The data included both Illumina paired-end and MinION sequencing based-call data for each sample. Unlike previous settings, rather than the default output, the optimal SPAdes assembly graph detected by Unicycler were used for npGraph algorithm as well. This step had been proved to be useful in selecting the best graph available, amongst SPAdes runs with ranging k-mer values, that have dead-ends and number of contigs minimized [87]. Likewise, as mentioned before, the quality of the initial assembly graph would considerably influence the final results of npGraph.

Due to the lack of available reference genomes, fewer statistics were reported by QUAST for the comparison of the results. Instead, we investigated the number of circular sequences and PlasmidFinder 1.3 [144] mappings to obtain an evaluation on the accuracy and completeness of the assemblies. Table 4.2 shows the benchmark results of npGraph (using minimap2) against Unicycler on three datasets of bacterial species *Citrobacter freundii*, *Enterobacter cloacae* and *Klebsiella oxytoca*.

From the first dataset, there was high similarity between final contigs generated by two

TABLE 4.2: Assembly of real datasets using `Unicycler` and `npGraph` with the optimized `SPAdes` output. Circular contigs are highlighted in **bold**, fragmented assemblies are presented as X|Y where X is the total length and Y is the number of supposed contigs making up X.

	`Unicycler`	`npGraph`	**Replicons (based on `PlasmidFinder` 1.3)**
Citrobacter freundii	**5,029,534**	**5,029,486**	Chromosome
CAV1374	**109688**	**109688**	IncFIB(pHCM2)_1_pHCM2_AL513384
	100,873	**100,873**	IncFIB(pB171)_1_pB171_AB024946
	85,575	**85,575**	IncL/M(pMU407)_1_pMU407_U27345
	43,621	**43,621**	repA_1_pKPC-2_CP013325
	3,223	**3,223**	-
	1,916	**1,916**	ColRNAI_1__DQ298019
	14,464|3	14,456|2	-
Enterobacter cloacae	4,806,666|2	4,858,438|2	Chromosome
CAV1411	**90,451**	90,693|2	IncR_1__DQ449578
	33,610	**33,610**	repA_1_pKPC-2_CP013325
	13,129|2	14,542|4	-
Klebsiella oxytoca	6,153,947|5	**6,155,762**	Chromosome
CAV1015	**113,105**	**113,105**	IncFII(SARC14)_1_SARC14_JQ418540; IncFII(S)_1__CP000858
	111,395	**111,395**	-
	108,418	109,209|13	IncFIB(K)_1_Kpn3_JN233704
	76,183	**76,186**	IncL/M(pMU407)_1_pMU407_U27345
	11,638	11,892|2	-

assemblers. They shared the same number of circular ultimate sequences, including the chromosomal and other six replicons contigs. The only divergence lied on the biggest sequence ($\simeq$ 5.029Mbp) when the `Unicycler`'s chromosome was 48 nucleotides longer than that of `npGraph`. Five out of six identical replicons were confirmed as plasmids based on the occurrences of appropriate Origin of replication sequences (PlasmidFinder database). In detail,

two megaplasmids (longer than 100Kpb) were classified as IncFIB while the other two mid-size replicons, 85.6Kbp and 43.6Kpb, were incL and repA respectively, leaving the shortest one with $2Kbp$ of length as ColRNAI plasmid. The remaining circular sequence without any hits to the database was 3.2Kbp long suggesting that it could be phage or newly replicon's DNA. Lastly, there were still 14.5Kbp unfinished sequences resulted in 3 linear contigs from `Unicycler` and 2 for `npGraph` respectively.

The assembly task for *Enterobacter cloacae* was more challenging as the chromosomal DNA sequence was not been fully resolved using either method. The chromosome size was estimated to be approximately 4.8Mbp but had been broken into two smaller pieces. `npGraph` returned longer stretches of length 3.324Mbp and 1.534Mbp while the figures were 2.829Mbp and 1.978Mbp from `Unicycler`'s output. However, the number of circular sequences detected by `Unicycler` was one more than the other (2 versus 1). They were corresponding to 2 plasmids, namely IncR and repA. While the latter were recognized by both methods, the longer plasmid sequence was fragmented running `npGraph`. Similar to the previous dataset, there were around 14Kpb of data were unable to be finished by the assemblers.

Finally, assembly for *Klebsiella oxytoca* saw fragmented chromosome using `Unicycler` but it was a fully complete contig for `npGraph` with 6.156Mbp of size. The two assemblers shared 3 common circular sequences where two of them were confirmed plasmids. The first identical sequences represented a megaplasmid ($\simeq$ 113Kbp) with two variations of IncFII's origin of replication DNA being identified. The other agreed plasmid were IncL/M with 76Kbp of length. Particularly, there was one circular contig with length greater than 100Kbp but returned no hits to the plasmid database, suggesting the importance of *de novo* replicon assembly in combination with further interrogations. `Unicycler` detected another megaplasmid of size 108.4Kbp which was fractured by `npGraph`. The dissolution was also observed in `npGraph` for the final contig of length 11.6Kbp where it failed to combine two smaller sequences into one.

In addition to what presented in Table 4.2, dot plots for the pair-wise alignments between the assembly contigs were generated and can be found in Appendix Figure C.1. Interestingly, beside all other agreements, there was a structural difference using two methods for *E. cloacae* CAV1411 genome assembly. This was caused by the inconsistency of a fragment's

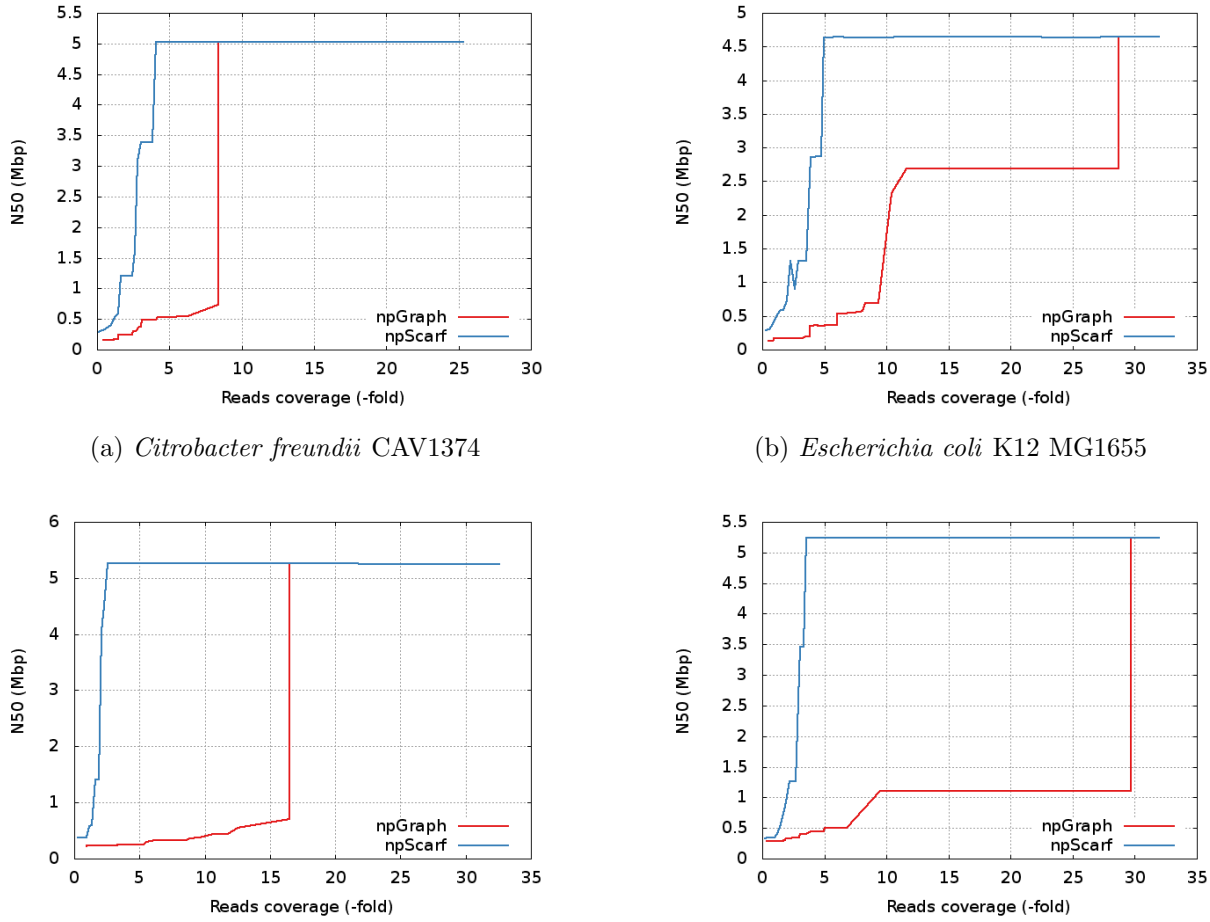
direction on the final output contigs. However, when compare to a reference genome of the same bacteria strain (GenBank ID: CP011581.1 [163]), contigs generated by `npGraph` demonstrated a consistent alignment which was not the case for `Unicycler` results (Appendix Figure C.2). Even though this might reflect a novel variation between bacterial samples of the same strain, it was more likely a misassembly by using `Unicycler`.

Overall, by testing with synthetic and real data, `npGraph` proved to be able to generate assemblies of comparative quality compared to other powerful batch-mode hybrid assemblers, such as `hybridSPAdes` or `Unicycler`. Furthermore, similar to `npScarf`, it has the advantage in term of supporting real-time assembly. The next section will address this utility and the interactive GUI bundled in `npGraph`.

4.3.3.3 Real-time mode hybrid assembly

Figure 4.5 demonstrates the real-time mode performance of `npScarf` and `npGraph` via N50 statistics during the assembly of 4 example datasets. This experiment would discover the rate of completing genome assemblies of the new method, set aside the accuracy aspect which had already been discussed previously. `npScarf_wg` basically scaffolds the pre-assembly contigs in the same manner with the original version thus was not discussed here.

As can be observed from all the plots, `npGraph` and `npScarf` both converged to the same ultimate completeness but with different paces and patterns. Apparently it took more data for `npGraph` to finish the same genome than the other. The reason stems from the fact that the new algorithm implemented a more 'conservative' approach of bridge construction with at least 3 supporting long-reads for each to prevent any potential mis-bridging. Unlike `npScarf` when the connections could be undone and rectified later if needed, a bridge in `npGraph` will remain unchanged once created. The plot for *E. coli* data clarifies this behaviour when a fluctuation can be observed in `npScarf` assembly at $\simeq$ 3-folds data coverage. On the other hand, the N50 length of `npGraph` is always a monotonic increasing function. The sharp 'jumping' patterns suggested that the linking information from long-read data had been stored and exploited at certain time point decided by the algorithm. Once a unique path has been determined, the bridge can be formed to connect the fragments together into longer sequences.



(a) *Citrobacter freundii* CAV1374

(b) *Escherichia coli* K12 MG1655

(c) *Klebsiella* 30660 NJST258

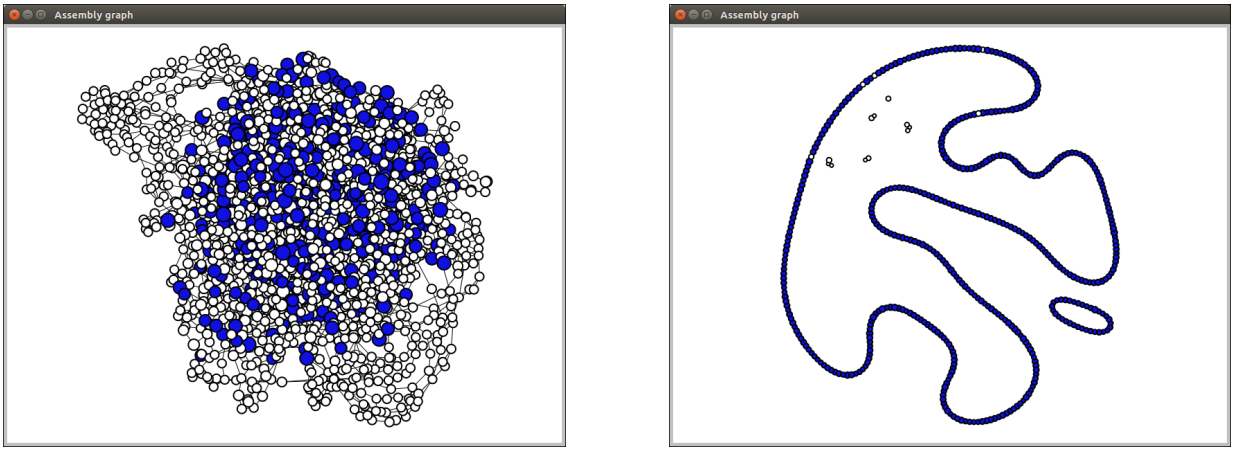
(d) *Klebsiella* NTUH K2044

Figure 4.5: N50 statistics of real-time scaffolding by `npScarf` versus `npGraph`.

Figure 4.6 shows an example of the real-time graph resolving process being displayed on GUI. The result graph, after cleaning, would only report the significant connected components that represents the final contigs. Smaller fragments, even unfinished but with high remaining coverage, are also presented as potential candidates for further downstream analysis. Further annotation utility can be implemented in the future better monitoring the features of interests as in `npScarf`.

4.3.4 Conclusions

Assembly graph is the data structure describing the assembly process at a lower level of more details. The current chapter brings this informative knowledge to the original `npScarf`'s



(a) Initial graph

(b) Resolved graph

FIGURE 4.6: Assembly graph of *Shigella dysenteriae* Sd197 synthetic data being resolved by `npGraph` and displayed on the GUI Graph View. The `SPAdes` assembly graph contains 2186 nodes and 3061 edges, after the assembly shows 2 circular paths representing the chromosome and one plasmid.

mechanism in two ways, either partly for the gap-filling module only (`npScarf_wag`) or completely as the building block of the assembly construction method (`npGraph`).

While the first approach showed improvements in terms of base's accuracy for the final results, it had limited ability in rectifying the misassemblies from the original version due to false positive alignments at contig level. The issue can be tackled by applying the assembly graph completely as in `npGraph`. With a more conservative bridging method applied, the new method may consume more data to confidently construct the assembly but at the same time, the number of mis-located fragments has reduced significantly. Furthermore, users can monitor the assembly in an interactive way providing the assembly GUI.

Compared to the other hybrid assemblers of similar methodology, such as `hybridSPAdes` and `Unicycler`, there are still rooms for further development of `npGraph`. More comprehensive pre-processing step is needed for a better input graph possible since it would affect the completeness of final results. Importantly, a robust real-time voting system should be implemented to be able to detect the most probable path amongst all candidates in an efficient way. This would alleviate the data consumption of the `npGraph` while at the same time, maintain high confident and accurate scaffolding.

5

MinION sequencing analysis for viral genomes

A virus can change the fate of the world; power has nothing to do with being tiny or giant!

–Mehmet Murat Ildan

This chapter describes another MinION sequencing application for short circular genomes, *e.g.* bacterial plasmids or viruses. The text emphasizes the relevant computational methods of the application. This study exploits the excessive length of nanopore reads and their potential to harbor multiple copies of small-sized viral DNA sequences. These genomes, given sufficient coverage depth, can be reconstructed by a high-resolution consensus calling of the single molecule Even though this study does not present real-time results from the device, it is straight-forward to adapt such pipeline using the developed modules since they operate rapidly in a read-by-read manner.

The analyses reported in this chapter come solely from my contribution to a joint project studying plant viral genomics. The lab works, including but not limited to DNA extraction, amplification as well as library preparation and sequencing steps were established independently thus will not be described in details. The data has not yet been published but permission to document part of the preliminary research findings for this thesis has been granted by all collaborators.

5.1 Introduction

There have been attempts to sequence bacterial plasmids using MinION device as part of a whole genome assembly [136, 138] or in exclusively studies [139, 164]. As mentioned earlier, studying the whole structure of *replicons* is important as they are responsible for the horizontal gene transfer between strains, *e.g.* the dissemination of AMR factors in *superbugs*. At the same time, it is critical to understand the mechanism of viral transmission and evolution in real-time to monitor and control epidemics [165–168]. As a result, nanopore sequencing for viral samples has already been employed in clinics or even in the field during outbreaks, *e.g.* influenza [169], Ebola [170] or Zika [171] viruses.

The ONT MinION benefits sequencing short genomes, such as viruses, in many ways especially in terms of genome assembly. In fact, we regularly obtain nanopore reads that cover the whole length of the circular IncP-α plasmid RP4 (around 60kpb long) genomes [164], which significantly assist in the assembly phase. Even without associated Illumina data, nanopore data can generate results with decent quality after consensus calling thanks to

sufficient high coverage [50]. The multiplexing method is usually applied to reduce the costs and increase the scale of microorganisms study of interests [171].

Here another approach is presented for viral sequencing using ONT platform. A special amplification technique is employed to increase the copy numbers of the whole circular DNA molecules before being subjected to the sequencing step by MinION and relevant computational task for subsequent assembly.

Rolling-circle DNA amplification PCR is the most widely used amplification method for many organisms in general but for this study, we employed different whole genome amplification technique in order to obtain longest sequences possible for MinION sequencing. The cloning method is known as rolling-circle amplification (RCA), a one-step whole-genome multiplication that has been applied for small circular DNA molecules, especially virus families [172–178].

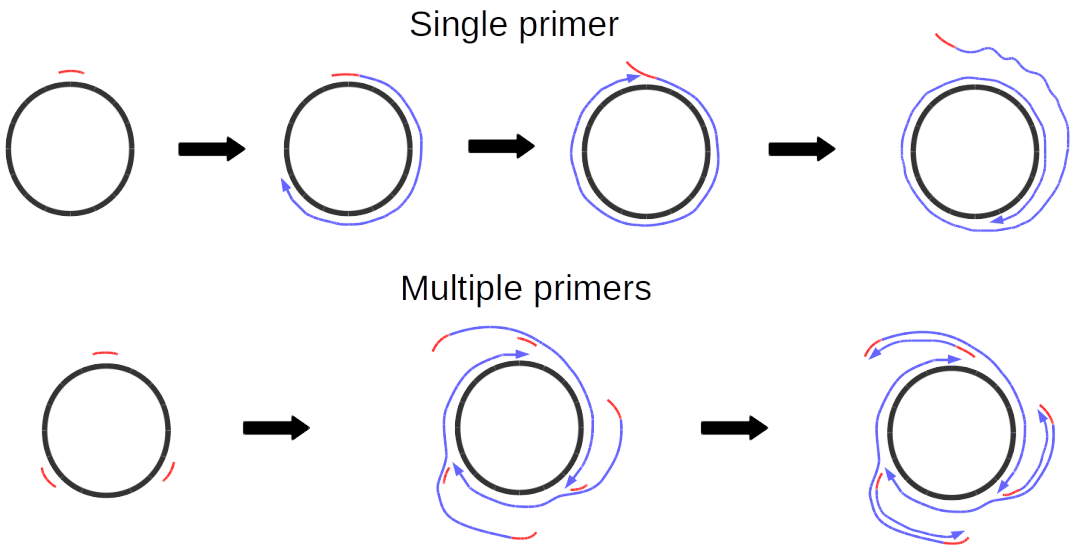


Figure 5.1: General mechanism of Rolling-Circle Amplification. Figure is adapted from [179] under Elsevier license (attached at the end of this thesis). Primer sequences are highlighted in red, the circular genome subjected for amplification is in black and the synthesized clones are in blue.

The basic principle for RCA is shown in Figure 5.1 [179]. On top is an example of the process with only one imaginary primer binding to a template circular. The synthesis starts from this point and move across the whole length of the molecule in the complementary direction as being guided by the DNA polymerase *phi29* of bacteriophage *Baccilus subtilis*.

Due to this particular enzyme's features, the incorporation can continue passing the binding site whilst the newly synthesized strand being displaced from the track. Depending on the size of the template circle, the copying process can go several rounds, resulting in elongated molecules consisting of multiple copies of the original. From the diagram in the bottom of Figure 5.1, multiple random primers are used for the amplification as it should usually be in practice. In this case, the abundance of primers from the reaction mixture can bind to the displaced strand and trigger additional syntheses, creating the branching patterns of the cloning process.

The RCA products are called concatemers as they are long concatemeric molecules consisting of consecutive copies of the target DNA sequence. Normally, restriction enzymes are used to chop them into separated monomers before taking further steps. For our method, concatemers are directly sequenced by MinION and computational methods are used afterward to detect such patterns.

Viral samples The viral genomes used in this study belonged to the plant infectious family *Caulimoviridae*, or *caulimovirids* with size varied around 7-9Kbp. Amongst nine genera detected [180, 181], two of them were investigated, namely *Badnavirus* and *Caulimovirus*, represented by *Banana streak MY virus* (BSMYV) and *Cauliflower mosaic virus* (CaMV) respectively.

5.2 Bioinformatics analyses

5.2.1 Data description

A multiplex sequencing has been conducted for 4 samples with the Rapid Barcode Sequencing kit. The barcode assignment is given in Table 5.1.

5.2.2 Reference-based detection of concatemers

Raw signal data from MinION were base-called and demultiplexed by `Albacore` version 2.1.0. This resulted in 4 DNA sequence files corresponding to 4 barcoded samples, only pass

Table 5.1: Viral samples subjected to MinION barcoding sequencing

Barcode	Sample	Description	Pass reads	N50
08	CaMV+	*Cauliflower mosaic* virus	14,385	6,707
09	BSMYV+	*Banana streak MY* virus	3,389	8,178
10	BSMYV-	*Banana streak MY* virus, negative control	4,653	6,023
12	CaMV-	*Cauliflower mosaic* virus, negative control	9,942	4,919

reads from those sequences were used for further analyses.

Work flow Due to the application of rolling circle amplification, each long read is expected to contain more than one copy of the virus DNA, potentially sitting next to each other in a *concatemer*. We create an in-house pipeline to detect and extract the *monomer* sequences out of the reads to build their consensus as shown in Figure 5.2.

