EVALUATION AND IMPROVEMENT OF GENOME ASSEMBLY

XIE LUYU

NATIONAL UNIVERSITY OF SINGAPORE

2019

EVALUATION AND IMPROVEMENT OF GENOME ASSEMBLY

XIE LUYU

(B.S. Fudan University)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2019

Supervisor:

Professor Wong Lim Soon

Examiners:

Professor Sung Wing Kin

Professor Zhang Louxin

Professor Tak-Wah Lam, The University of Hong Kong

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

XIE Luyu

XIE Luyu

August 2019

Acknowledgments

My deepest gratitude goes first and foremost to my supervisor Prof. Wong Limsoon, for his patience, encouragement, and guidance. His expertise, profound knowledge, and logical thinking inspired me and guided me in my approach to academic research. He also gave me a great freedom to explore many topics and discover my interests. This made my graduate experience enjoyable.

Besides my supervisor, I thank the rest of my thesis advisory committee: Prof. Sung Wing Kin and Prof. Tan Kian Lee. Their insightful comments pointed out the deficiencies of my work, and motivated me to make a significant improvement.

My sincere thanks also go to my undergraduate supervisor Prof. Zhou Shuigeng, for his encouragement and advice on my career.

I thank my seniors Wilson Goh Wen Bin, Fan Mengyuan, Lim Junliang Kevin, Yong Chern Han, Abha Belorkar, Iana Pyrogova, Yang Ruijie, Lu Bingxin, Ma Junqi, and Liu Hui, for their useful information and inspiring suggestions.

I thank my friends and labmates: Jin Wenhao, Wang Zhiqiang, Stefano Perna, Ramesh Rajaby, Mohammad Neamul Kabir, Chan Wei Xin, Herty Liany, and Xu Weinan, for many discussions and all the fun we had.

Last but not the least, I thank my parents Prof. Xie Baogui and Prof. Jiang Yuji, for their constant encouragement and support in my life and study. Their suggestions helped me find the research topics I'm interested in, which is finally developed into this thesis.

Contents

Summary

After ten years of development, Next-generation sequencing (NGS) has already been successfully commercialized and widely applied in many scenarios. As a basic application, genomic *de novo* assembly has also benefited from the development of many assemblers. During a comparison between assemblies from different assemblers, it was found that existing metrics only provided limited insight on specific aspects of genome assembly quality, and sometimes even disagreed with each other. For better integrative comparison between assemblies, I propose a new genome assembly metric, PDR