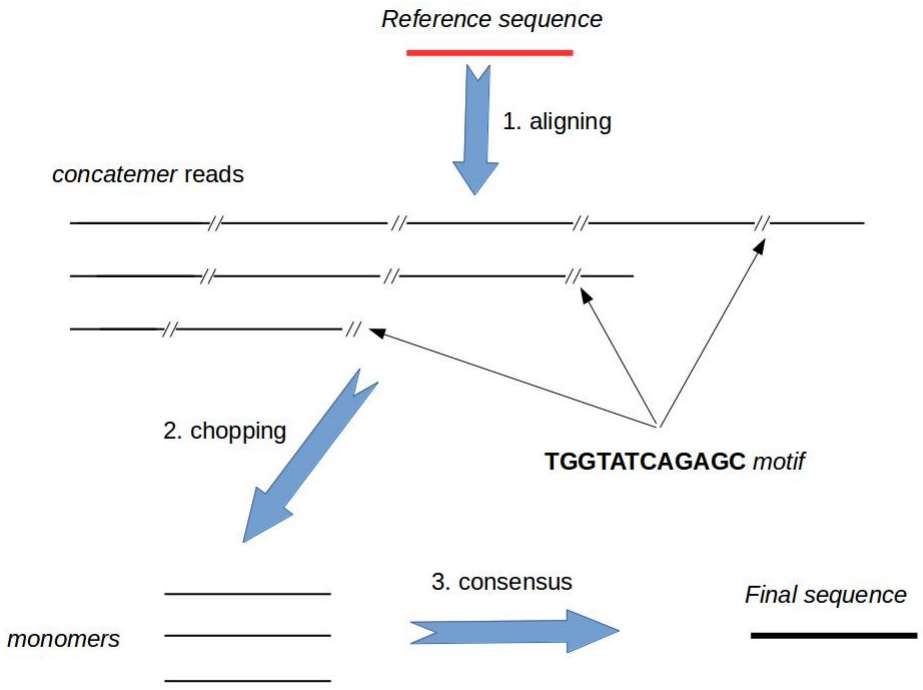


Figure 5.2: Pipeline to reconstruct viral genome sequences from MinION long reads.

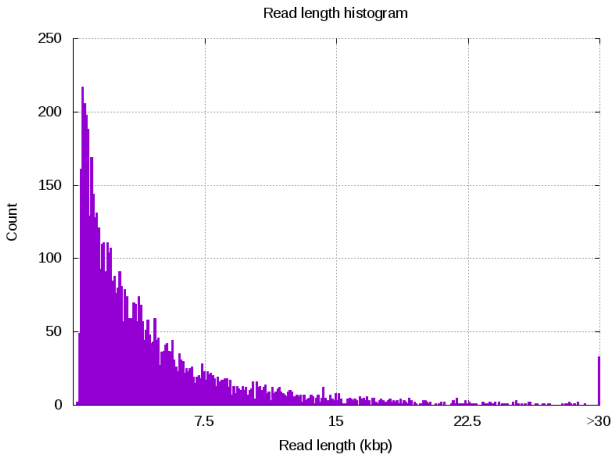
In more details, the pipeline is implemented in the following steps:

1. align nanopore reads to the corresponding reference by `minimap2` [44] version 2.11-r797 and keep only the ones covering $> 80\%$ of the target; then induce locations of monomer on each reads based on the alignments.

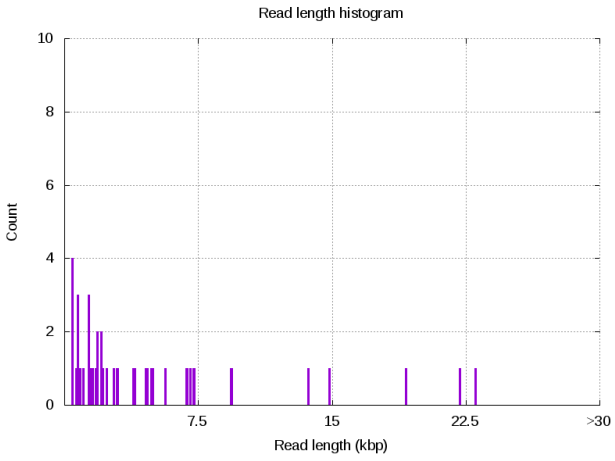
2. for each monomer inferred, scanning for the nearest hit to the tRNA^{Met} 12 nucleotides motif **TGGTATCAGAGC** where the DNA synthesis is primed in *Caulimovirids* replication cycle [182, 183]. Those loci are then used as final breakpoints for a later chopping step to extract monomers.

3. build consensus sequence on those extracted monomers by `Racon` [50] version 1.2.1.

Align to the reference Figures 5.3a and 5.3b show the read length histogram of only mapped reads from nanopore data of barcode 08 and 09 respectively. Two negative control samples (barcode 10 and 12) returned no hits when aligned to their corresponding reference thus not shown. As can be observed, CaMV virus (barcode 08) has richer sequencing data compared to BSMYV sample (barcode 09).



(a) Mapped reads from barcode 08



(b) Mapped reads from barcode 09

FIGURE 5.3: Mapped read length histogram of barcode 08 and 09.

Concatemer extraction Chart D.1 presents the number of k-concatemers (concatemers with exact k monomers extracted) for each positive sample (barcode 08 and 09). The longest

concatemer detected was a 7-concatemer of CaMV DNA sequence. There was also one 6-concatemer and one 4-concatemer from this sample. In addition, the numbers of detected k-concatemers for *Cauliflower mosaic* virus were more than 1 for $k = 3$ (2 reads) or $k = 2$ (6 reads). Monomer reads made up the most abundance group with 33 CaMV sequences and 5 BSMYV sequences. There were no other concatemers detected for the latter sample.

The extraction step generated in total 68 and 5-folds coverage of monomers respectively for barcode 08 and 09. By using these monomer sequences, we can pile them up and call the consensus sequence out of the nanopore data. The obtained sequence coverage played a critical role for the accuracy of the result: for barcode 08, the consensus was 99.5% identical to its reference while it was only 93.72% in case of barcode 09.

5.2.3 Reference-free method

Assuming no references are given, the duplication of a DNA sequence in a nanopore read can still be detectable by using self-alignment to study the repeat pattern. A proposed approach to investigate periodic repeat patterns is to use auto-correlation function (ACF) in digital signal processing (DSP) technique. Similar DSP-based methods have been developed for fast biological sequence alignment and repeat detection [184, 185]. Here I present another application of signal processing techniques to detect the concatemers of the aforementioned viral sample using nanopore data. Tools are developed to work on both base-called and raw signal sequences. For better demonstrating purpose, analysis result from the longest 7-concatemer read (from barcode 08 sample) is described without loss of generalization.

5.2.3.1 Auto-correlation function

Given an infinite sequence of discrete signals $S = \dots s_1 s_2 \dots s_k \dots$ where $S(i) = s_i$, its ACF $f(n)$ is defined as the cross correlation to itself

$$f(n) = \sum_k s_k * s_{k-n} \tag{5.1}$$

where $*$ operator returns a similarity score when compare two operands, *e.g.* complex conjugate for $s(i) \in \mathbb{C}$ [184], and n stands for the lag value. By increasing this value, we

have a sliding dot product between the signal vector with itself which in expect give peaks when repeat parts of the sequence are overlapped as demonstrated in Figure 5.4. Note that for every signal sequence S, there is always a peak at $n = 0$ as a result from self-overlapping. In addition, the ACF values are symmetric with regard to this center value.

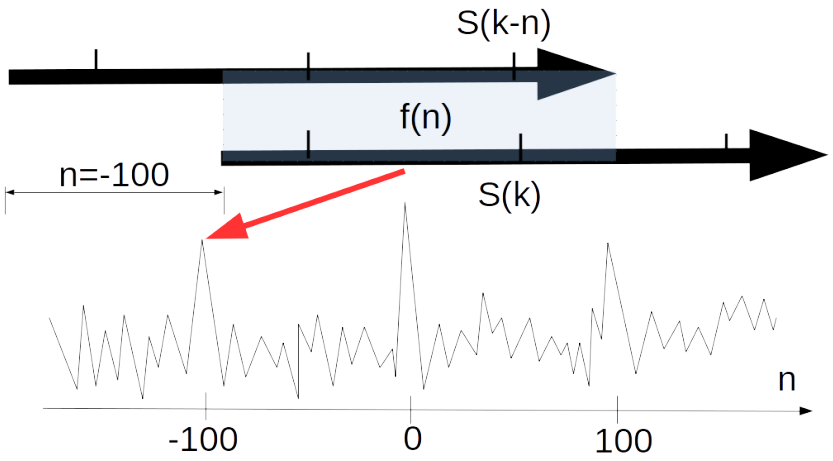


Figure 5.4: Example of ACF sliding dot product for a sequence with two repeats. The plots shows 3 dominating peaks corresponding to $n = -100, n = 0, n = 100$.

In fact, the signal sequence to process is definite: $S = \{s_i\}, i = 1 \dots L$ for sequence with length L. Out of range values are normally zero padded $s_j \equiv 0 \, \forall j \leq 0, j > L$ or duplicated so that $s_i \equiv s_{i+n \times L} \, \forall n \in \mathbb{Z}$. In this study, the former filling method is used.

The following content will focus on ACF-based attempts in determining the concatemeric pattern of nanopore long reads. Firstly, in the next section, a straightforward strategy will be employed directly on the base-called sequence of nucleotides. The sliding dot product algorithm for the time-domain signal, together with a simple filter will be implemented initially. This simple but applicable method would serve as a proof-of-concept for the idea of using DSP technique to the problem. After that, we introduce a more robust method that allows fast calculations not only on the DNA sequences, but also on the raw squiggle signals from the underlying ONT sequencing process. The ability to operate quickly per read would enable the concatemers detection to work in a streaming fashion so that it can be integrated in a real-time pipeline as aforementioned.

5.2.3.2 Concatemers detection protocol using DSP

To apply DSP algorithm, the first step is to convert the nucleotides sequence into appropriate digital signal. Given a DNA sequence, we have $S(i) = s_i \in \{A, C, G, T\}\ \forall i \in \mathbb{N}$. A straightforward conversion is to map letters to their corresponding numerical values, *e.g.* $\{1, 2, 3, 4\}$ respectively. Consequently, the $*$ operator from Equation 5.1 can be simply adapted to the Kronecker delta function

$$s_i * s_j \equiv \delta_{s_i,s_j} = \begin{cases} 0 & if\ s_i = s_j \\ 1 & if\ s_i \neq s_j \end{cases}$$

As shown in Equation 5.2, the ACF values are normalized by the overlap length to alleviate the position-dependant behaviour of the function that is important to locate peaks that represent monomer overlaps in next step. Due to the errors (indels/mismatches) of MinION sequence data, these overlaps are not aligned perfectly as of one-to-one mapping of base pairs but rather scattered hits along the overlapped region. This results in a 'group of nearby spikes' rather than a single prominent peak, making it more difficult to locate exact coordinates of monomers in the read. For that reason, a low-pass filter (LPF) is needed to help reduce the noises, or *smooth* the signal before further analysis. Primarily, we use *running average* with a fixed window size (ws) of nearby values for such task. We set the window size equal to 10 for this scenario.

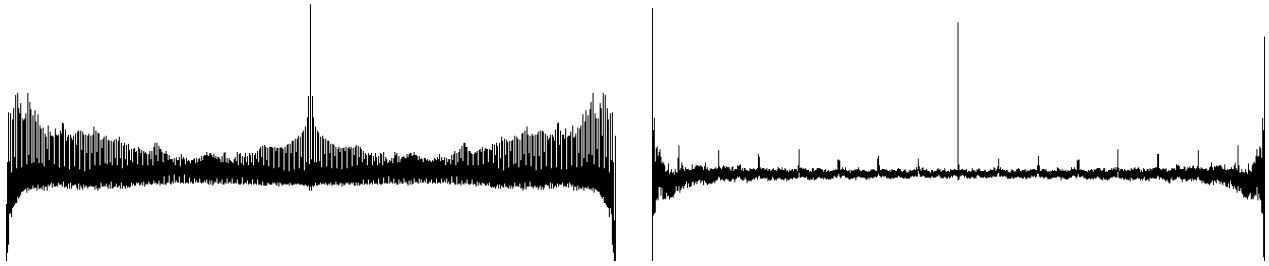
Overall, for this case, we investigate the signal of the normalized Kronecker delta function (NKDF) as shown in Equation 5.2

$$f(n) = \frac{\sum_{i=1}^{L} \delta_{s_i,s_{i-n}}}{L - |n|}, n \in (-L; L) \tag{5.2}$$

then apply the running average filter on it

$$\overline{f(n)} = \frac{\sum_{i=-ws/2}^{ws/2} f(n+i)}{ws}$$

Figure 5.5 presents the result signal for the detected 7-concatemer versus a random synthetic read with similar length. According to the plots, there is only one distinct peak



(a) ACF for a random DNA sequence

(b) ACF for the 7-concatemer read

Figure 5.5: ACF values for a random read (5.5a) versus the 7-concatemer detected (5.5b).

from the ACF signal of the random read which reflects the trivial case of self-alignment. On the other hand, from the longest concatemer read available, we can observe 7 clear peaks on each side of the symmetric squiggle representing the same copy number of the viral monomer. These peaks are possibly identified by a peak picking algorithm which can determine the significant local maxima located across the range with similar distances.

On the other hand, the formula 5.2 implies greater variance of the signal values toward the ends of the range $(-L; L)$ due to fewer overlap data points involved. This tailing issue would hinders the algorithm to locate the monomers at two ends of a read. This effect can be observed from Figure 5.5b where the squiggles become more diverse leaving further the center lag ($n = 0$). For that reason, the peak picker will start the scan from $n = 0$ to either side for distinct spikes of higher confident and determine the period before reaching the noisy peaks toward the ends. Figure D.2 from the Appendix gives more examples of ACF-induced signals for several other k-concatemers using this calculation method. Similarly, in all cases the fluctuations at the ends of ACF graphs are plotted beside the 'genuine' maxima. This introduces difficulties to detect 'real' hits close to the ends especially when $k = 1$ as we have no other peak for comparison.

In general, via above experiment, the DSP approach utilizing ACF-based estimation has been proved to be feasible to detect the concatemeric pattern of nanopore RCA reads. However, there are several points needed to be further developed and improved from this protocol to be able to work in an applicable pipeline. The most important task is to optimize the algorithm to reduce the running time. The current sliding method takes $\mathcal{O}(L^2)$ time for a read of length L. Even though it is relatively reasonable for viral concatemers of

medium length ($\simeq$ 50Kbp), it would become time-consuming to process reads with high copy numbers which are favoured and aimed for in this study. Also, the turn-around speed per read is critical in real-time analysis, not to mention the desired application at the level of sequencing raw signal from the very pre-basecalling stage. Another essential operation is a robust LPF mechanism since the simple moving average method is unable to comprehensively remove the high-frequency noises from the target signal. Lastly, a peak picking algorithm is needed to spot the periodical maxima and chop the repeat sequence at those breakpoints into monomers of interest. All of those steps will be addressed in the following section.

5.2.3.3 Rapid method to detect concatemeric signal

Generalized problem for real signal data. While Kronecker delta function is fast to calculate, it is only suitable for sequences of discrete, categorical values *e.g.* DNA letters. Nanopore raw signal for a read is given as a series of real values of electrical current sampled thousands times per second when the biological molecular transiting through the pore. Processing concatemers at raw signal level before base-calling would accelerate the whole pipeline to a new level, as well as improving the quality of signal for the next stage.

From this context, the sequence S, $S(i) = s_i \in \mathbb{R}\ \forall i$ would have ACF written as

$$f(n) = \sum_k s_k s_{k-n}$$

where the $\star$ operator in Equation 5.1 is replaced by the multiplicity in $\mathbb{R}$ instead of the previous Kronecker delta function for discrete values of $\mathbb{Z}$. Due to the symmetric property of ACF, only the right half of the range will be considered instead of the whole plot as demonstrated in Figure 5.5, corresponding with lag from 0 to $L - 1$. The trivial peak of self-alignment thus locates in the very first element of the array.

Rapid estimation method for normalized ACF signal Calculation of ACF function given above is straight-forward and efficient using Fast Fourier Transform (FFT) [186–188] with complexity $\mathcal{O}(L \log L)$. However, to additionally normalize the signal while maintaining the speed aspect of the algorithm is not a trivial task. To achieve such goal, the Normalized Square Difference Function (NSDF) [189] is implemented. This function can be calculated

as shown in Equations 5.3.

$$\begin{aligned} h(n) &= 1 - \frac{\sum_k (s_k - s_{k-n})^2}{\sum_k (s_k^2 + s_{k-n}^2)} \\ &= \frac{\sum_k 2 s_k s_{k-n}}{\sum_k (s_k^2 + s_{k-n}^2)} \\ &= \frac{2f(n)}{g(n)} \end{aligned} \tag{5.3}$$

The values of $h(n)$ would fall in $(0, 1]$ with $\forall n$, representing a normalized measure of proximity between S and its n-delay signal.

Algorithm 3: Algorithm to calculate the NSDF by using FFT.

Input: Time-domain sequence signal S of lenth L

Output: NSDF signal H

Function `ACF`(S, R)**:**
- $R := \texttt{fftForward}(S)$ `// Convert signal to frequency domain by FFT`
- $R[i] := R[i] \star R[i], \forall i = 0 \ldots (L-1)$ `// complex conjugate element-by-element`
- $R := \texttt{fftReverse}(R)$ `// Convert back to time domain by reverse FFT`
- **return**

Function `SS`(S, M)**:**
- $SS[i] = S^2[i], \forall i = 0 \ldots (L-1)$
- $M[0] = 2 * \texttt{sum}(SS)$ `// get sum of all elements in SS`
- $M[i] = M[i-1] - SS[i-1] - SS[L-i], \forall i = 1 \ldots (L-1)$ `// update incrementally`
- **return**

begin
- `ACF`(S,R) `// calculate ACF and assign to R`
- `SS`(S,M) `// calculate lag sum square and assign to M`
- $H[n] = \frac{2R[n]}{M[n]}, \forall n = 1 \ldots (L-1)$ `// calculate NSDF as in Equation 5.3`
- **return** H

More than that, this function is determined by $f(n)$ and $g(n)$ which can be both measured rapidly by using FFT (using `JTransform` [190], a Java library for DSP) and incremental

calculation as shown in Algorithm 3.

LPF with Blackman windowing function. The plain moving average method is helpful to show the trend of the overall signal but not sensitive enough to completely remove the noises encountered. Figure D.3 in the Appendix plots the NSDF of the 7-concatemer read with large smooth window sizes of 10, 000 and 20, 000. As a peak scanning method is sensitive to abundant of local optima, a 'smoother' transition is expected via a signal filtering step. For that reason, a finite impulse response (FIR) filter by windowing is implemented.

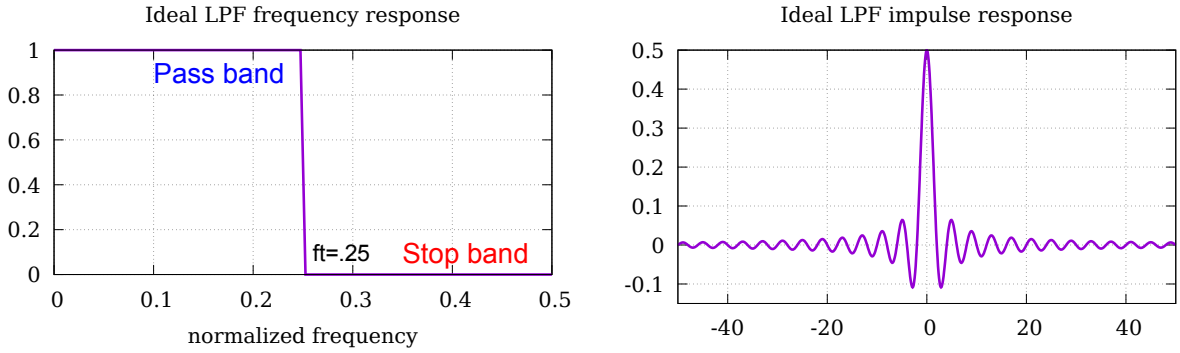


Figure 5.6: Example of ideal LPF at $f_t = 0.25Hz$ and its corresponding impulse response.

Figure 5.6 gives example for an ideal LPS with transition (cutoff) frequency $f_t = 0.25$. The left plot presents the frequency response of the filter in frequency domain (normalized with respect to the sampling frequency). In this perfect scenario, a brick-wall filtering is expected when any frequency values below $0.25Hz$ would pass and all higher components are stop. This system would have an equivalent impulse response as shown on the right plot, given by function `IRF`:

$$\texttt{IRF}(x) = 2f_t \; sinc(2\pi f_t x)$$

where $sinc(x)$ is the *sine cardinal* function

$$sinc(x) = \begin{cases} \frac{\sin x}{x} & if\ x \neq 0 \\ 1 & if\ x = 0 \end{cases}$$

In practice, to implement a finite impulse response LPF, only a window of the shifted sinc function is sampled for a finite number of filter weights. We set this window's length, or

filter length, as the length of signal L.

$$\texttt{IRF}(n) = \begin{cases} \dfrac{\sin\left[2\pi f_t(n - \dfrac{M}{2})\right]}{\pi(n - \dfrac{M}{2})} & if\ n \neq \frac{M}{2} \\ 2f_t & if\ n = \frac{M}{2} \end{cases} \tag{5.4}$$

where M is the filter order, defined as $M = L - 1$.

Furthermore, to reduce the ripple from the ringing artifact due to crude approach of truncating the infinite ideal impulse response [191, 192], we apply Blackman windowing [193, 194] instead of a rectangle window ($w(n) = 1$) on the IRF values.

$$w(n) = 0.42 - 0.5\cos\frac{2\pi n}{M} + 0.8\cos\frac{4\pi n}{M} \tag{5.5}$$

Overall, to apply the LPS with Blackman windowing on the NSDF signal, we follow the steps below.

(1) Calculate the weights of the LPF as a function of cutoff frequency f_t as in Equation 5.4

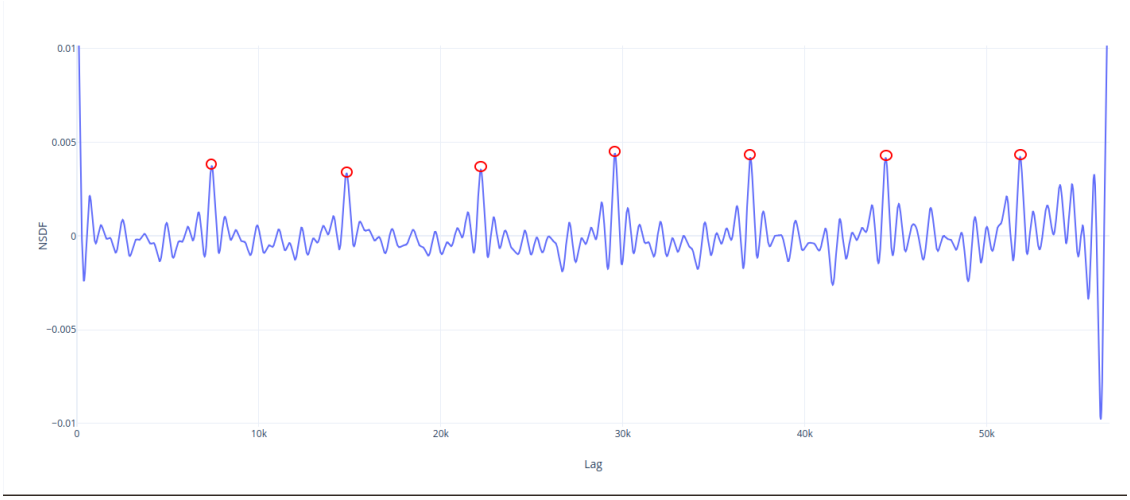
(2) Multiply the result with the windowing calculated in Equation 5.5

(3) Apply FFT for the result filter values.

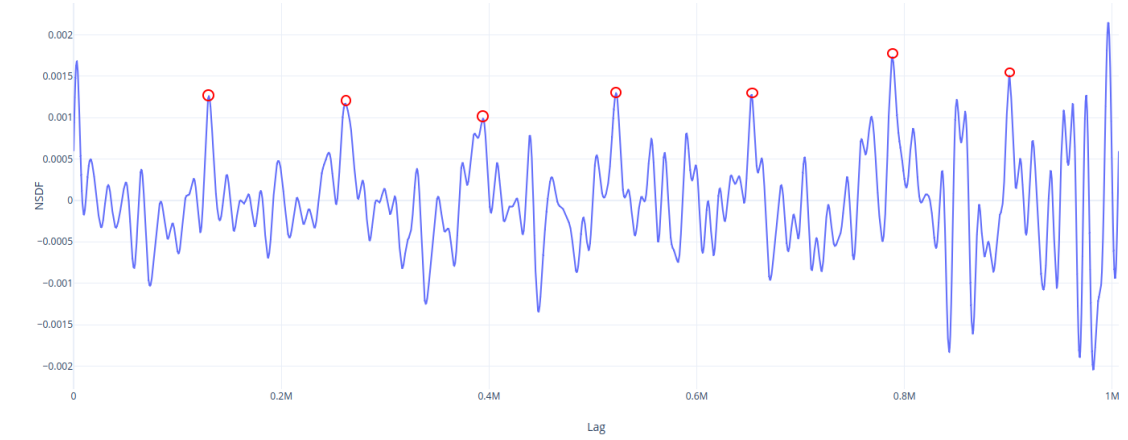
(4) Calculate FFT of the NSDF and multiply its values element-by-element to the filter values in the frequency domain.

(5) Calculate the reverse FFT to get the result in the time domain.

The cutoff frequency (before being normalized) is empirically set as $100Hz$ assuming that a 100-times RCA duplication is virtually impossible or due to process's artifacts. This threshold can be set more stringent with knowledge about specific reference genome size and obtained sequencing read length. Figure 5.7 shows the filtered signals for the only 7-concatemer read, at both pre- and post- base-calling stage. Additionally, Appendix Figure D.4 depict plots from applying the method with cutoff frequency of $30Hz$. As expected, the signal will become 'smoother' as the transition frequency decreased but would risk the specificity of the signal and generalization of the method when apply to other dataset. The filtered signal is now ready to be used as input for a pick peaking method to detect the repeat pattern of the concatemeric reads.



(a) DNA sequence



(b) Raw signal

Figure 5.7: Peak picking results for a 7-concatemers NSDF signal processed with LPF using $cutFreq = 100$ of (5.7a) the base-called DNA sequence and (5.7b) its corresponding raw signal.

Pick peaking algorithm A list of candidates are first selected out of all local maxima by choosing the highest ones locating between every positively sloped zero crossing and negatively sloped zero crossing [189]. By setting the cutoff frequency, a respective minimum period (or monomer length) is determined as a consequence: $minLen = \frac{1}{cutFreq}$. Any peaks on the left of this value will be ignored and the algorithm will iteratively scan to the right in the remaining peaks list for the correct first peak. The correct first peak at n_1 must satisfy criteria as below:

(1) There are at least $\lfloor \frac{L}{n_1} \rfloor - 1$ more peaks with similar distances along the signal sequence.

(2) The sum of these peaks' height must be more than a certain portion, *e.g.* 0.8, of the counterpart for all candidates.

The algorithm will stop at the very first coordinate that meets above 2 conditions and return the list of periodic peaks which can be used to break the concatemer into monomers.

Figure 5.7 also reflects the peak-picking results applied for the longest concatemeric read. In both signal level, we detect exactly 7 peaks (highlighted in red dots) corresponding to the breakpoints that can be used for a 7-concatemer chopping process. Interestingly, when the peaks from DNA signal shows very stable repeat pattern, there is a slight variance from raw signal toward the far end. This phenomenon is shown clearer in Appendix Figure D.4b when the sixth peak is shifted more to the right, considerably making the sixth monomer longer and the seventh monomer shorter than average. We argue that there should be an unsmooth transition of the squiggle signal reading from the sequencing process [195]. This is due to the nonuniform translocation time of the DNA molecule caused by the motor enzyme at the *homopolymeric* regions [196–198] and normally responsible for the high indels from nanopore sequencing [199–201]. However, the base-calling in this case has successfully manipulated the situation, resulting in more evenly-distributed peaks being detected.

Overall, via above example, we demonstrated that a 7-concatemer can be detected using the reference-free method based on NSDF signal investigation. Similarly, any k-concatemer ($k \geq 1$) can be identified by the same manner and its monomers can be extracted by chopping at respective break-points. However, in order to obtain results of high confident, only reads corresponding to $k \geq 2$ should be selected as they would show more than 2 peaks for an affirmative picking task.

5.3 Conclusion

This chapter demonstrated a method of using MinION sequencing together with RCA for small-sized circular genomes, in particular two samples from *Caulimovirids*. Results from this preliminary finding showed that concatemer molecules generated from RCA can be sequenced directly by the thumbnail sequencer without using restriction enzymes to physically divide them into monomers.

In fact, this step can be accomplished by using computational approaches, given or not a reference sequence. In the reference-based method, corresponding monomers' break-points for RCA reads could be detected straightforward, resulting in abundance of single-copied sequences that could be subjected to a consensus calling. For the *de novo* detection algorithm, concatemeric reads were more confidently identified if they contains higher number of monomer duplication. For that reason, we can apply the reference-free method on data from CaMV barcoded sample to extract monomers and run the consensus calling with 35-folds coverage. RCA for BSMYV, on the other hand, returned solely monomers which were exclusively detectable given the reference DNA. Further improvement from sequencing phase should be made in order to achieve longer stretches of clones so that the pipeline can operate completely without prior knowledge about the target genome.

The implemented module could operate rapidly read-by-read thus it is able to adapt for a streaming pipeline. For instance, a viral or plasmid sequencing would run until sufficient copies of monomers are produced for each sample. Furthermore, the whole process can be established at raw signal level, before the base-calling step. For reference-based algorithm, the alignment step can be carried out between sequences of squiggle signal, as described in *Read-Until* protocol [93]. In case a reference is not provided, an DSP implementation based on NSDF is available for the task of determining concatemers.

In summary, this approach provide another promising method to study relative small but highly varied genomes of microorganisms. For instance, the longer concatemeric sequences will offer superior resolution to the nucleotides of individual virus thus enhancing the knowledge about inter- and intra-sample variations of this rapid evolving species. Continuing efforts are made to improve the monomer copy numbers from RCA reads so that we can utilize the methods in more comprehensive ways for real-life applications.

6

Conclusion

In the end we come up with a conclusion that we need to start from somewhere.

–Deyth Banger

6.1 Thesis summary

The advent of Third-generation sequencing methods, particularly ONT platforms, have brought novel opportunities together with challenges to the field of bioinformatics similar to those arisen from the arrival of Second-generation sequencing methods before [202, 203]. Chapter 1 has provided a brief review as well as comparison between these two generations of sequencing methods, aiming at their applications in genome assembly. Specifically, the long spanning reads generated in real-time from ONT MinION device had been found to be useful in generating more complete assemblies despite of their error profiles [34, 35, 85].

Throughout this thesis, computational methods have been designed and applied to complete genomes as fast as possible. For that purpose, Chapter 2 described `npScarf`, a streaming hybrid assembler that could finish the fragmented short-read assemblies using the consecutive nanopore reads. In addition, we have shown that the pipeline could be used to determine the locations of highlighted genes of interests on the whole genome during its construction in real-time. To adapt the parallel mechanism of MinION using barcode sequencing in our streaming pipelines, `npBarcode` has been developed and its applications were mentioned in Chapter 3. The scalability of `npScarf`, as well as of other streaming analyses, to multiple samples were verified through this chapter. Chapter 4, on the other hand, focused on designs to help improve the accuracy of the assembly output. This resulted in `npScarf_wg` and especially `npGraph` which employed the assembly graph for better fidelity. Lastly, Chapter 5 conducted computational analysis on the MinION data of concatemeric long reads in an attempt to produce viral genome assembly in an efficient way.

6.2 Key contributions

Overall, the thesis focuses on genome assembly methods using the MinION data and how to exploit its real-time feature to reduce the turn-around time of the computational analyses. Accordingly, the main contributions fall into two categories as followed.

Contributions to genome assembly methods We offered a set of open-source software and utilities for genome assembly using Nanopore data. In combination with short-read

assemblies, `npScarf` can finish and return complete genomes in a short amount of time while consuming less resources compared with other methods. In addition, `npBarcode`, a robust multiplexer for barcode sequencing, can be used together with `npScarf` to scaffolding multiple samples at the same time.

We also addressed an use case of using a combination of `npScarf` with other prominent methods consuming the same inputs. Through this example, ones can measure the advantages and disadvantages of each method to come up with the best solution possible. Importantly, if short-reads assembly graph is provided, `npGraph` can be employed for more accurate results. It also comes with a GUI to monitor the process in an user-friendly manner.

Finally, we proposed another assembly method for small circular genomes using concatemeric MinION reads after RCA. Either a reference-based or reference-free assembly algorithm can be applied on the long reads to generate the monomers of whole genome sequences with decent quality given sufficient read coverage.

Contributions to real-time analysis To date, `npScarf` and its derived version `npGraph` are the only assembly methods exclusively designed to assemble genomes in real-time in accordance with MinION's output. We have successfully addressed, implemented and applied a streaming pipeline to rapidly finish short-read assemblies using real-time sequencing. As a result, it enabled real-time analyses that rely on positional information, including but not limited to identifying genes encoding bacterial genomic islands and plasmids. These functional regions in the bacterial genomes can be horizontally transferred between organisms, which is one of the main mechanisms for acquiring AMR in pathogenic bacteria.

Furthermore, the streaming demultiplexer `npBarcode` allows the pipeline to run in parallel for multiple samples. This feature is critical to clinical applications such as diagnosis and treatments, especially in emergency units, which normally require quick response from multiple testing activities. On the other hand, the ability to operate and to monitor completion status in real-time (even with GUI) allows users to decide when to stop the sequencing process thus fulfilling the task of saving time and cost for genome analyses. The reference-free concatemeric assembler from Chapter 5 can even work with raw signal from the sequencing device hence open up the possibility to be integrated into the *ReadUntil* pipeline [93] which

had been designed to further reduce the turn-around time of the nanopore data analyses.

6.3 Future directions

Improve software performance The thesis has shown continuous development of a real-time hybrid assembler. It has evolved from the original `npScarf` to a version with assembly graph (`npScarf_wag`) and ultimately an exhaustive graph-traversing algorithm (`npGraph`). The efforts indeed have leveled up the accuracy of the assembly output considerably. Even though, there are still rooms to enhance the performance of the software given the inputs kept intact. Specifically, there is a need for more robust voting system and/or a decision making mechanism to be able to determine bridges with higher confidence in real-time. Other than that, a revertible bridge construction on the graph is also considered for a self-rectifying assembly mechanism of `npGraph`.

`npScarf`, on the other hand, performs better with mid- or large-size datasets. However, an optimization is required to minimize the resources required when working with huge and complicated genomes, *e.g.* of eukaryotic organisms. Besides, `npScarf` and `npGraph` are currently working as two separate modules due to different methodology and implementation. For convenience, they will be integrated in one common interface of a hybrid assembly software supporting different scenarios.

Application in metagenomics Preliminary works on mock datasets of metagenomics using `npScarf` have demonstrated that our approach is feasible for this case (data not shown). As a consequence, one of the development direction is to implement a hybrid assembler that can solve metagenomics assembly graph in real-time with `npGraph`.

In order to achieve this goal, we first plan to run `SPAdes` with `--meta` option [204] to generate a metagenomics assembly graph. After that the contigs (nodes from the graph) will be subjected to a binning algorithm, *e.g.* `metaBAT` [205], to discover the populations of the community. This information is then used to determine multiplicity and membership of each contigs so that `npGraph` algorithm can work with and produce assembly for each of those detected populations.

Non-hybrid methods It is desired to have a non-hybrid real-time *de novo* assembly as the yield and quality of the nanopore reads being generated are increased significantly compared to the early stage. In fact, there are several fast algorithms being developed recently for such task in batch-mode, *i.e.* `miniasm` [44] or `wtdbg2` (Ruan, J. and Li, H. (2019) *Fast and accurate long-read assembly with wtdbg2.* bioRxiv. doi:10.1101/530972). However, to adapt those approaches for a streaming pipeline is non trivial. The modified OLC or DBG algorithm for nanopore data are both consuming much more resource due to the errors. Furthermore, the error correction step, *e.g.* `racon` [50] using POA graph data structure, is costly thus make it difficult to operate in real-time during the sequencing process.

6.4 Closing remarks

Long-reads data from TGS platforms, such as MinION, are considered as the current game-changer for genomics studies, especially genome assembly which has been partly addressed throughout the thesis.

Recently, however, there has been arguments concerning the quality of long-reads assembly from the DNA reference resource, such as NCBI database, that would affect the downstream analyses at protein level due to relatively high error rates especially indels [206, 207]. In the meantime, the hybrid approaches can combine the high fidelity of sequence data from Illumina to overcome the current defects of TGS assembly. Plus, the innovation of technology in combination with the improvement of computational methods can leverage the genome assembly for not only microorganisms, but also human and multiploidy species of greater complexity.

To sum up, the author strongly believe that the long reads real-time sequencing methods, *e.g.*by using ONT platforms, and their applications will continue to be the trending direction in life science. Advancements made by scientific and commercial bodies in this field would greatly facilitate the genome sequencing practice in order to ultimately resolve the long-lasting assembly problem.

References

[1] J. D. Watson, F. H. Crick, *et al. Molecular structure of nucleic acids.* Nature **171**(4356), 737 (1953).

[2] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, *et al. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd.* Science **269**(5223), 496 (1995).

[3] A. Goffeau, B. G. Barrell, H. Bussey, R. Davis, *et al. Life with 6000 genes.* Science **274**(5287), 546 (1996).

[4] The C. elegans Sequencing Consortium. *Genome sequence of the nematode C. elegans: A platform for investigating biology.* Science **282**, 2012 (1998).

[5] A. T. Chinwalla, L. L. Cook, K. D. Delehaunty, G. A. Fewell, L. A. Fulton, R. S. Fulton, T. A. Graves, L. W. Hillier, E. R. Mardis, J. D. McPherson, *et al. Initial sequencing and comparative analysis of the mouse genome.* Nature **420**(6915), 520 (2002).

[6] I. H. G. S. Consortium. *Finishing the euchromatic sequence of the human genome.* Nature **431**(7011), 931 (2004).

[7] H. A. Lewin, G. E. Robinson, W. J. Kress, W. J. Baker, J. Coddington, K. A. Crandall, R. Durbin, S. V. Edwards, F. Forest, M. T. P. Gilbert, *et al. Earth biogenome project: Sequencing life for the future of life.* Proceedings of the National Academy of Sciences **115**(17), 4325 (2018).

[8] F. Sanger, S. Nicklen, and A. R. Coulson. *DNA sequencing with chain-terminating inhibitors.* Proceedings of the National Academy of Sciences **74**(12), 5463 (1977).

[9] J. D. Freeman, R. L. Warren, J. R. Webb, B. H. Nelson, and R. A. Holt. *Profiling the T-cell receptor beta-chain repertoire by massively parallel sequencing.* Genome research **19**(10), 1817 (2009).

[10] M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlén, and P. Nyrén. *Real-time DNA sequencing using detection of pyrophosphate release.* Analytical biochemistry **242**(1), 84 (1996).

[11] M. Ronaghi, M. Uhlén, and P. Nyren. *A sequencing method based on real-time pyrophosphate.* Science **281**(5375), 363 (1998).

[12] C.-Y. Lee, Y.-C. Chiu, L.-B. Wang, Y.-L. Kuo, E. Y. Chuang, L.-C. Lai, and M.-H. Tsai. *Common applications of next-generation sequencing technologies in genomic research.* Translational Cancer Research **2**(1), 33 (2013).

[13] E. R. Mardis. *Dna sequencing technologies: 2006–2016.* Nature protocols **12**(2), 213 (2017).

[14] A. Philippidis. *TOP 10 Sequencing Companies: Plunging Cost, Wider Use of Technology Help Drive NGS Growth.* Genetic Engineering & Biotechnology News **38**(9), 6 (2018).

[15] D. J. Munroe and T. J. Harris. *Third-generation sequencing fireworks at marco island.* Nature biotechnology **28**(5), 426 (2010).

[16] C. Bleidorn. *Third generation sequencing: technology and its potential impact on evolutionary biodiversity research.* Systematics and biodiversity **14**(1), 1 (2016).

[17] A. Rhoads and K. F. Au. *Pacbio sequencing and its applications.* Genomics, proteomics & bioinformatics **13**(5), 278 (2015).

[18] J. Korlach, P. J. Marks, R. L. Cicero, J. J. Gray, D. L. Murphy, D. B. Roitman, T. T. Pham, G. A. Otto, M. Foquet, and S. W. Turner. *Selective aluminum passivation for*

targeted immobilization of single DNA polymerase molecules in zero-mode waveguide nanostructures. Proceedings of the National Academy of Sciences **105**(4), 1176 (2008).

[19] M. J. Levene, J. Korlach, S. W. Turner, M. Foquet, H. G. Craighead, and W. W. Webb. *Zero-mode waveguides for single-molecule analysis at high concentrations.* science **299**(5607), 682 (2003).

[20] J. Eid, A. Fehr, J. Gray, K. Luong, S. Turner, *et al. Real-time DNA sequencing from single polymerase molecules.* Science (New York, N.Y.) **323**(5910), 133 (2009).

[21] S. Koren, G. Harhay, T. Smith, J. Bono, D. Harhay, S. Mcvey, D. Radune, N. Bergman, and A. M. Phillippy. *Reducing Assembly Complexity of Microbial Genomes with Single-Molecule Sequencing.* Genome Biology **14**(9), R101 (2013).

[22] K. Berlin, S. Koren, C.-s. Chin, J. Drake, and M. Jane. *Assembling Large Genomes with Single-Molecule Sequencing and Locality Sensitive Hashing.* Nat Biotech **33**, 623 (2015).

[23] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis, and A. M. Phillippy. *Hybrid Error Correction and de novo Assembly of Single-molecule Sequencing Reads.* Nature Biotechnology **30**(7), 693 (2012).

[24] F. J. Ribeiro, D. Przybylski, S. Yin, T. Sharpe, S. Gnerre, A. Abouelleil, A. M. Berlin, A. Montmayeur, T. P. Shea, B. J. Walker, S. K. Young, C. Russ, C. Nusbaum, I. MacCallum, and D. B. Jaffe. *Finished bacterial genomes from shotgun sequence data.* Genome research **22**(11), 2270 (2012).

[25] J. J. Kasianowicz, E. Brandin, D. Branton, and D. W. Deamer. *Characterization of individual polynucleotide molecules using a membrane channel.* Proceedings of the National Academy of Sciences **93**(24), 13770 (1996).

[26] G. Church, D. W. Deamer, D. Branton, R. Baldarelli, and J. Kasianowicz. *Characterization of individual polymer molecules based on monomer-interface interactions* (1998). US Patent 5,795,782.

[27] J. Clarke, H.-C. Wu, L. Jayasinghe, A. Patel, S. Reid, and H. Bayley. *Continuous base identification for single-molecule nanopore dna sequencing.* Nature nanotechnology **4**(4), 265 (2009).

[28] L. E. Baum and T. Petrie. *Statistical inference for probabilistic functions of finite state Markov chains.* The annals of mathematical statistics **37**(6), 1554 (1966).

[29] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains.* The annals of mathematical statistics **41**(1), 164 (1970).

[30] H. Teng, M. D. Cao, M. B. Hall, T. Duarte, S. Wang, and L. J. Coin. *Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning.* GigaScience **7**(5), giy037 (2018).

[31] W. De Coster, P. De Rijk, A. De Roeck, T. De Pooter, S. D'Hert, M. Strazisar, K. Sleegers, and C. Van Broeckhoven. *Structural variants identified by Oxford Nanopore PromethION sequencing of the human genome.* Genome research pp. gr–244939 (2019).

[32] S. M. Nicholls, J. C. Quick, S. Tang, and N. J. Loman. *Ultra-deep, long-read nanopore sequencing of mock microbial community standards.* GigaScience **8**(5), giz043 (2019).

[33] M. D. Cao, D. Ganesamoorthy, M. A. Cooper, and L. J. M. Coin. *Realtime analysis and visualization of MinION sequencing data with npReader.* Bioinformatics **32**(5), 764 (2016).

[34] S. Goodwin, J. Gurtowski, S. Ethe-Sayers, P. Deshpande, M. C. Schatz, and W. R. McCombie. *Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome.* Genome Research **25**(11), 1750 (2015).

[35] M.-A. Madoui, S. Engelen, C. Cruaud, C. Belser, L. Bertrand, A. Alberti, A. Lemainque, P. Wincker, and J.-M. Aury. *Genome assembly using Nanopore-guided long and error-free DNA reads.* BMC Genomics **16**(1), 327 (2015).

[36] N. J. Loman, J. Quick, and J. T. Simpson. *A complete bacterial genome assembled de novo using only nanopore sequencing data.* Nature Methods **12**(8), 733 (2015).

[37] J. T. Simpson, R. E. Workman, P. Zuzarte, M. David, L. Dursi, and W. Timp. *Detecting dna cytosine methylation using nanopore sequencing.* Nature methods **14**(4), 407 (2017).

[38] M. Chaisson and G. Tesler. *Mapping Single Molecule Sequencing Reads Using Basic Local Alignment with Successive Refinement (BLASR): Application and Theory.* BMC Bioinformatics **13**(1), 238 (2012).

[39] R. S. Harris. *Improved pairwise alignment of genomic DNA.* Ph.D. thesis, The Pennsylvania State University (2007).

[40] S. M. Kie\lbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith. *Adaptive Seeds Tame Genomic Sequence Comparison.* Genome Research (2011).

[41] H. Li. *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM* p. 3 (2013). 1303.3997.

[42] G. Myers. *Efficient local alignment discovery amongst noisy long reads.* In D. Brown and B. Morgenstern, eds., *Algorithms in Bioinformatics*, pp. 52–67 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014).

[43] I. Sović, M. Šikić, A. Wilm, S. N. Fenlon, S. Chen, and N. Nagarajan. *Fast and sensitive mapping of nanopore sequencing reads with GraphMap.* Nature communications **7**, 11307 (2016).

[44] H. Li. *Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences.* Bioinformatics **32**(14), 2103 (2016).

[45] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy. *Mash: fast genome and metagenome distance estimation using MinHash.* Genome biology **17**(1), 132 (2016).

[46] K. Berlin, S. Koren, C.-S. Chin, J. P. Drake, J. M. Landolin, and A. M. Phillippy. *Assembling large genomes with single-molecule sequencing and locality-sensitive hashing.* Nature Biotechnology **33**(6), 623 (2015).

[47] C. Lee, C. Grasso, and M. F. Sharlow. *Multiple sequence alignment using partial order graphs.* Bioinformatics **18**(3), 452 (2002).

[48] C. Lee. *Generating consensus sequences from partial order multiple sequence alignment graphs.* Bioinformatics **19**(8), 999 (2003).

[49] C. Grasso and C. Lee. *Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems.* Bioinformatics **20**(10), 1546 (2004).

[50] R. Vaser, I. Sović, N. Nagarajan, and M. Šikić. *Fast and accurate de novo genome assembly from long uncorrected reads.* Genome research (2017).

[51] L. Garibyan and N. Avashia. *Research techniques made simple: polymerase chain reaction (pcr).* The Journal of investigative dermatology **133**(3), e6 (2013).

[52] I. J. Tsai, T. D. Otto, and M. Berriman. *Improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps.* Genome biology **11**(4), R41 (2010).

[53] M. Boetzer and W. Pirovano. *Toward almost closed genomes with gapfiller.* Genome biology **13**(6), R56 (2012).

[54] D. Paulino, R. L. Warren, B. P. Vandervalk, A. Raymond, S. D. Jackman, and I. Birol. *Sealer: a scalable gap-closing application for finishing draft genomes.* BMC bioinformatics **16**(1), 230 (2015).

[55] L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. *Comparison of next-generation sequencing systems.* BioMed Research International **2012** (2012).

[56] J. R. Miller, S. Koren, and G. Sutton. *Assembly Algorithms for Next-Generation Sequencing Data.* Genomics **95**(6), 315 (2010).

[57] Z. Li, Y. Chen, D. Mu, J. Yuan, Y. Shi, H. Zhang, J. Gan, N. Li, X. Hu, B. Liu, *et al. Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph.* Briefings in functional genomics **11**(1), 25 (2012).

[58] G. G. Sutton, O. White, M. D. Adams, and A. R. Kerlavage. *TIGR Assembler: A new tool for assembling large shotgun sequencing projects.* Genome Science and Technology **1**(1), 9 (1995).

[59] P. Green. *Documentation for PHRAP and CROSSMATCH (version 0.990319).* University of Washington, Seattle (1999).

[60] X. Huang and A. Madan. *CAP3: A DNA sequence assembly program.* Genome research **9**(9), 868 (1999).

[61] J. T. Simpson and M. Pop. *The theory and practice of genome sequence assembly.* Annual review of genomics and human genetics **16**, 153 (2015).

[62] R. Staden. *A strategy of dna sequencing employing computer programs.* Nucleic acids research **6**(7), 2601 (1979).

[63] R. M. Idury and M. S. Waterman. *A new algorithm for dna sequence assembly.* Journal of computational biology **2**(2), 291 (1995).

[64] P. A. Pevzner, H. Tang, and M. S. Waterman. *An Eulerian Path Approach to DNA Fragment Assembly.* Proceedings of the National Academy of Sciences **98**(17), 9748 (2001).

[65] P. Ferragina and G. Manzini. *Indexing Compressed Text.* Journal of the ACM **52**(4), 552 (2005).

[66] Z. Ning, A. J. Cox, and J. C. Mullikin. *SSAHA: a fast search method for large DNA databases.* Genome research **11**(10), 1725 (2001).

[67] D. R. Zerbino and E. Birney. *Velvet: algorithms for de novo short read assembly using de Bruijn graphs.* Genome research **18**(5), 821 (2008).

[68] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, A. V. Pyshkin, A. V. Sirotkin, N. Vyahhi, G. Tesler, M. A. Alekseyev, and P. A. Pevzner. *SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing.* Journal of Computational Biology **19**(5), 455 (2012).

[69] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. *ABySS: A parallel assembler for short read sequence data.* Genome Research **19**(6), 1117 (2009).

[70] T. Magoc, S. Pabinger, S. Canzar, X. Liu, Q. Su, D. Puiu, L. J. Tallon, and S. L. Salzberg. *GAGE-B: An evaluation of genome assemblers for bacterial organisms.* Bioinformatics **29**(14), 1718 (2013).

[71] S. Koren and A. M. Phillippy. *One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly.* Current Opinion in Microbiology **23**, 110 (2015).

[72] A. Bashir, A. A. Klammer, W. P. Robins, C.-S. Chin, D. Webster, E. Paxinos, D. Hsu, M. Ashby, S. Wang, P. Peluso, R. Sebra, J. Sorenson, J. Bullard, J. Yen, M. Valdovino, E. Mollova, K. Luong, S. Lin, B. LaMay, A. Joshi, L. Rowe, M. Frace, C. L. Tarr, M. Turnsek, B. M. Davis, A. Kasarskis, J. J. Mekalanos, M. K. Waldor, and E. E. Schadt. *A Hybrid Approach for the Automated Finishing of Bacterial Genomes.* Nature Biotechnology **30**(7), 701 (2012).

[73] E. Karlsson, A. Lärkeryd, A. Sjödin, M. Forsman, and P. Stenberg. *Scaffolding of a bacterial genome using MinION nanopore sequencing.* Scientific Reports **5**, 11996 (2015).

[74] C. M. Hudson, Z. W. Bent, R. J. Meagher, and K. P. Williams. *Resistance determinants and mobile genetic elements of an NDM-1-encoding Klebsiella pneumoniae strain.* PloS one **9**(6), e99209 (2014).

[75] P. M. Ashton, S. Nair, T. Dallman, S. Rubino, W. Rabsch, S. Mwaigwisya, J. Wain, and J. O'Grady. *MinION nanopore sequencing identifies the position and structure of a bacterial antibiotic resistance island*. Nature Biotechnology **33**(3), 296 (2015).

[76] V. Jayakumar and Y. Sakakibara. *Comprehensive evaluation of non-hybrid genome assembly tools for third-generation pacbio long-read sequence data*. Briefings in bioinformatics (2017).

[77] C.-S. Chin, D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler, S. W. Turner, and J. Korlach. *Nonhybrid, finished microbial genome assemblies from Long-Read SMRT sequencing data*. Nature Methods **10**(6), 563 (2013).

[78] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter. *A Whole-Genome Assembly of Drosophila*. Science **287**(5461), 2196 (2000).

[79] S. Koren, B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy. *Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation*. Genome research pp. gr–215087 (2017).

[80] C.-S. Chin, P. Peluso, F. J. Sedlazeck, M. Nattestad, G. T. Concepcion, A. Clum, C. Dunn, R. O'Malley, R. Figueroa-Balderas, A. Morales-Cruz, *et al. Phased diploid genome assembly with single-molecule real-time sequencing*. Nature methods **13**(12), 1050 (2016).

[81] Y. Lin, J. Yuan, M. Kolmogorov, M. W. Shen, M. Chaisson, and P. A. Pevzner. *Assembly of long error-prone reads using de bruijn graphs*. Proceedings of the National Academy of Sciences **113**(52), E8396 (2016).

[82] M. Kolmogorov, J. Yuan, Y. Lin, and P. A. Pevzner. *Assembly of long, error-prone reads using repeat graphs.* Nature biotechnology **37**(5), 540 (2019).

[83] V. Deshpande, E. D. K. Fung, S. Pham, and V. Bafna. *Cerulean: A Hybrid Assembly Using High Throughput Short and Long Reads.* In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8126 LNBI, pp. 349–363 (2013). 1307.7933.

[84] M. Boetzer and W. Pirovano. *SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information.* BMC bioinformatics **15**(1), 211 (2014).

[85] E. Karlsson, A. Lärkeryd, A. Sjödin, M. Forsman, and P. Stenberg. *Scaffolding of a bacterial genome using MinION nanopore sequencing.* Scientific reports **5** (2015).

[86] R. L. Warren, C. Yang, B. P. Vandervalk, B. Behsaz, A. Lagman, S. J. M. Jones, and I. Birol. *LINKS: Scalable, alignment-free scaffolding of draft genomes with long reads.* GigaScience **4**(1), 35 (2015).

[87] R. R. Wick, L. M. Judd, C. L. Gorrie, and K. E. Holt. *Unicycler: resolving bacterial genome assemblies from short and long sequencing reads.* PLOS Computational Biology **13**(6), e1005595 (2017).

[88] D. Phillips. *Programming Real-Time Computer Systems* (1966).

[89] M. Ben-Ari and M. Ben-Arî. *Principles of concurrent and distributed programming* (Pearson Education, 2006).

[90] D. Croushore. *Frontiers of real-time data analysis.* Journal of economic literature **49**(1), 72 (2011).

[91] J. Quick, P. Ashton, S. Calus, C. Chatt, S. Gossain, J. Hawker, S. Nair, K. Neal, K. Nye, T. Peters, E. De Pinna, E. Robinson, K. Struthers, M. Webber, A. Catto, T. J. Dallman, P. Hawkey, and N. J. Loman. *Rapid draft sequencing and real-time nanopore sequencing in a hospital outbreak of Salmonella.* Genome Biology **16**(1), 114 (2015).

[92] A. L. Greninger, S. N. Naccache, S. Federman, G. Yu, P. Mbala, V. Bres, D. Stryke, J. Bouquet, S. Somasekar, J. M. Linnen, R. Dodd, P. Mulembakani, B. S. Schneider, J.-J. Muyembe-Tamfum, S. L. Stramer, and C. Y. Chiu. *Rapid metagenomic identification of viral pathogens in clinical samples by real-time nanopore sequencing analysis.* Genome Medicine **7**(1), 99 (2015).

[93] M. Loose, S. Malla, and M. Stout. *Real time selective sequencing using nanopore technology.* bioRxiv (2016).

[94] H. S. Edwards, R. Krishnakumar, A. Sinha, S. W. Bird, K. D. Patel, and M. S. Bartsch. *Real-time selective sequencing with rubric: Read until with basecall and reference-informed criteria.* bioRxiv p. 460014 (2018).

[95] M. D. Cao, D. Ganesamoorthy, A. G. Elliott, H. Zhang, M. A. Cooper, and L. J. Coin. *Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinIONTM sequencing.* GigaScience **5**(1), 32 (2016).

[96] S. Bialasiewicz, T. P. Duarte, S. H. Nguyen, V. Sukumaran, A. Stewart, S. Appleton, M. E. Pitt, A. Bainomugisa, A. V. Jennison, R. Graham, *et al. Rapid Diagnosis of Capnocytophaga canimorsus Septic Shock in an Immunocompetent Individual Using Real-Time Nanopore Sequencing* (2018).

[97] M. D. Cao, S. H. Nguyen, D. Ganesamoorthy, A. G. Elliott, M. A. Cooper, and L. J. Coin. *Scaffolding and completing genome assemblies in real-time with nanopore sequencing.* Nature Communications **8**, 14515 (2017).

[98] S. H. Nguyen, T. P. Duarte, L. J. Coin, and M. D. Cao. *Real-time demultiplexing Nanopore barcoded sequencing data with npBarcode.* Bioinformatics **33**(24), 3988 (2017).

[99] J. J. Kasianowicz, E. Brandin, D. Branton, and D. W. Deamer. *Characterization of individual polynucleotide molecules using a membrane channel.* Proceedings of the National Academy of Sciences **93**(24), 13770 (1996).

[100] D. Branton, D. W. Deamer, A. Marziali, H. Bayley, S. A. Benner, T. Butler, M. Di Ventra, S. Garaj, A. Hibbs, X. Huang, S. B. Jovanovich, P. S. Krstic, S. Lindsay, X. S. Ling, C. H. Mastrangelo, A. Meller, J. S. Oliver, Y. V. Pershin, J. M. Ramsey, R. Riehn, G. V. Soni, V. Tabard-Cossa, M. Wanunu, M. Wiggin, and J. A. Schloss. *The potential and challenges of nanopore sequencing.* Nature biotechnology **26**(10), 1146 (2008).

[101] D. Stoddart, A. J. Heron, E. Mikhailova, G. Maglia, and H. Bayley. *Single-nucleotide discrimination in immobilized DNA oligonucleotides with a biological nanopore.* Proceedings of the National Academy of Sciences of the United States of America **106**(19), 7702 (2009).

[102] S. L. Castro-Wallace, C. Y. Chiu, K. K. John, S. E. Stahl, K. H. Rubins, A. B. R. McIntyre, J. P. Dworkin, M. L. Lupisella, D. J. Smith, D. J. Botkin, T. A. Stephenson, S. Juul, D. J. Turner, F. Izquierdo, S. Federman, D. Stryke, S. Somasekar, N. Alexander, G. Yu, C. Mason, and A. S. Burton. *Nanopore DNA Sequencing and Genome Assembly on the International Space Station.* Tech. rep. (2016).

[103] B. Istace, A. Friedrich, L. D'Agata, S. Faye, E. Payen, O. Beluche, C. Caradec, S. Davidas, C. Cruaud, G. Liti, A. Lemainque, S. Engelen, P. Wincker, J. Schacherer, and J.-M. Aury. *de novo assembly and population genomic survey of natural yeast isolates with the Oxford Nanopore MinION sequencer.* Tech. rep. (2016).

[104] T. J. Treangen and S. L. Salzberg. *Repetitive DNA and Next-generation Sequencing: Computational Challenges and Solutions.* Nature Reviews Genetics **13**(1), 36 (2012).

[105] N. J. Loman and A. R. Quinlan. *Poretools: a toolkit for analyzing nanopore sequence data.* Bioinformatics **30**(23), 3399 (2014).

[106] M. Jain, I. T. Fiddes, K. H. Miga, H. E. Olsen, B. Paten, and M. Akeson. *Improved data analysis for the MinION nanopore sequencer.* Nature Methods **12**(4), 351 (2015).

[107] A. Carattoli, E. Zankari, A. Garcia-Fernandez, M. Voldby Larsen, O. Lund, L. Villa, F. Moller Aarestrup, and H. Hasman. *In Silico Detection and Typing of Plasmids*

using PlasmidFinder and Plasmid Multilocus Sequence Typing. Antimicrobial Agents and Chemotherapy **58**(7), 3895 (2014).

[108] T. Seemann. *Prokka: rapid prokaryotic genome annotation.* Bioinformatics **30**(14), 2068 (2014).

[109] M. G. I. Langille, W. W. L. Hsiao, and F. S. L. Brinkman. *Detecting genomic islands using bioinformatics approaches.* Nature reviews. Microbiology **8**(5), 373 (2010).

[110] Y. Mantri and K. P. Williams. *Islander: a database of integrative islands in prokaryotic genomes, the associated integrases and their DNA site specificities.* Nucleic acids research **32**(Database issue), D55 (2004).

[111] Y. Zhou, Y. Liang, K. H. Lynch, J. J. Dennis, and D. S. Wishart. *PHAST: A Fast Phage Search Tool.* Nucleic Acids Research **39**(suppl), W347 (2011).

[112] J. Quick, A. R. Quinlan, and N. J. Loman. *A Reference Bacterial Genome Dataset Generated on the MinION Portable Single-molecule Nanopore Sequencer.* GigaScience **3**(1), 22 (2014).

[113] B. J. Walker, T. Abeel, T. Shea, M. Priest, A. Abouelliel, S. Sakthikumar, C. A. Cuomo, Q. Zeng, J. Wortman, S. K. Young, and A. M. Earl. *Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement.* PLoS ONE **9**(11), e112963 (2014).

[114] A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler. *QUAST: quality assessment tool for genome assemblies.* Bioinformatics **29**(8), 1072 (2013).

[115] M. David, L. J. Dursi, D. Yao, P. C. Boutros, and J. T. Simpson. *Nanocall: An Open Source Basecaller for Oxford Nanopore Sequencing Data.* bioRxiv p. 046086 (2016).

[116] V. Boža, B. Brejová, and T. Vina. *DeepNano: Deep Recurrent Neural Networks for Base Calling in MinION Nanopore Reads* (2016). `1603.09195`.

[117] J. B. Kruskal. *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem.* Proceedings of the American Mathematical Society **7**(1), 48 (1956).

[118] A. M. Bolger, M. Lohse, and B. Usadel. *Trimmomatic: a flexible trimmer for Illumina sequence data.* Bioinformatics **30**(15), 2114 (2014).

[119] E. Zankari, H. Hasman, S. Cosentino, M. Vestergaard, S. Rasmussen, O. Lund, F. M. Aarestrup, and M. V. Larsen. *Identification of Acquired Antimicrobial Resistance Genes.* Journal of Antimicrobial Chemotherapy **67**(11), 2640 (2012).

[120] E. van der Helm, L. Imamovic, M. M. Hashim Ellabaan, W. van Schaik, A. Koza, and M. O. Sommer. *Rapid resistome mapping using nanopore sequencing.* Nucleic acids research **45**(8), e61 (2017).

[121] O. Gotoh. *An improved algorithm for matching biological sequences.* Journal of molecular biology **162**(3), 705 (1982).

[122] M. E. Pitt, A. G. Elliott, M. D. Cao, D. Ganesamoorthy, I. Karaiskos, H. Giamarellou, C. S. Abboud, M. A. Blaskovich, M. A. Cooper, and L. J. Coin. *Multifactorial chromosomal variants regulate polymyxin resistance in extensively drug-resistant klebsiella pneumoniae.* Microbial genomics **4**(3) (2018).

[123] M. D. Cao, D. Ganesamoorthy, A. Elliott, H. Zhang, M. Cooper, and L. Coin. *Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinIONTM sequencing.* bioRxiv (2015).

[124] R. M. Martin and M. A. Bachman. *Colonization, infection, and the accessory genome of Klebsiella pneumoniae.* Frontiers in cellular and infection microbiology **8**, 4 (2018).

[125] S. S. Magill, J. R. Edwards, W. Bamberg, Z. G. Beldavs, G. Dumyati, M. A. Kainer, R. Lynfield, M. Maloney, L. McAllister-Hollod, J. Nadle, *et al. Multistate point-prevalence survey of health care–associated infections.* New England Journal of Medicine **370**(13), 1198 (2014).

[126] A. A. Kalanuria, W. Zai, and M. Mirski. *Ventilator-associated pneumonia in the ICU.* Critical care **18**(2), 208 (2014).

[127] K. Talha, Z. Hasan, F. Selina, and M. Palash. *Organisms associated with ventilator associated pneumonia in intensive care unit.* Mymensingh medical journal: MMJ **18**(1 Suppl), S93 (2009).

[128] R. Podschun and U. Ullmann. *Klebsiella spp. as nosocomial pathogens: epidemiology, taxonomy, typing methods, and pathogenicity factors.* Clinical microbiology reviews **11**(4), 589 (1998).

[129] I. Karaiskos and H. Giamarellou. *Multidrug-resistant and extensively drug-resistant Gram-negative pathogens: current and emerging therapeutic approaches.* Expert opinion on pharmacotherapy **15**(10), 1351 (2014).

[130] C. M. Hudson, Z. W. Bent, R. J. Meagher, and K. P. Williams. *Resistance Determinants and Mobile Genetic Elements of an NDM-1 Encoding Klebsiella pneumoniae Strain.* PLoS ONE **9**(6), e99209 (2014).

[131] S. Navon-Venezia, K. Kondratyeva, and A. Carattoli. *Klebsiella pneumoniae: a major worldwide source and shuttle for antibiotic resistance.* FEMS microbiology reviews **41**(3), 252 (2017).

[132] L. Chen, R. Todd, J. Kiehlbauch, M. Walters, and A. Kallen. *Notes from the Field: Pan-Resistant New Delhi Metallo-Beta-Lactamase-Producing Klebsiella pneumoniae-Washoe County, Nevada, 2016.* MMWR. Morbidity and mortality weekly report **66**(1), 33 (2017).

[133] H. M. Zowawi, B. M. Forde, M. Alfaresi, A. Alzarouni, Y. Farahat, T.-M. Chong, W.-F. Yin, K.-G. Chan, J. Li, M. A. Schembri, S. A. Beatson, and D. L. Paterson. *Stepwise evolution of pandrug-resistance in Klebsiella pneumoniae.* Scientific Reports **5**, 15082 (2015).

[134] M. O. Sommer, C. Munck, R. V. Toft-Kehler, and D. I. Andersson. *Prediction of antibiotic resistance: time for a new preclinical paradigm?* Nature Reviews Microbiology **15**(11), 689 (2017).

[135] J. L. Gardy and N. J. Loman. *Towards a genomics-informed, real-time, global pathogen surveillance system.* Nature Reviews Genetics **19**(1), 9 (2018).

[136] J. K. Lemon, P. P. Khil, K. M. Frank, and J. P. Dekker. *Rapid nanopore sequencing of plasmids and resistance gene detection in clinical isolates.* Journal of clinical microbiology pp. JCM–01069 (2017).

[137] A. A. Votintseva, P. Bradley, L. Pankhurst, C. del Ojo Elias, M. Loose, K. Nilgiriwala, A. Chatterjee, E. G. Smith, N. Sanderson, T. M. Walker, *et al. Same-day diagnostic and surveillance data for tuberculosis via whole genome sequencing of direct respiratory samples.* Journal of clinical microbiology pp. JCM–02483 (2017).

[138] R. R. Wick, L. M. Judd, C. L. Gorrie, and K. E. Holt. *Completing bacterial genome assemblies with multiplex MinION sequencing.* Microbial genomics **3**(10) (2017).

[139] R. Li, M. Xie, N. Dong, D. Lin, X. Yang, M. H. Y. Wong, E. W.-C. Chan, and S. Chen. *Efficient generation of complete sequences of MDR-encoding plasmids by rapid assembly of MinION barcoding sequencing data.* GigaScience **7**(3), gix132 (2018).

[140] S. George, L. Pankhurst, A. Hubbard, A. Votintseva, N. Stoesser, A. E. Sheppard, A. Mathers, R. Norris, I. Navickaite, C. Eaton, *et al. Resolving plasmid structures in Enterobacteriaceae using the MinION nanopore sequencer: assessment of MinION and MinION/Illumina hybrid data assembly approaches.* Microbial genomics **3**(8) (2017).

[141] A. E. Darling, A. Tritt, J. A. Eisen, and M. T. Facciotti. *Mauve assembly metrics.* Bioinformatics **27**(19), 2756 (2011).

[142] J. T. Robinson, H. Thorvaldsdóttir, W. Winckler, M. Guttman, E. S. Lander, G. Getz, and J. P. Mesirov. *Integrative genomics viewer.* Nature biotechnology **29**(1), 24 (2011).

[143] H. Thorvaldsdóttir, J. T. Robinson, and J. P. Mesirov. *Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration.* Briefings in bioinformatics **14**(2), 178 (2013).

[144] A. Carattoli, E. Zankari, A. Garcìa-Fernandez, M. V. Larsen, O. Lund, L. Villa, F. M. Aarestrup, and H. Hasman. *Plasmidfinder and pmlst: in silico detection and typing of plasmids.* Antimicrobial agents and chemotherapy pp. AAC–02412 (2014).

[145] A. Carattoli. *Resistance plasmid families in Enterobacteriaceae.* Antimicrobial agents and chemotherapy **53**(6), 2227 (2009).

[146] S. Desmet, S. Nepal, J. M. van Dijl, M. Van Ranst, M. A. Chlebowicz, J. W. Rossen, J. K. Van Houdt, P. Maes, K. Lagrou, and E. Bathoorn. *Antibiotic Resistance Plasmids Cointegrated into a Megaplasmid Harboring the blaOXA-427 Carbapenemase Gene.* Antimicrobial agents and chemotherapy **62**(3), e01448 (2018).

[147] C. C. Papagiannitsis, M. Dolejska, R. Izdebski, P. Giakkoupi, A. Skálová, K. Chudějová, H. Dobiasova, A. C. Vatopoulos, L. P. Derde, M. J. Bonten, *et al. Characterisation of IncA/C2 plasmids carrying an In416-like integron with the blaVIM-19 gene from Klebsiella pneumoniae ST383 of Greek origin.* International journal of antimicrobial agents **47**(2), 158 (2016).

[148] L. Chen, K. D. Chavda, R. G. Melano, M. R. Jacobs, B. Koll, T. Hong, A. D. Rojtman, M. H. Levi, R. A. Bonomo, and B. N. Kreiswirth. *Comparative genomic analysis of KPC-encoding pKpQIL-like plasmids and their distribution in New Jersey and New York hospitals.* Antimicrobial agents and chemotherapy **58**(5), 2871 (2014).

[149] L. Poirel, R. A. Bonnin, and P. Nordmann. *Genetic features of the widespread plasmid coding for the carbapenemase OXA-48.* Antimicrobial agents and chemotherapy **56**(1), 559 (2012).

[150] A. Potron, L. Poirel, and P. Nordmann. *Derepressed transfer properties leading to the efficient spread of the plasmid encoding carbapenemase OXA-48.* Antimicrobial agents and chemotherapy **58**(1), 467 (2014).

[151] T. Lassmann, O. Frings, and E. L. L. Sonnhammer. *Kalign2: High-performance Multiple Alignment of Protein and Nucleotide Sequences Allowing External Features.* Nucleic Acids Research **37**(3), 858 (2009).

[152] R. Rozov, G. Goldshlager, E. Halperin, and R. Shamir. *Faucet: streaming de novo assembly graph construction.* Bioinformatics **34**(1), 147 (2017).

[153] F. Giordano, L. Aigrain, M. A. Quail, P. Coupland, J. K. Bonfield, R. M. Davies, G. Tischler, D. K. Jackson, T. M. Keane, J. Li, *et al. De novo yeast genome assemblies from MinION, PacBio and MiSeq platforms.* Scientific reports **7**(1), 3935 (2017).

[154] D. Antipov, A. Korobeynikov, J. S. McLean, and P. A. Pevzner. *hybridSPAdes: an algorithm for hybrid assembly of short and long reads.* Bioinformatics **32**(7), 1009 (2016).

[155] A. D. Prjibelski, I. Vasilinetc, A. Bankevich, A. Gurevich, T. Krivosheeva, S. Nurk, S. Pham, A. Korobeynikov, A. Lapidus, and P. A. Pevzner. *ExSPAnder: a universal repeat resolver for DNA fragment assembly.* Bioinformatics **30**(12), i293 (2014).

[156] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise.* pp. 226–231 (AAAI Press, 1996).

[157] S. Kullback and R. A. Leibler. *On information and sufficiency.* The annals of mathematical statistics **22**(1), 79 (1951).

[158] E. W. Dijkstra. *A note on two problems in connexion with graphs.* Numerische mathematik **1**(1), 269 (1959).

[159] W. Huang, L. Li, J. R. Myers, and G. T. Marth. *ART: a Next-generation Sequencing Read Simulator.* Bioinformatics **28**(4), 593 (2012).

[160] Y. Ono, K. Asai, and M. Hamada. *PBSIM: PacBio Reads Simulator Toward Accurate Genome Assembly.* Bioinformatics (2012).

[161] A. Mikheenko, A. Prjibelski, V. Saveliev, D. Antipov, and A. Gurevich. *Versatile genome assembly evaluation with QUAST-LG.* Bioinformatics **34**(13), i142 (2018).

[162] M. Brudno, S. Malde, A. Poliakov, C. B. Do, O. Couronne, I. Dubchak, and S. Batzoglou. *Glocal alignment: finding rearrangements during alignment.* Bioinformatics **19**(suppl_1), i54 (2003).

[163] R. F. Potter, A. W. Dsouza, and G. Dantas. *The rapid spread of carbapenem-resistant Enterobacteriaceae.* Drug Resistance Updates **29**, 30 (2016).

[164] J. Lu, Y. Wang, J. Li, L. Mao, S. H. Nguyen, T. Duarte, L. Coin, P. Bond, Z. Yuan, and J. Guo. *Triclosan at environmentally relevant concentrations promotes horizontal transfer of multidrug resistance genes within and across bacterial genera.* Environment international **121**, 1217 (2018).

[165] J. Gardy, N. J. Loman, and A. Rambaut. *Real-time digital pathogen surveillance the time is now.* Genome Biology **16**(1), 155 (2015).

[166] K. G. Andersen, B. J. Shapiro, C. B. Matranga, R. Sealfon, A. E. Lin, L. M. Moses, O. A. Folarin, A. Goba, I. Odia, P. E. Ehiane, *et al. Clinical sequencing uncovers origins and evolution of Lassa virus.* Cell **162**(4), 738 (2015).

[167] E. C. Holmes, G. Dudas, A. Rambaut, and K. G. Andersen. *The evolution of ebola virus: Insights from the 2013–2016 epidemic.* Nature **538**(7624), 193 (2016).

[168] G. Dudas, L. M. Carvalho, T. Bedford, A. J. Tatem, G. Baele, N. R. Faria, D. J. Park, J. T. Ladner, A. Arias, D. Asogun, *et al. Virus genomes reveal factors that spread and sustained the ebola epidemic.* Nature **544**(7650), 309 (2017).

[169] J. Wang, N. E. Moore, Y.-M. Deng, D. A. Eccles, and R. J. Hall. *MinION nanopore sequencing of an influenza genome.* Frontiers in microbiology **6**, 766 (2015).

[170] J. Quick, N. J. Loman, S. Duraffour, J. T. Simpson, *et al. Real-time, portable genome sequencing for Ebola surveillance.* Nature **530**(7589), 228 (2016).

[171] J. Quick, N. D. Grubaugh, S. T. Pullan, I. M. Claro, A. D. Smith, K. Gangavarapu, G. Oliveira, R. Robles-Sikisaka, T. F. Rogers, N. A. Beutler, *et al. Multiplex PCR method for MinION and Illumina sequencing of Zika and other virus genomes directly from clinical samples.* nature protocols **12**(6), 1261 (2017).

[172] A. Rector, R. Tachezy, and M. Van Ranst. *A sequence-independent strategy for detection and cloning of circular DNA virus genomes by using multiply primed rolling-circle amplification.* Journal of virology **78**(10), 4993 (2004).

[173] A. K. Inoue-Nagata, L. C. Albuquerque, W. B. Rocha, and T. Nagata. *A simple method for cloning the complete begomovirus genome using the bacteriophage φ29 DNA polymerase.* Journal of virological methods **116**(2), 209 (2004).

[174] J. Schubert, A. Habekuß, K. Kazmaier, and H. Jeske. *Surveying cereal-infecting geminiviruses in Germanydiagnostics and direct sequencing using rolling circle amplification.* Virus research **127**(1), 61 (2007).

[175] D. Knierim and E. Maiss. *Application of Phi29 DNA polymerase in identification and full-length clone inoculation of tomato yellow leaf curl Thailand virus and tobacco leaf curl Thailand virus.* Archives of virology **152**(5), 941 (2007).

[176] D. N. Shepherd, D. P. Martin, P. Lefeuvre, A. L. Monjane, B. E. Owor, E. P. Rybicki, and A. Varsani. *A protocol for the rapid isolation of full geminivirus genomes from dried plant tissue.* Journal of virological methods **149**(1), 97 (2008).

[177] D. Haible, S. Kober, and H. Jeske. *Rolling circle amplification revolutionizes diagnosis and genomics of geminiviruses.* Journal of virological methods **135**(1), 9 (2006).

[178] M. Homs, S. Kober, G. Kepp, and H. Jeske. *Mitochondrial plasmids of sugar beet amplified via rolling circle method detected during curtovirus screening.* Virus research **136**(1-2), 124 (2008).

[179] R. Johne, H. Müller, A. Rector, M. Van Ranst, and H. Stevens. *Rolling-circle amplification of viral DNA genomes using phi29 polymerase.* Trends in microbiology **17**(5), 205 (2009).

[180] A. D. Geering, T. Scharaschkin, and P.-Y. Teycheney. *The classification and nomenclature of endogenous viruses of the family Caulimoviridae.* Archives of virology **155**(1), 123 (2010).

[181] D. Mollov, B. Lockhart, D. C. Zlesak, and N. Olszewski. *Complete nucleotide sequence of rose yellow vein virus, a member of the family Caulimoviridae having a novel genome organization.* Archives of virology **158**(4), 877 (2013).

[182] A. Bhat, T. Hohn, and R. Selvarajan. *Badnaviruses: the current global scenario.* Viruses **8**(6), 177 (2016).

[183] A. C. Sukal, D. B. Kidanemariam, J. L. Dale, R. M. Harding, and A. P. James. *Characterization of a novel member of the family Caulimoviridae infecting Dioscorea nummularia in the Pacific, which may represent a new genus of dsDNA plant viruses.* PloS one **13**(9), e0203038 (2018).

[184] A. L. Rockwood, D. K. Crockett, J. R. Oliphant, and K. S. Elenitoba-Johnson. *Sequence alignment by cross-correlation.* Journal of biomolecular techniques: JBT **16**(4), 453 (2005).

[185] G. Ravi, S. Divya, M. Ankush, and S. Kuldip. *A novel signal processing measure to identify exact and inexact tandem repeat patterns in dna sequences.* EURASIP Journal on Bioinformatics and Systems Biology **2007**(1), 43596 (2007).

[186] W. M. Gentleman and G. Sande. *Fast Fourier Transforms: for fun and profit.* In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, pp. 563–578 (ACM, 1966).

[187] C. Van Loan. *Computational frameworks for the fast Fourier transform*, vol. 10 (Siam, 1992).

[188] M. Heideman, D. Johnson, and C. Burrus. *Gauss and the history of the fast Fourier transform.* IEEE ASSP Magazine **1**(4), 14 (1984).

[189] P. McLeod and G. Wyvill. *A smarter way to find pitch.* In *ICMC* (2005).

[190] H. Eichelberger and L. Ii. *Jtransform – a Java source code transformation framework.* Tech. rep., TR 303, Institute for Computer Science, Wurzburg University (2002).

[191] S. W. Smith *et al. The scientist and engineer's guide to digital signal processing* (1997).

[192] I. Bankman. *Handbook of medical image processing and analysis* (Elsevier, 2008).

[193] F. J. Harris. *On the use of windows for harmonic analysis with the discrete Fourier transform.* Proceedings of the IEEE **66**(1), 51 (1978).

[194] R. B. Blackman and J. W. Tukey. *The measurement of power spectra from the point of view of communications engineering.* Bell System Technical Journal **37**(1), 185 (1958).

[195] F. J. Rang, W. P. Kloosterman, and J. de Ridder. *From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy.* Genome biology **19**(1), 90 (2018).

[196] E. A. Manrao, I. M. Derrington, A. H. Laszlo, K. W. Langford, M. K. Hopper, N. Gillgren, M. Pavlenok, M. Niederweis, and J. H. Gundlach. *Reading DNA at single-nucleotide resolution with a mutant MspA nanopore and phi29 DNA polymerase.* Nature biotechnology **30**(4), 349 (2012).

[197] G. M. Cherf, K. R. Lieberman, H. Rashid, C. E. Lam, K. Karplus, and M. Akeson. *Automated forward and reverse ratcheting of DNA in a nanopore at 5-å precision.* Nature biotechnology **30**(4), 344 (2012).

[198] P. Sarkozy, Á. Jobbágy, and P. Antal. *Calling homopolymer stretches from raw nanopore reads by analyzing k-mer dwell times.* In *EMBEC & NBC 2017*, pp. 241–244 (Springer, 2017).

[199] M. Jain, S. Koren, K. H. Miga, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, *et al. Nanopore sequencing and assembly of a human genome with ultra-long reads.* Nature biotechnology **36**(4), 338 (2018).

[200] M. C. Stancu, M. J. Van Roosmalen, I. Renkens, M. M. Nieboer, S. Middelkamp, J. De Ligt, G. Pregno, D. Giachino, G. Mandrile, J. E. Valle-Inclan, *et al. Mapping and phasing of structural variation in patient genomes using nanopore sequencing.* Nature communications **8**(1), 1326 (2017).

[201] C. L. Ip, M. Loose, J. R. Tyson, M. de Cesare, B. L. Brown, M. Jain, R. M. Leggett, D. A. Eccles, V. Zalunin, J. M. Urban, *et al. MinION Analysis and Reference Consortium: Phase 1 data release and analysis.* F1000Research **4** (2015).

[202] E. E. Schadt, S. Turner, and A. Kasarskis. *A window into third-generation sequencing.* Human molecular genetics **19**(R2), R227 (2010).

[203] R. Thakur, R. Bandopadhyay, B. Chaudhary, and S. Chatterjee. *Now and Next-Generation Sequencing Techniques: Future of Sequence Analysis Using Cloud Computing.* Frontiers in Genetics **3**, 280 (2012).

[204] S. Nurk, D. Meleshko, A. Korobeynikov, and P. A. Pevzner. *metaSPAdes: a new versatile metagenomic assembler.* Genome Research **27**(5), 824 (2017).

[205] D. D. Kang, J. Froula, R. Egan, and Z. Wang. *MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities.* PeerJ **3**, e1165 (2015).

[206] M. Watson and A. Warr. *Errors in long-read assemblies can critically affect protein prediction.* Nature Biotechnology p. 1 (2019).

[207] S. Koren, A. M. Phillippy, J. T. Simpson, N. J. Loman, and M. Loose. *Reply to errors in long-read assemblies can critically affect protein prediction.* Nature Biotechnology p. 1 (2019).

A

Supplementary materials for Chapter 2

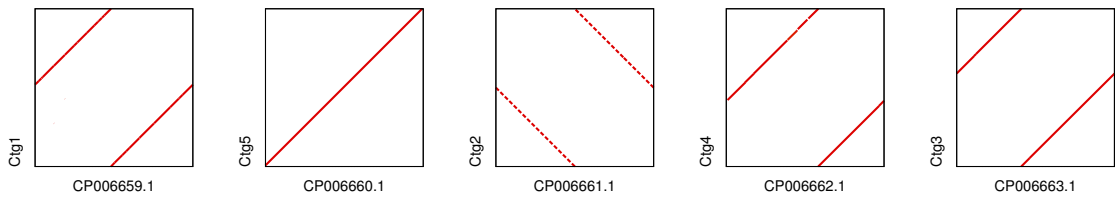


FIGURE A.1: Alignment of the `npScarf`'s assembly for *K. pneumoniae* ATCC BAA-2146 to its draft reference genomes (GeneBank Accession GCA_000364385.2). The five contigs were in complete agreement with the chromosome and four plasmids.

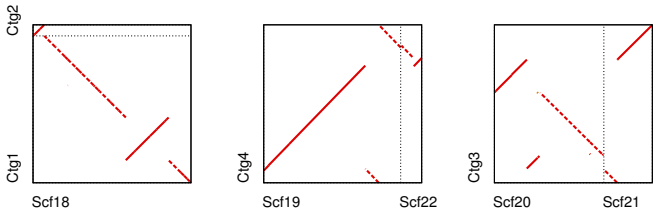


FIGURE A.2: Alignment of the `npScarf`'s assembly for *K. pneumoniae* ATCC 13883 to its draft reference genomes (GeneBank Accession GCA_000742135.1). For display convenience, IDs of reference sequences are abbreviated by the last two digits before the dot in the original accession code: Contigs 1 and 2 were aligned to the reference Scaffold 18 (KN046818.1). Contig 4 was aligned to two Scaffolds 19 (KN046819.1) and 22 (KN046822.1) and Contig 3 to Scaffolds 20 (KN046820.1) and 21 (KN046821.1).

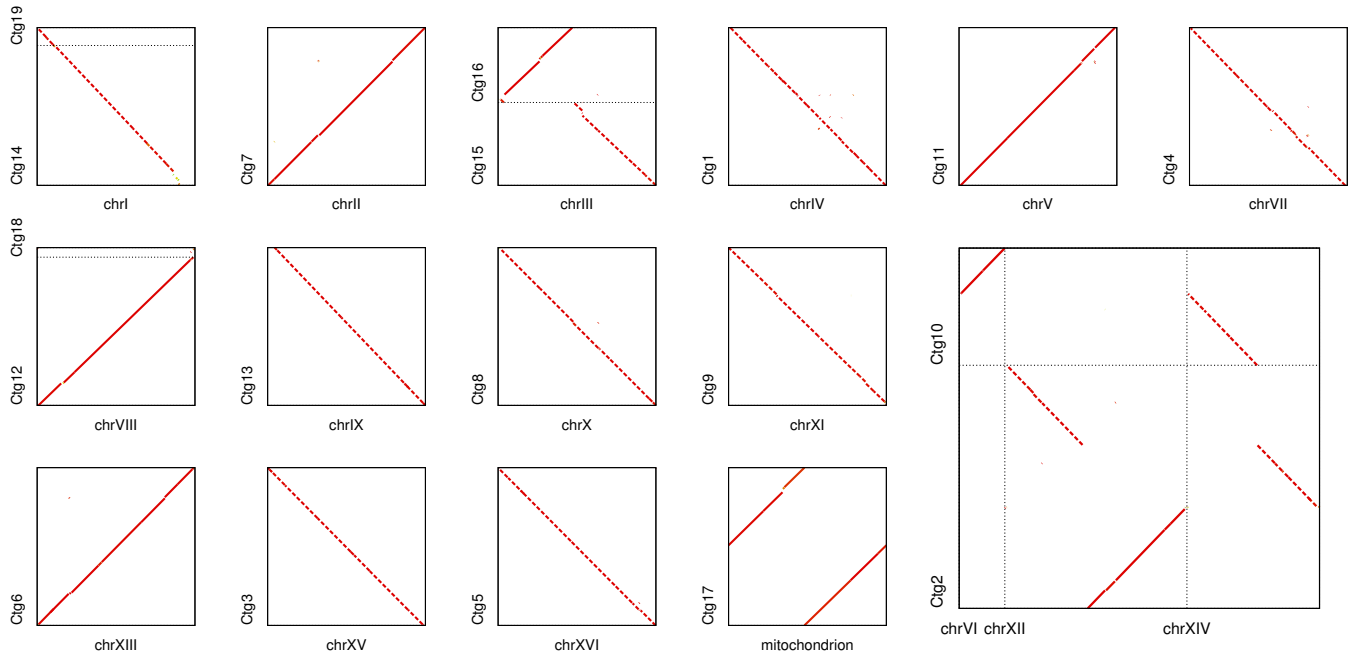


FIGURE A.3: Alignment of the `npScarf`'s assembly for *S. cerevisiae* W303 to the reference genome of the S288C strain. Ten chromosomes (II, IV, V, VII, IX, X, XI, XIII, XV, XVI and the mitochondrion) were constructed into individual contigs, three chromosomes (I, III and VIII) were into two contigs each, and three chromosomes (VI, XII and XIV) were fused into two contigs because of misassembly.

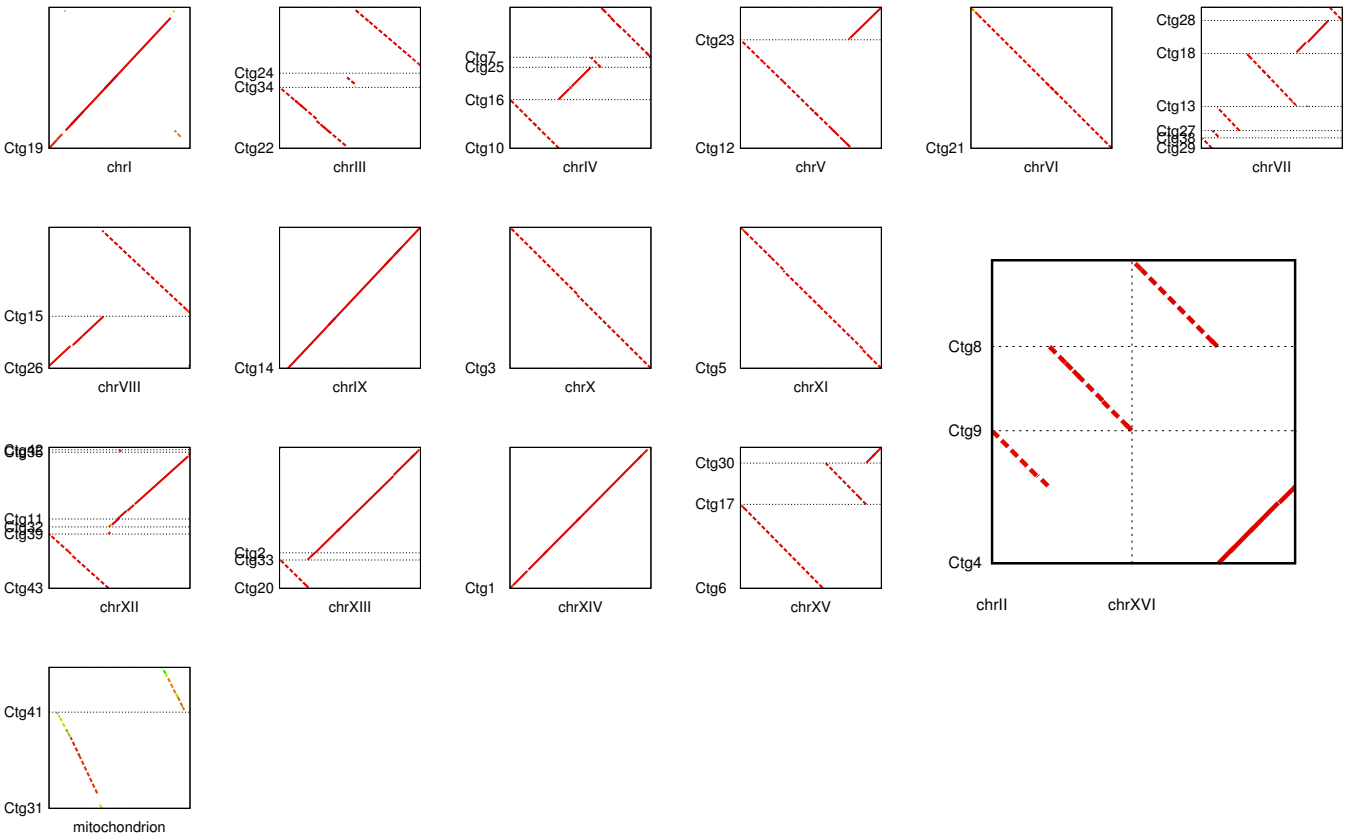


FIGURE A.4: Alignment of the Canu's assembly for *S. cerevisiae* W303 to the reference genome of the S288C strain. Chromosomes II and XVI were fused onto three contigs.

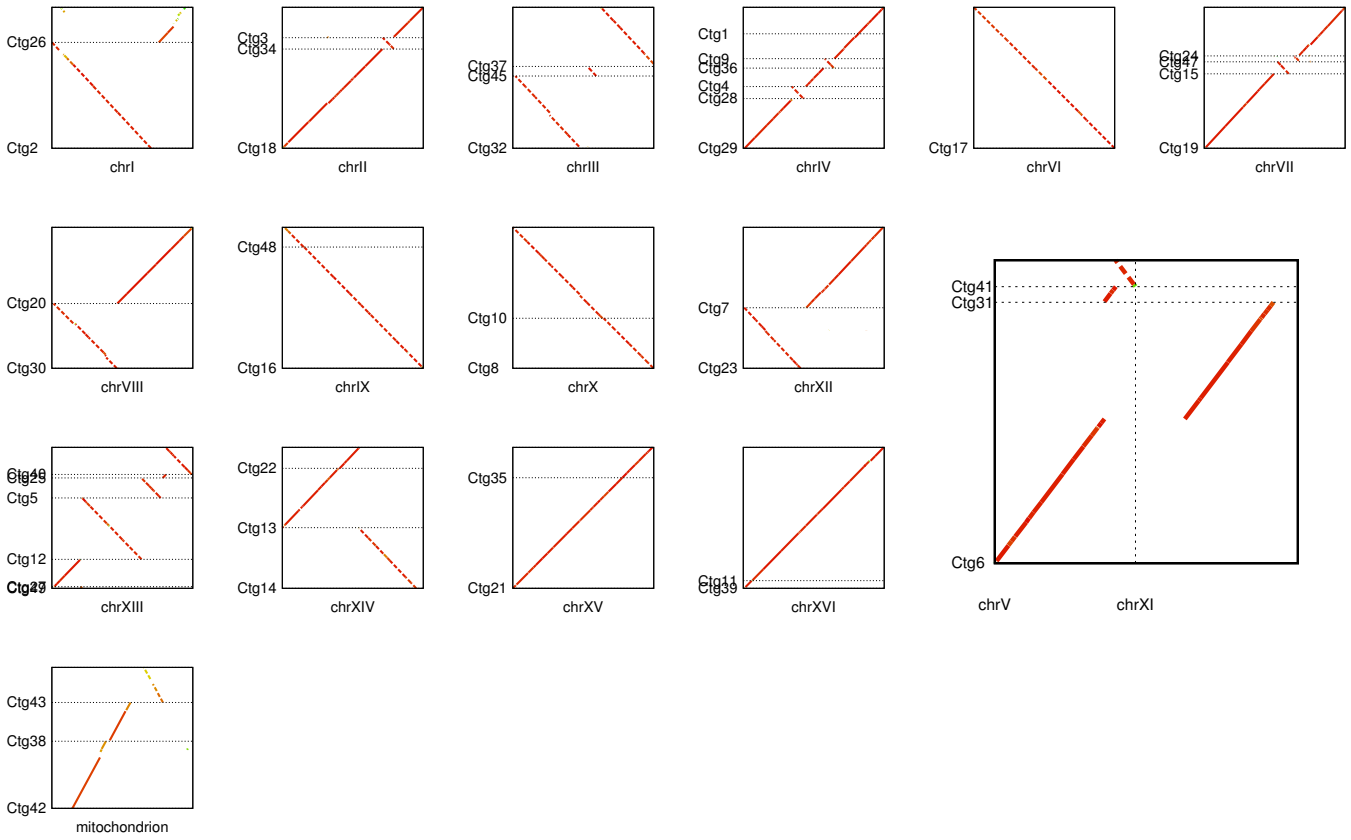


Figure A.5: Alignment of the miniasm's assembly for *S. cerevisiae* W303 to the reference genome of the S288C strain. Chromosomes V and XI were fused onto three contigs.

Supplementary Table A.1 shows the memory usage of the tools on the datasets described in the paper. The scaffolders (SSPACE, LINK and `npScarf`) were run on the short read assemblies outputed by SPAdes. We hence only present the memory footprint from running these tools only. The memory requirement for each pipeline should be the *maximum* between memory usage of the tool and SPAdes. For NaS and Nanocorr, the error correction steps were distributed across hundreds of jobs, each consumed a small memory footprint. We report here only the memory usage from running The values reported in the table were from running Celera Assembler. Note that we ran Celera Assembler on differing configurations, and we reported the here the memory usage of the configuration resulting in the most complete assembly. Memory reported for Canu and Miniasm was the *maximum* among all tasks in the pipeline (including Pilon and BWA-MEM) because each task can be run one at a time; However, the reported memory usage for `npScarf` was the *sum* of memory for running `npScarf` and BWA-MEM.

TABLE A.1: Memory usage (Gb) of the different tools

	Kp2146	**Kp13883**	**E. coli K12**	**ST H58**	**Sc W303**
SPAdes	35.67	34.32	34.24	10.45	85.99
+ SSPACE	3.41	4.01	5.09	2.39	36.84
+ LINK	28.47	16.33	44.83	19.89	233.49
+ `npScarf` (rt)	2.11	1.11	2.32	1.27	4.27
NaS + CA	8.02	8.10	9.21	8.23	83.60
Nanocorr + CA	3.76	3.99	6.91	1.57	159.91
Canu + Pilon	-	6.78	6.30	-	56.20
Miniasm + Pilon	2.99	6.08	6.04	1.97	89.51

B

Supplementary materials for Chapter 3

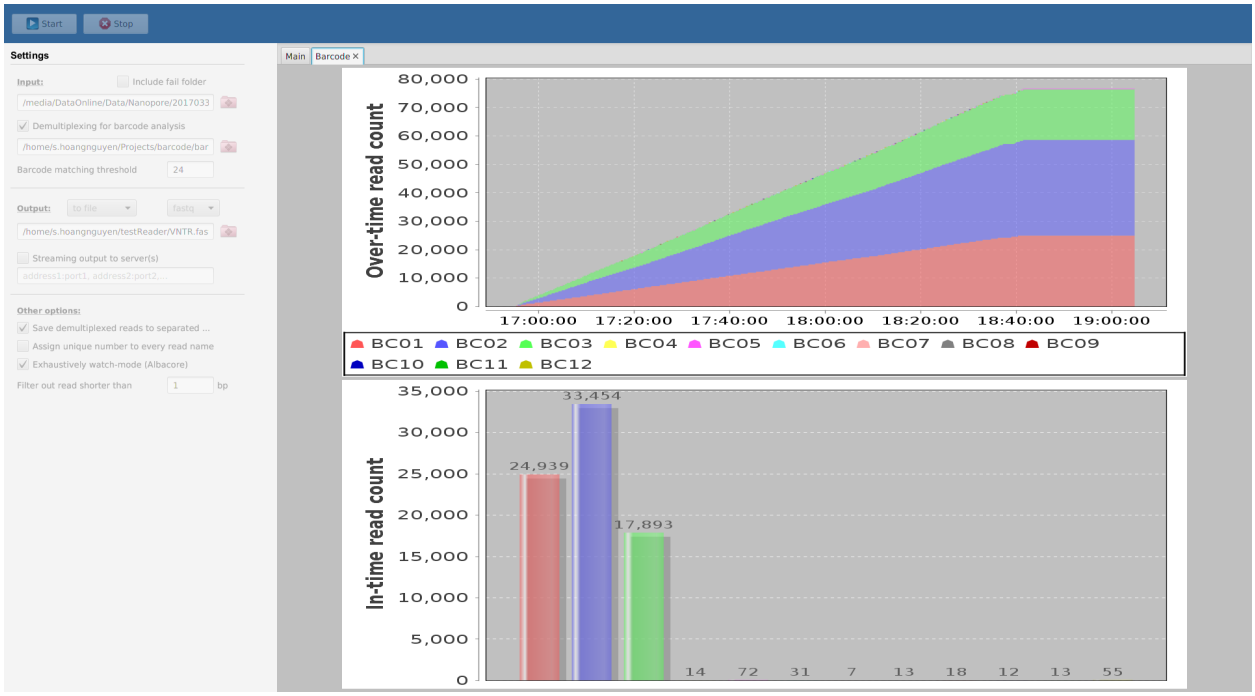


FIGURE B.1: Another application of `npBarcode` with GUI for ONT sequencing using PCR barcoding kit (3 libraries, Albacore base-caller).

TABLE B.1: Information for each of 8 samples used in the Native Barcoding sequencing protocol.

ID	Strain	Publish ID	Barcode ID	Input (ng)
GP_023	*Streptococcus pneumoniae*	ATCC 700677	NB01	1,000
GN_092	*Klebsiella pneumoniae*	3_GR_13 [122]	NB02	1,500
GN_093	*Acinetobacter baumannii*	n/a	NB03	1,500
GN_096	*Klebsiella pneumoniae*	5_GR_13 [122]	NB04	935
GN_101	*Pseudomonas aeruginosa*	n/a	NB05	1,500
GN_106	*Klebsiella pneumoniae*	11_BR_13 [122]	NB06	1,500
GN_132	*Klebsiella quasipneumoniae*	21_GR_13 [122]	NB07	1,000
GN_133	*Klebsiella pneumoniae*	22_GR_12 [122]	NB08	1,500

Table B.2: Pairwise comparison (using MUMmer) between 8 samples in our Native Barcode Sequencing. Value in cell (i, j) is percentage (bases) of genome i aligned to genome j. Highly identical genomes and their figures are highlighted.

	GP_023	**GN_092**	GN_093	**GN_096**	GN_101	**GN_106**	**GN_132**	**GN_133**
GP_023	100	0.30	0.16	0.27	0.09	0.34	0.28	0.30
GN_092	0.14	100	0.30	**87.28**	1.09	**92.74**	**78.42**	**97.11**
GN_093	0.13	0.32	100	0.32	0.30	0.25	0.23	0.3
GN_096	0.14	**88.84**	0.31	100	1.27	**88.10**	**79.75**	**86.24**
GN_101	0.03	0.90	0.15	1.00	100	1.02	0.73	0.72
GN_106	0.17	**93.96**	0.17	**87.47**	1.19	100	**79.65**	**91.86**
GN_132	0.13	**79.99**	0.16	**79.94**	0.84	**80.50**	100	**79.66**
GN_133	0.15	**99.99**	0.26	**87.31**	0.93	**93.47**	**80.50**	100

Table B.3: True Positive and True Negative rate on identification of Gram Negative species GP_023.
It can be seen that `npBarcode` has the highest rate of true positive (sensitivity 88.26%) and the lowest rate of true negative (specificity 99.19%). The common behavior for all demultiplex algorithms is to takes higher/lower risk of wrong classification in exchange for the greater/lesser discovery rate possible. This ratio has to be managed in a proper way depending on different situations yet can be adjusted by parameter calibration. For this use case with default parameter set, `npBarcode`, Porechop and Metrichor built-in demultiplex return comparable results while poreFUME is slightly more conservative.

	npBarcode	**Porechop**	**poreFUME**	**Metrichor**
True Positive	2,924	2,812	2,515	2,875
True Negative	22,125	22,147	22,177	22,155
False Positive	181	159	129	151
False Negative	389	501	798	438
Sensitivity (%)	88.26	84.88	75.91	86.78
Specificity (%)	99.19	99.29	99.42	99.32

TABLE B.4: Real-time emulation of time to detect resistance genes from DNA sequencing. **Bold** represents genes or gene family detected in final assembly and # displays more than three genes grouped within this family. Genes displayed in order of time detected. Brackets represent class of antibiotic this gene confers resistance which includes: A, aminoglycoside; B, beta-lactam; F; fosfomycin; Fu; fusidic acid, M, macrolide; P, phenicol; Q, quinolone; R, rifampicin; S, sulphonamide; T, tetracycline; Tr, trimethoprim; V, vancomycin.

Time (mins)	Resistance gene(s) detected
Isolate: 1_GR_13 Total run time: 1279 mins	
10	**oqxA** (Q), **dfrA1** (Tr), **dfrA14** (Tr), **dfrA23** (Tr), **blaVIM-27#** (B), **sul1/3** (S), **strA** (A), **aph(3)-Ia/c** (A), strB (A), aadB (A), **mph(A** (M), **blaTEM-1B#** (B), **oqxB** (Q), **rmtB2** (A)
30	sul2 (S), **ARR-2/3/6** (R), **blaOXA-10#** (B), **blaVEB-1#** (B), **tet(G)** (T), **cmlA1** (P), floR (P), **blaSHV-11#** (B), **fosA** (F)
60	ARR-3 (R), aac(6)Ib# (A), ARR-7 (R), aac(2) (A), cml (P)
120	aac(3)-IIIc (A)
300	aac(3)-IIIb (A), aac(6)-Ic (A),
600	blaPAO (B), aph(3)-IIb (A), aph(6)-Ic (A)
900	catpC233 (P), blaOKP (B)
1200	**tet(A)** (T)
Isolate: 2_GR_12 Total run time: 2468 mins	
10	**blaKPC-2#** (B), **blaTEM-1A#**(B), **aac6Ib/-cr#** (A), **cmlA1** (P), **dfrA12** (Tr), **sul1/ 3** (S)
30	**rmtB2** (A), **oqxA** (Q), **dfrA14** (Tr), blaPAO (B)
60	**oqxB** (Q), strA (A), strB (A), **aph(3)-1a/c** (A), **sul2** (S), **blaSHV-11/12#** (B), **mph(A)** (M), **catA1** (P), **tet(G)** (T), **blaOXA-9#** (B), **aadA1/2#** (A), fusB (Fu), **aac(6)/aph(2)** (A)
120	**blaOXA-10#** (B), floR (P), **ARR-2/3/6** (R), aadB (A), **fosA** (F), **blaVEB-1#** (B), cml (P), **dfrA23** (Tr)
300	aac(3)-IIIc (A), catpC233 (P), aac(3)-IIIb (A), ARR-3 (R)
600	**tet(A)** (T)
900	aph(6)-Ic (A)
1200	-
Isolate: 16_GR_13 Total run time: 1277 mins	
10	blaOXA-436 (B), **sul1**/3 (S), **dfrA12** (Tr), **fosA** (F), aadB (A), strA (A), **sul2** (S), strB (A), blaCTX-M-64 (B)
30	**blaOXA-48#** (B), **aph(3)-1a/c** (A), **mph(A)** (M), **rmtB2** (A), floR (P), **blaCTX-M-15#** (B), **aac(6)Ib-cr#** (A), **blaOXA-1#** (B), **oqxB** (Q), **oqxA** (Q), **blaTEM-1B#** (B), **blaVEB-1#** (B), **cmlA1** (P)
60	**tet(G)** (T), **blaOXA-10#** (B), **blaSHV-11#** (B), **aac(3)-IIa#** (A), **ARR-2**/3/6 (R)
120	**aadA1/2#** (A), fusB (Fu), ermT (M), str (A), rmtG (A), **aac(6)**/aph(2) (A)
300	ARR-3 (R), catpC233 (P), cml (P), aph(6)-Ic (A), aac(3)-IIIb (A)
600	vanR (V), dfrA14 (Tr)
900	blaPAO (B)
1200	aph(3)-IIb (A)
Isolate: 20_GR_12 Total run time: 1277 mins	
10	**aac(6)Ib/-cr#** (A), **blaTEM-1A#** (B), **dfrA14** (Tr), **sul2** (S), strB (A), **blaKPC-2#** (B), **tet(A)** (T)
30	**oqxA** (Q), **blaSHV-11/12#** (B), **aph(3)-Ia** (A), **fosA** (F), **blaOXA-9** (B), **oqxB** (Q), aph(6)-Ic (A)
60	-
120	aac(2) (A), catpC233 (P), aac(3)-IIIb (A)
300	ermT (M), blaPAO (B), **aac(6)**/aph(2) (A), catpC221 (P)
600	rmtf (A), ermG (M)
900	aac(3)-IIIc (A), aac(6)-Ic (A)
1200	vatB (M), ARR-2/3/6 (R), aadB (A)

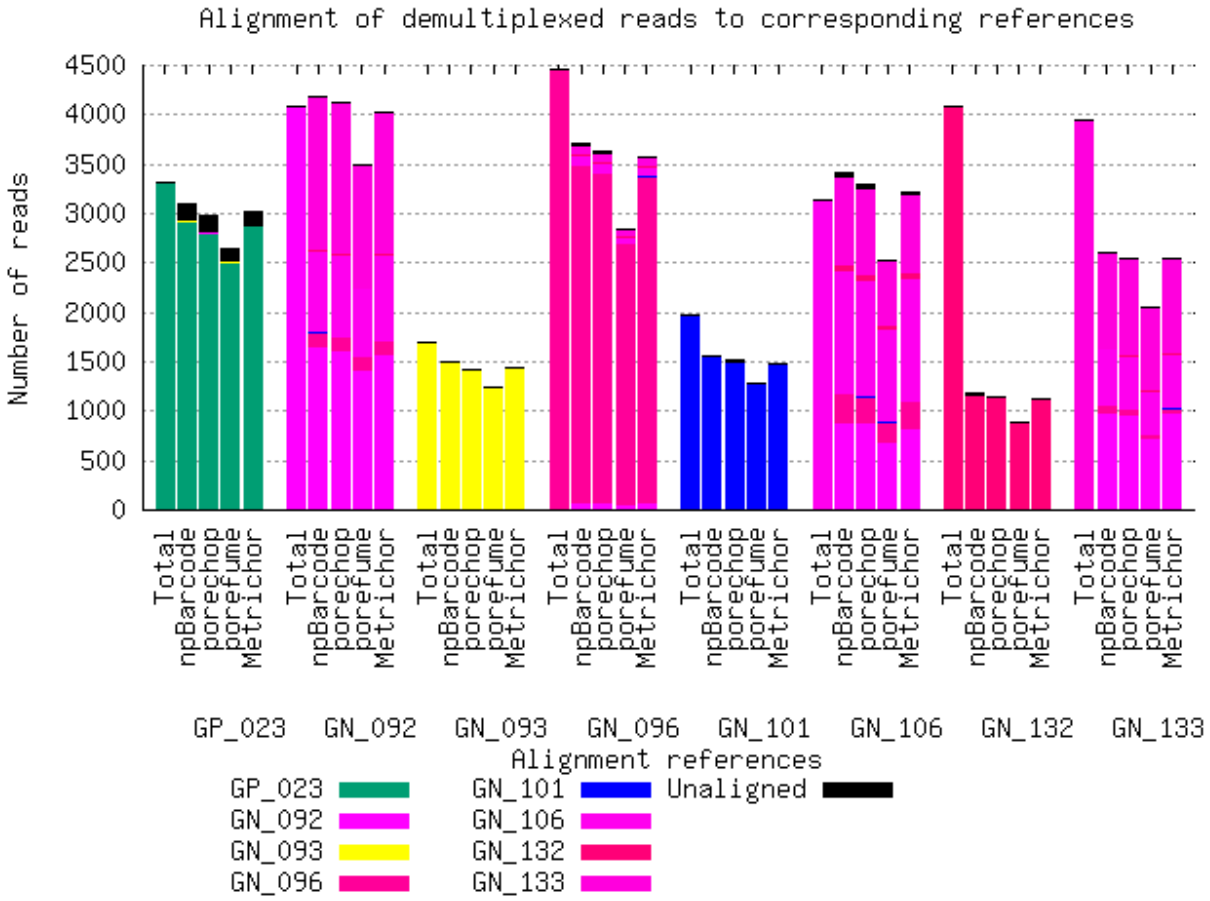


Figure B.2: Comparison of ONT Native Barcode Sequencing (8 libraries) demultiplexing accuracy. The bars present the number of aligned reads to references. Each group of bars correspond to a demultiplex bin. Bars in a group, from left to right, are from (1) the total dataset (2) `npBarcode` (3) Porechop (4) poreFUME demultiplex. GN_092, GN_106, GN_133 together with GN_096 and GN_132 are very close thus being filled by similar colors (pink-red).

Listing B.1: An example of *script.sh* used for `npBarcode`, assuming that all SPAdes output folders are located in the same directory as this script and have name containing the barcoded sample.

```
#!/bin/bash
dirname=‘find . -maxdepth 1 -type d -name "*${1}*" -print -quit‘
bwa index ${dirname}/contigs.fasta
bwa mem -t 16 -k11 -W20 -r10 -A1 -B1 -O1 -E1 -L0 -a -Y -K 10000 \
        ${dirname}/contigs.fasta - 2> /dev/null | \
jsa.np.npscarf -realtime -read 100 -time 1 -b - \
        -seq ${dirname}/contigs.fasta -spadesDir ${dirname} \
        -prefix ${1} > ${1}.log 2>&1
```

C

Supplementary materials for Chapter 4

TABLE C.1: Benchmarking `npGraph` against `npScarf` versions, `hybridSPAdes` and `Unicycler` hybrid assembler with the synthetic dataset.

Method	Assembly size (bp)	#Contigs	N50 (bp)	Mis-assemblies	Mismatch (per 100Kbp)	Indel (per 100Kbp)
random sequences no repeats						
npScarf	4110000	3	4000000	0	0.00	0.00
npScarf_wag	4109516	3	4000000	0	0.00	0.00
npGraph-bwa	4110000	3	4000000	0	0.00	0.00
npGraph-mm2	4110000	3	4000000	0	0.00	0.00
hybridSPAdes	4110231	3	4000077	0	0.00	0.07
Unicycler	4110000	3	4000000	0	0.00	0.00
random sequences some repeats						
npScarf	4109612	3	4001483	0	0.00	1.22
npScarf_wag	4108128	3	3999999	0	0.00	0.95
npGraph-bwa	4110000	3	4000000	0	0.00	0.00
npGraph-mm2	4110000	3	4000000	0	0.00	0.00
hybridSPAdes	4108283	3	4000077	0	0.02	0.05
Unicycler	4110000	3	4000000	0	0.00	0.00
random sequences many repeats						
npScarf	4251391	9	3952225	27	0.88	7.86
npScarf_wag	4554192	9	3999620	37	0.32	5.84
npGraph-bwa	4110000	3	4000000	0	0.32	0.15
npGraph-mm2	4110000	3	4000000	0	0.32	0.15
hybridSPAdes	4108646	3	4000077	0	0.68	0.17
Unicycler	4110000	3	4000000	0	0.32	0.15

Continued on next page

Table C.1 – *Continued from previous page*

Method	Assembly size (bp)	#Contigs	N50 (bp)	Mis-assemblies	Mismatch (per 100Kbp)	Indel (per 100Kbp)
Acinetobacter A1						
npScarf	3918192	3	3899455	4	3.15	0.77
npScarf_wag	4188998	5	3286149	6	6.94	1.87
npGraph-bwa	3917160	2	3908429	1	19.80	2.25
npGraph-mm2	3918048	2	3909317	4	21.42	1.51
hybridSPAdes	3886253	53	1403208	0	5.62	0.26
Unicycler	3917745	2	3909014	0	2.50	0.13
Acinetobacter AB30						
npScarf	4594626	11	4299677	1	69.95	51.04
npScarf_wag	-	-	-	-	-	-
npGraph-bwa	4317408	6	2766938	1	38.10	1.72
npGraph-mm2	4336804	1	4336804	0	23.16	1.55
hybridSPAdes	4286627	49	3308039	0	4.60	0.44
Unicycler	4333041	1	4333041	1	6.42	0.53
E. coli K12 MG1655						
npScarf	4647010	5	4609437	9	10.65	17.61
npScarf_wag	4678785	5	4641364	0	8.38	2.31
npGraph-bwa	4643368	1	4643368	0	9.37	0.75
npGraph-mm2	4643265	1	4643265	0	10.43	1.25
hybridSPAdes	4641097	1	4641097	0	1.19	0.13
Unicycler	4641650	1	4641650	0	3.43	0.26

Continued on next page

Table C.1 – *Continued from previous page*

Method	Assembly size (bp)	#Contigs	N50 (bp)	Mis-assemblies	Mismatch (per 100Kbp)	Indel (per 100Kbp)
E. coli O25b H4-ST131						
npScarf	5353639	7	5087544	14	22.43	6.69
npScarf_wag	5413499	7	5108146	6	26.34	3.95
npGraph-bwa	5251882	3	5112329	0	15.26	1.11
npGraph-mm2	5250391	3	5110666	0	13.56	1.05
hybridSPAdes	5249315	7	5109649	0	2.23	0.42
Unicycler	5249442	3	5109760	0	4.02	0.27
Klebsiella 30660 NJST258 1						
npScarf	5739591	9	5257627	8	11.55	12.93
npScarf_wag	5527526	5	5264334	0	9.86	2.43
npGraph-bwa	5535507	8	5264082	2	12.77	1.50
npGraph-mm2	5534879	8	5263608	0	6.49	1.14
hybridSPAdes	5528124	8	5263320	0	1.36	0.80
Unicycler	5537860	9	5263196	0	1.34	0.51
Klebsiella MGH 78578						
npScarf	5676644	6	5308928	17	20.26	14.74
npScarf_wag	5672565	5	5315270	6	19.15	3.73
npGraph-bwa	5698292	6	5316232	1	23.94	2.34
npGraph-mm2	5696273	6	5315757	0	19.35	1.84
hybridSPAdes	5665442	19	5315086	0	5.28	0.92
Unicycler	5694231	14	5315096	0	5.38	0.21

Continued on next page

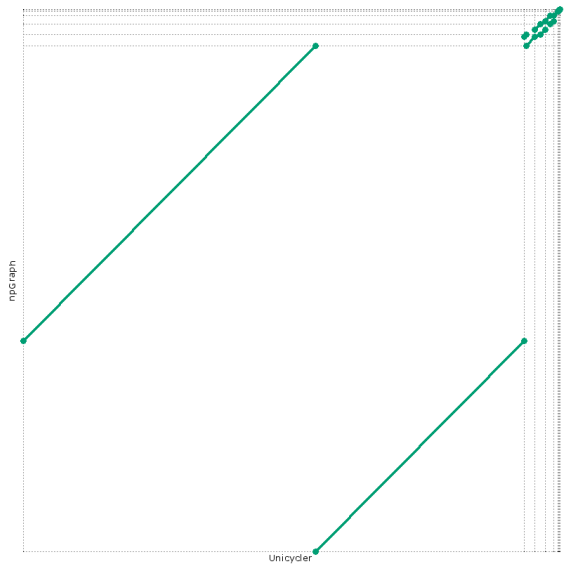
Table C.1 – *Continued from previous page*

Method	Assembly size (bp)	#Contigs	N50 (bp)	Mis-assemblies	Mismatch (per 100Kbp)	Indel (per 100Kbp)
Klebsiella NTUH-K2044						
npScarf	5468717	3	5239437	5	5.53	1.92
npScarf_wag	5471159	2	5249462	0	6.98	1.72
npGraph-bwa	5472856	2	5248714	0	8.06	1.41
npGraph-mm2	5472845	2	5248703	0	7.07	1.15
hybridSPAdes	5472760	2	5248601	0	2.34	0.60
Unicycler	5472697	2	5248545	0	2.41	0.35
Mycobacterium tuberculosis H37Rv						
npScarf	4446095	4	4389968	12	7.61	3.53
npScarf_wag	4407553	1	4407553	2	4.21	2.80
npGraph-bwa	4411594	1	4411594	0	6.21	1.07
npGraph-mm2	4411387	1	4411387	0	6.28	0.73
hybridSPAdes	4411162	1	4411162	0	1.41	0.32
Unicycler	4411538	1	4411538	0	2.22	0.34
Saccharomyces cerevisiae S288c						
npScarf	11875932	20	896738	48	77.76	12.63
npScarf_wag	12626448	37	630281	189	43.11	5.16
npGraph-bwa	11907663	37	774482	24	72.70	4.14
npGraph-mm2	11881030	39	795662	30	58.48	3.58
hybridSPAdes	11968398	246	731962	4	12.20	0.89
Unicycler	11847655	72	909114	0	21.81	1.04

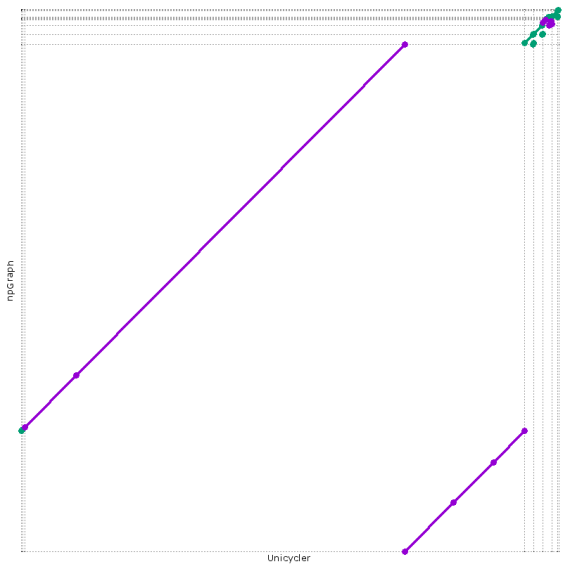
Continued on next page

Table C.1 – *Continued from previous page*

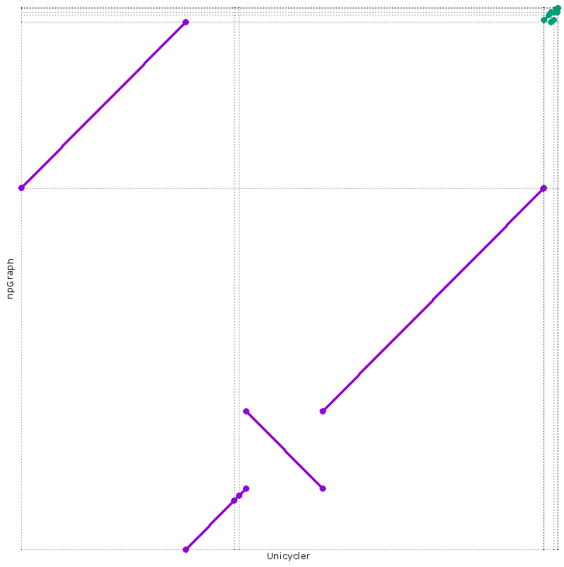
Method	Assembly size (bp)	#Contigs	N50 (bp)	Mis-assemblies	Mismatch (per 100Kbp)	Indel (per 100Kbp)
Shigella dysenteriae Sd197						
npScarf	4037570	70	179761	63	103.66	182.99
npScarf_wag	6571492	23	640874	1154	81.15	53.47
npGraph-bwa	4574433	6	4382010	133	93.64	12.47
npGraph-mm2	4565129	10	1512588	193	86.63	13.29
hybridSPAdes	4562642	244	77009	35	7.14	1.20
Unicycler	4560901	3	4369231	0	11.88	1.05
Shigella sonnei 53G						
npScarf	5335195	77	388533	68	104.26	116.07
npScarf_wag	-	-	-	-	-	-
npGraph-bwa	5210154	4	4987138	14	59.84	1.02
npGraph-mm2	5212051	4	4989035	19	30.70	1.01
hybridSPAdes	5223176	14	4987892	0	3.14	0.27
Unicycler	5220517	5	4988548	0	7.39	0.52
Streptococcus suis BM407						
npScarf	2220434	4	2120037	9	48.28	48.92
npScarf_wag	2245214	4	2128290	3	85.79	3.85
npGraph-bwa	2167238	6	2146682	0	26.08	0.69
npGraph-mm2	2166779	6	2146223	0	21.88	0.65
hybridSPAdes	2147092	48	1437972	0	0.79	0.19
Unicycler	2170829	2	2146250	0	2.67	0.32



(a) *Citrobacter freundii* CAV1374

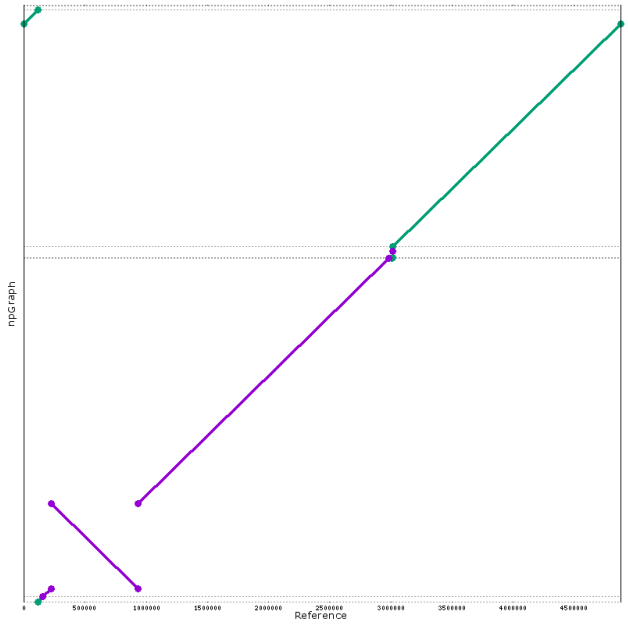


(b) *Klebsieall oxytoca* CAV1015

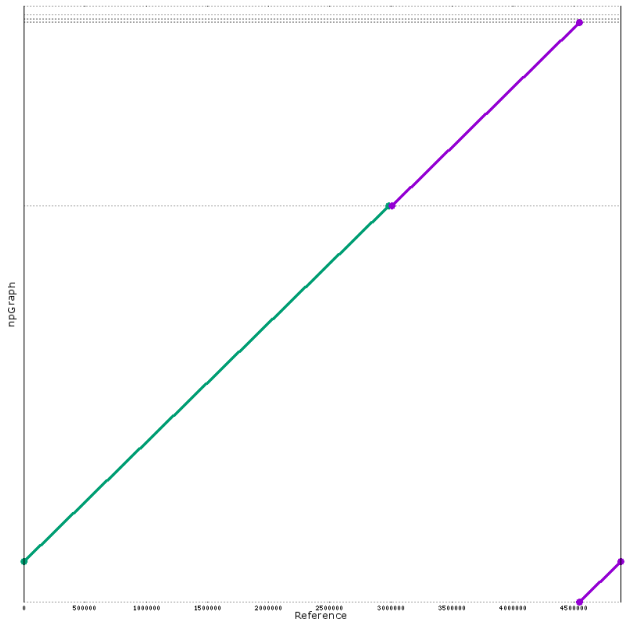


(c) *Enterobacter cloacae* CAV1411

FIGURE C.1: Dotplot generated by MUMmer for assembly results of `Unicycler` versus `npGraph`. Structural agreements between two methods were found in (a) *C.freundii* and (b) *K.oxytoca* assembly contigs. On the other hand, for (c) *E.cloacae* sample, there was a disagreement detected between 2 largest contigs given by two assembly algorithms. This case is investigated more thoroughly by using a reference from a same bacterial strain in Figure C.2.



(a) *E. cloacae* `Unicycler` assembly versus reference genome



(b) *E. cloacae* `npGraph` assembly versus reference genome

FIGURE C.2: Alignments of an *Enterobacter cloacae* reference genome to assembly sequences generated by (a) `Unicycler` and (b) `npGraph`. While the former presents a structural variant, the latter is virtually an 1-to-1 mapping.

D

Supplementary materials for Chapter 5

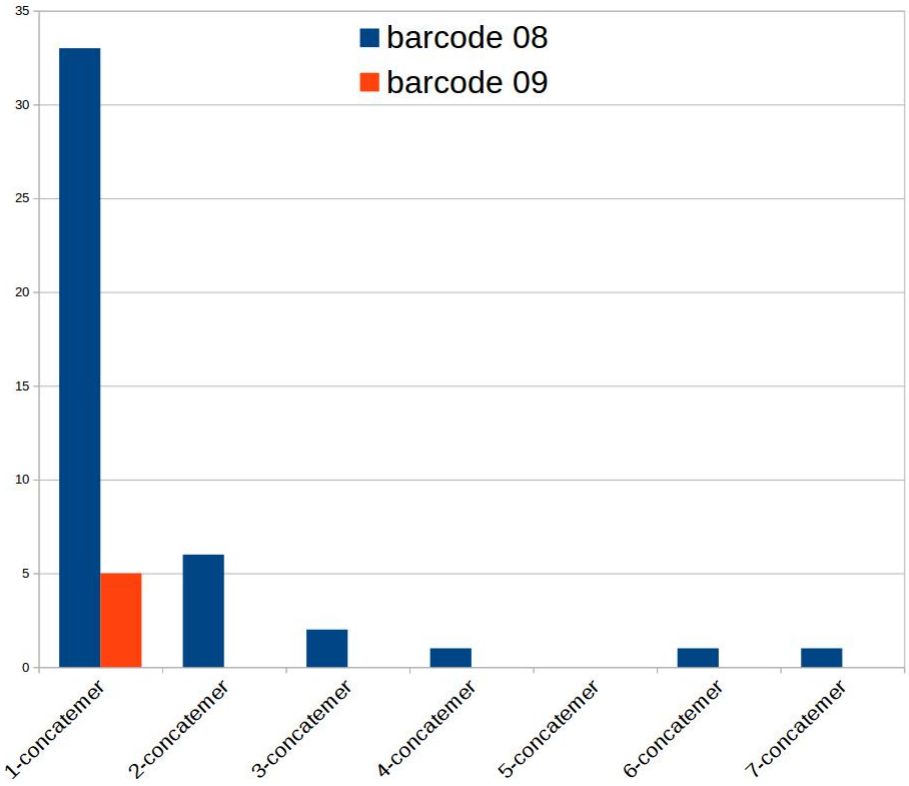
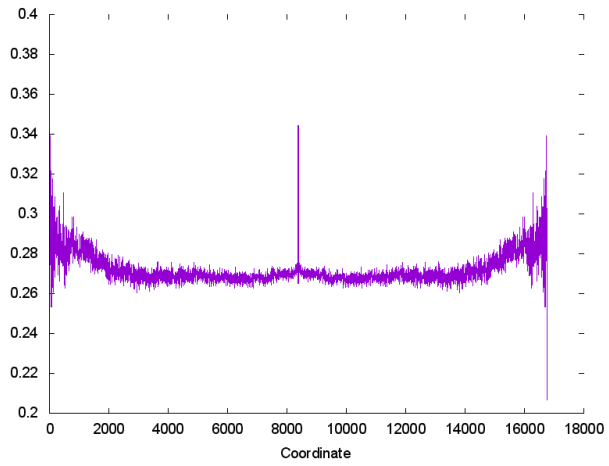
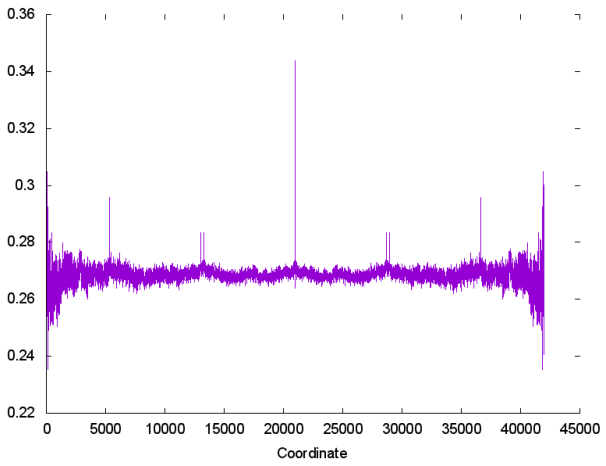


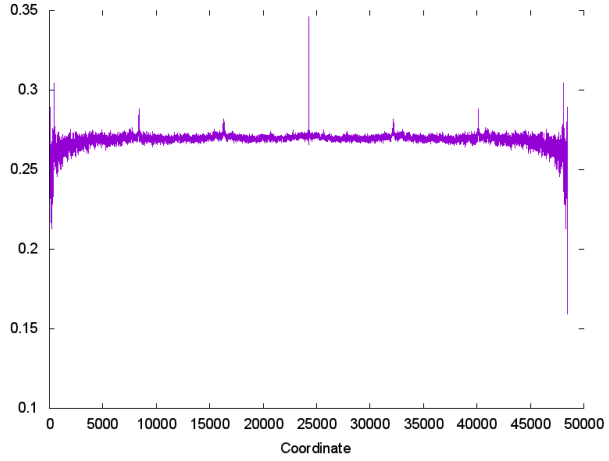
FIGURE D.1: Number of k-concatemers for each positive viral sample with barcode 08 (CaMV) and 09 (BSMYV). $k = 1 \ldots 7$



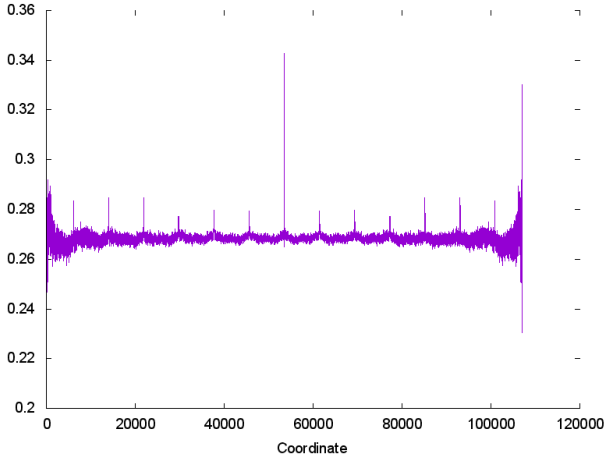
(a) ACF for a monomer read



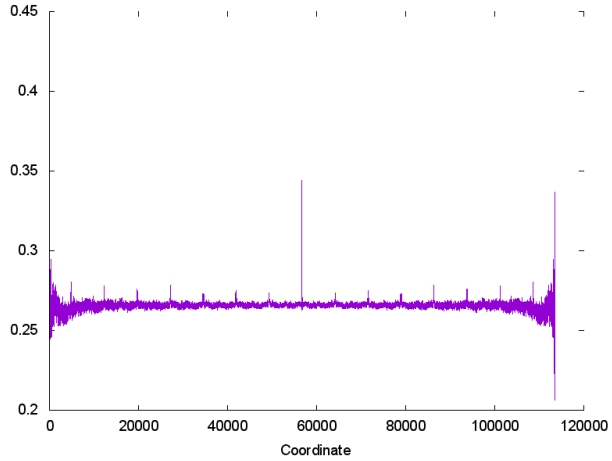
(b) ACF for a 2-concatemers read



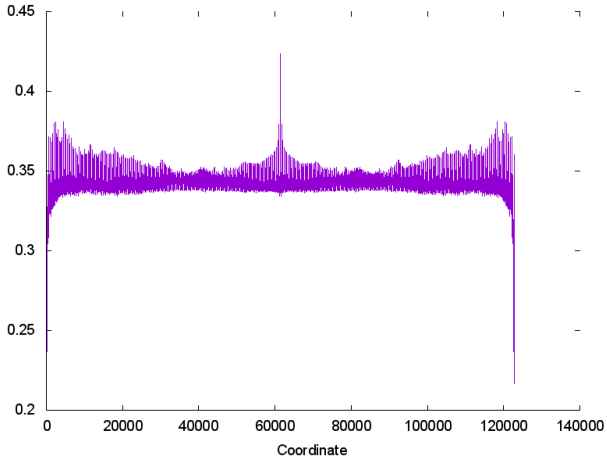
(c) ACF for a 3-concatemers read



(d) ACF for a 6-concatemers read

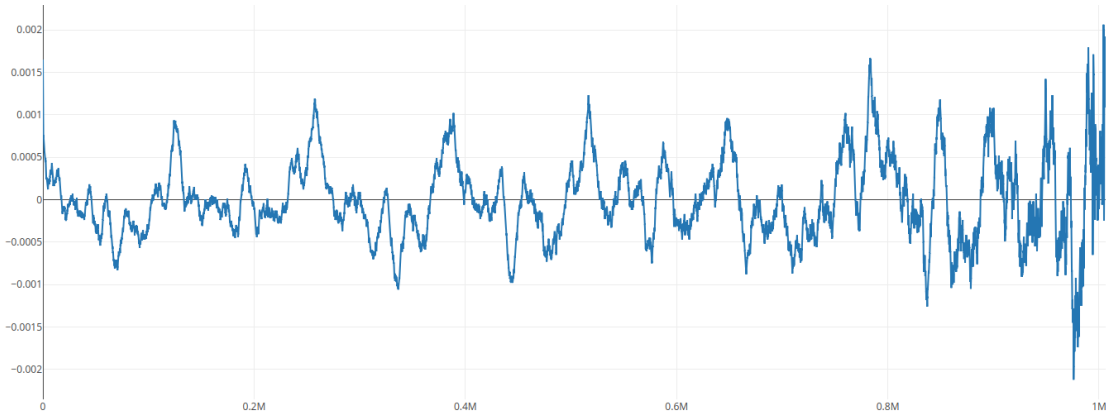


(e) ACF for the 7-concatemer read

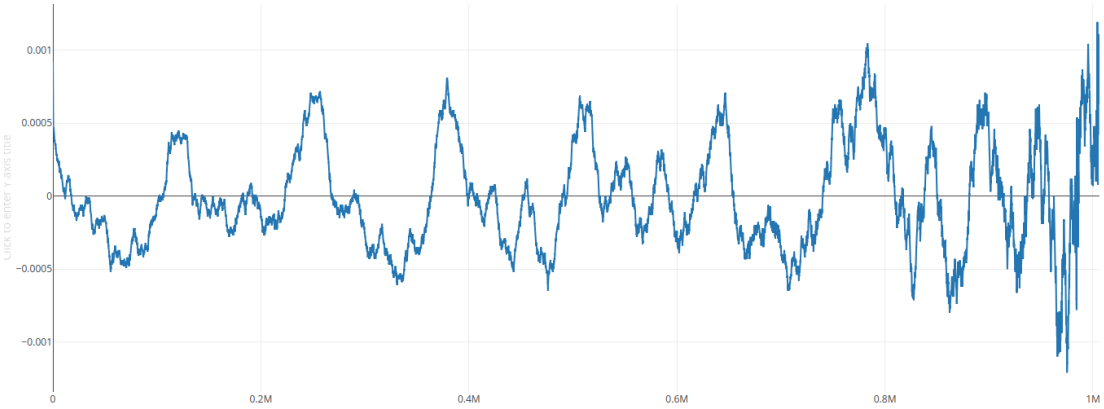


(f) ACF for a random DNA sequence

Figure D.2: ACF values for a random synthetic read and several k-concatemers nanopore read detected in *Cauliflower mosaic* sample (barcode 08).

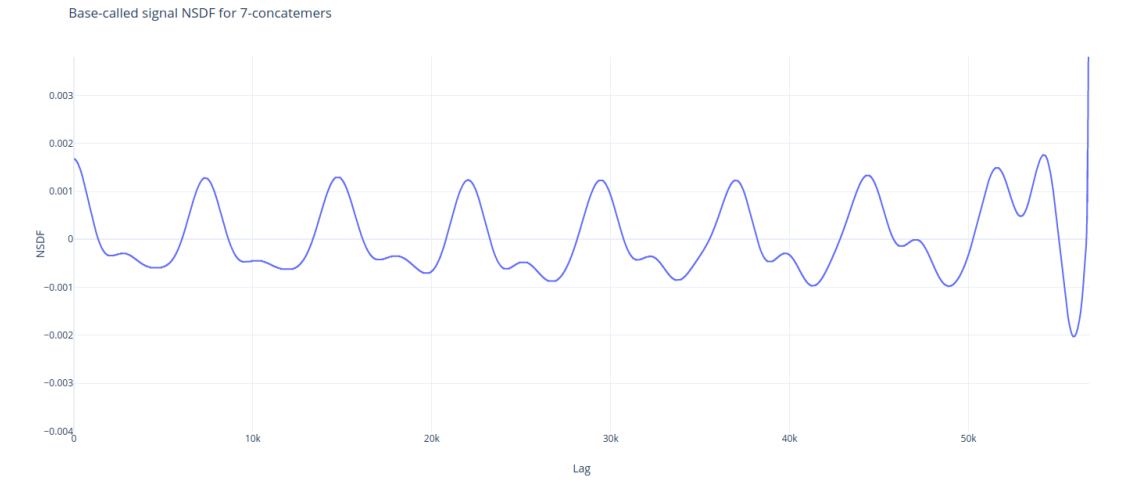


(a) NSDF signal averaged by 10k window

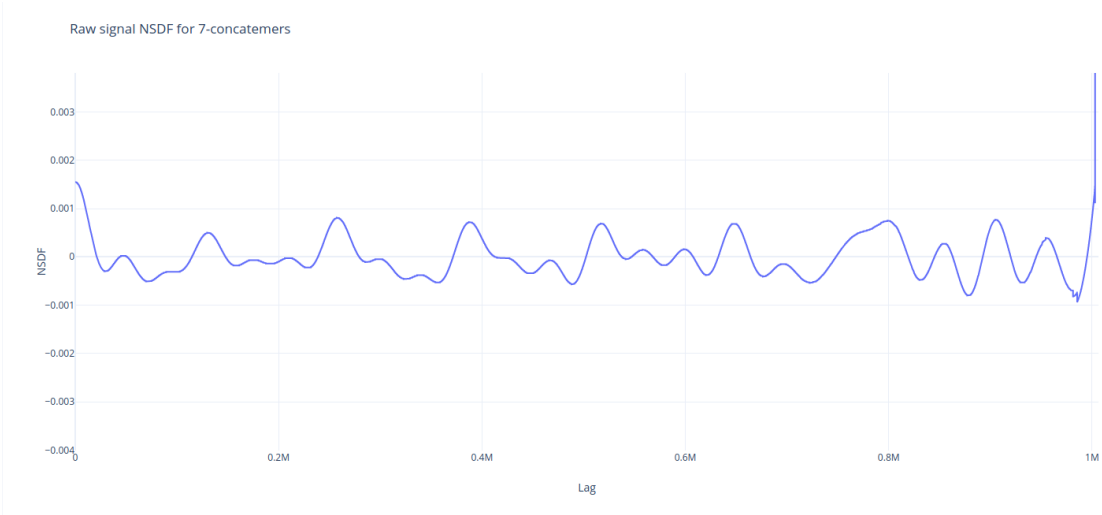


(b) NSDF signal averaged by 20k window

FIGURE D.3: NSDF signal processed by a running average filter step with different window size (ws): (D.3a) $ws = 10,000$ (D.3b) $ws = 20,000$. While the larger window sizes mean better de-noised signals, it also reduces the specificity of the local maxima. As can be seen from the Figure D.3b, the peaking regions are easier to spot but their blunt tips, making it more difficult to determine the exact coordinates. In Figure D.3a the window size $10,000$ returned clear spikes for easier monomer length evaluation. However, the seesaw pattern were not eliminated completely.



(a) NDSF for DNA sequence



(b) NSDF for raw signal

FIGURE D.4: 7-concatemers NSDF signal processed after LPF using $cutFreq = 30$ of: (D.4a) base-called DNA sequence (D.4b) raw signal sequence.

List of Symbols

$G = \{V, E\}$: graph composed of vertices set and edges set

$\overrightarrow{u}, \overrightarrow{v}$: vertex of a bidirected graph with direction property

$v+$, $v-$: DNA content of v spelled in template or reversed complement order

$S = \{s_1, \ldots s_L\}$: a signal sequence of length L

$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$: set of *natural*, *integer*, *real* and *complex* numbers respectively

ELSEVIER LICENSE
TERMS AND CONDITIONS

Mar 07, 2019

This Agreement between Mr. Son Nguyen ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

License Number	4543471436155
License date	Mar 07, 2019
Licensed Content Publisher	Elsevier
Licensed Content Publication	Trends in Microbiology
Licensed Content Title	Rolling-circle amplification of viral DNA genomes using phi29 polymerase
Licensed Content Author	Reimar Johne,Hermann Müller,Annabel Rector,Marc van Ranst,Hans Stevens
Licensed Content Date	May 1, 2009
Licensed Content Volume	17
Licensed Content Issue	5
Licensed Content Pages	7
Start Page	205
End Page	211
Type of Use	reuse in a thesis/dissertation
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Format	electronic
Are you the author of this Elsevier article?	No
Will you be translating?	No
Original figure numbers	Figure 1
Title of your thesis/dissertation	Real-time analysis for Nanopore sequencing data
Expected completion date	Mar 2019
Estimated size (number of pages)	200
Requestor Location	Mr. Son Nguyen 4/66 Cedar Street Greenslopes Brisbane, QLD 4120 Australia Attn: Mr. Son Nguyen
Publisher Tax ID	GB 494 6272 12
Total	0.00 USD
Terms and Conditions	

INTRODUCTION

1. The publisher for this copyrighted material is Elsevier. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions

apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

GENERAL TERMS

2. Elsevier hereby grants you permission to reproduce the aforementioned material subject to the terms and conditions indicated.

3. Acknowledgement: If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:

"Reprinted from Publication title, Vol /edition number, Author(s), Title of article / title of chapter, Pages No., Copyright (Year), with permission from Elsevier [OR APPLICABLE SOCIETY COPYRIGHT OWNER]." Also Lancet special credit - "Reprinted from The Lancet, Vol. number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier."

4. Reproduction of this material is confined to the purpose and/or media for which permission is hereby given.

5. Altering/Modifying Material: Not Permitted. However figures and illustrations may be altered/adapted minimally to serve your work. Any other abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of Elsevier Ltd. (Please contact Elsevier at permissions@elsevier.com). No modifications can be made to any Lancet figures/tables and they must be reproduced in full.

6. If the permission fee for the requested use of our material is waived in this instance, please be advised that your future requests for Elsevier materials may attract a fee.

7. Reservation of Rights: Publisher reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

8. License Contingent Upon Payment: While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by publisher or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received on a timely basis, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

9. Warranties: Publisher makes no representations or warranties with respect to the licensed material.

10. Indemnity: You hereby indemnify and agree to hold harmless publisher and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

11. No Transfer of License: This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

12. No Amendment Except in Writing: This license may not be amended except in a writing signed by both parties (or, in the case of publisher, by CCC on publisher's behalf).

13. Objection to Contrary Terms: Publisher hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement

between you and publisher (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

14. Revocation: Elsevier or Copyright Clearance Center may deny the permissions described in this License at their sole discretion, for any reason or no reason, with a full refund payable to you. Notice of such denial will be made using the contact information provided by you. Failure to receive such notice will not alter or invalidate the denial. In no event will Elsevier or Copyright Clearance Center be responsible or liable for any costs, expenses or damage incurred by you as a result of a denial of your permission request, other than a refund of the amount(s) paid by you to Elsevier and/or Copyright Clearance Center for denied permissions.

LIMITED LICENSE

The following terms and conditions apply only to specific license types:

15. **Translation**: This permission is granted for non-exclusive world **English** rights only unless your license was granted for translation rights. If you licensed translation rights you may only translate this content into the languages you requested. A professional translator must perform all translations and reproduce the content word for word preserving the integrity of the article.

16. **Posting licensed content on any Website**: The following terms and conditions apply as follows: Licensing material from an Elsevier journal: All content posted to the web site must maintain the copyright information line on the bottom of each image; A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or the Elsevier homepage for books at http://www.elsevier.com; Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

Licensing material from an Elsevier book: A hyper-text link must be included to the Elsevier homepage at http://www.elsevier.com . All content posted to the web site must maintain the copyright information line on the bottom of each image.

Posting licensed content on Electronic reserve: In addition to the above the following clauses are applicable: The web site must be password-protected and made available only to bona fide students registered on a relevant course. This permission is granted for 1 year only. You may obtain a new license for future website posting.

17. **For journal authors:** the following clauses are applicable in addition to the above:

Preprints:

A preprint is an author's own write-up of research results and analysis, it has not been peer-reviewed, nor has it had any other value added to it by a publisher (such as formatting, copyright, technical enhancement etc.).

Authors can share their preprints anywhere at any time. Preprints should not be added to or enhanced in any way in order to appear more like, or to substitute for, the final versions of articles however authors can update their preprints on arXiv or RePEc with their Accepted Author Manuscript (see below).

If accepted for publication, we encourage authors to link from the preprint to their formal publication via its DOI. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help users to find, access, cite and use the best available version. Please note that Cell Press, The Lancet and some society-owned have different preprint policies. Information on these policies is available on the journal homepage.

Accepted Author Manuscripts: An accepted author manuscript is the manuscript of an article that has been accepted for publication and which typically includes author-incorporated changes suggested during submission, peer review and editor-author communications.

Authors can share their accepted author manuscript:

- immediately
  - via their non-commercial person homepage or blog
  - by updating a preprint in arXiv or RePEc with the accepted manuscript

- via their research institute or institutional repository for internal institutional uses or as part of an invitation-only research collaboration work-group
- directly by providing copies to their students or to research collaborators for their personal use
- for private scholarly sharing as part of an invitation-only work group on commercial sites with which Elsevier has an agreement

- After the embargo period
  - via non-commercial hosting platforms such as their institutional repository
  - via commercial sites with which Elsevier has an agreement

In all cases accepted manuscripts should:

- link to the formal publication via its DOI
- bear a CC-BY-NC-ND license - this is easy to do
- if aggregated with other manuscripts, for example in a repository or other site, be shared in alignment with our hosting policy not be added to or enhanced in any way to appear more like, or to substitute for, the published journal article.

Published journal article (JPA): A published journal article (PJA) is the definitive final record of published research that appears or will appear in the journal and embodies all value-adding publishing activities including peer review co-ordination, copy-editing, formatting, (if relevant) pagination and online enrichment.
Policies for sharing publishing journal articles differ for subscription and gold open access articles:
Subscription Articles: If you are an author, please share a link to your article rather than the full-text. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help your users to find, access, cite, and use the best available version.
Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.
If you are affiliated with a library that subscribes to ScienceDirect you have additional private sharing rights for others' research accessed under that agreement. This includes use for classroom teaching and internal training at the institution (including use in course packs and courseware programs), and inclusion of the article for grant funding purposes.
Gold Open Access Articles: May be shared according to the author-selected end-user license and should contain a CrossMark logo, the end user license, and a DOI link to the formal publication on ScienceDirect.
Please refer to Elsevier's posting policy for further information.
18. **For book authors** the following clauses are applicable in addition to the above:
Authors are permitted to place a brief summary of their work online only. You are not allowed to download and post the published electronic version of your chapter, nor may you scan the printed edition to create an electronic version. **Posting to a repository:** Authors are permitted to post a summary of their chapter only in their institution's repository.
19. **Thesis/Dissertation**: If your license is for use in a thesis/dissertation your thesis may be submitted to your institution in either print or electronic form. Should your thesis be published commercially, please reapply for permission. These requirements include permission for the Library and Archives of Canada to supply single copies, on demand, of the complete thesis and include permission for Proquest/UMI to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission. Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.

Elsevier Open Access Terms and Conditions
You can publish open access with Elsevier in hundreds of open access journals or in nearly 2000 established subscription journals that support open access publishing. Permitted third party re-use of these open access articles is defined by the author's choice of Creative Commons user license. See our open access license policy for more information.

Terms & Conditions applicable to all Open Access articles published with Elsevier:
Any reuse of the article must not represent the author as endorsing the adaptation of the article nor should the article be modified in such a way as to damage the author's honour or reputation. If any changes have been made, such changes must be clearly indicated.
The author(s) must be appropriately credited and we ask that you include the end user license and a DOI link to the formal publication on ScienceDirect.
If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source it is the responsibility of the user to ensure their reuse complies with the terms and conditions determined by the rights holder.
Additional Terms & Conditions applicable to each Creative Commons user license:
CC BY: The CC-BY license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article and to make commercial use of the Article (including reuse and/or resale of the Article by commercial entities), provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by/4.0.
CC BY NC SA: The CC BY-NC-SA license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article, provided this is not done for commercial purposes, and that the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. Further, any new works must be made available on the same conditions. The full details of the license are available at http://creativecommons.org/licenses/by-nc-sa/4.0.
CC BY NC ND: The CC BY-NC-ND license allows users to copy and distribute the Article, provided this is not done for commercial purposes and further does not permit distribution of the Article if it is changed or edited in any way, and provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, and that the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by-nc-nd/4.0.
Any commercial reuse of Open Access articles published with a CC BY NC SA or CC BY NC ND license requires permission from Elsevier and will be subject to a fee.
Commercial reuse includes:

- Associating advertising with the full text of the Article
- Charging fees for document delivery or access
- Article aggregation
- Systematic distribution via e-mail lists or share buttons

Posting or linking by commercial companies for use by customers of those companies.

20. **Other Conditions**:

v1.9

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.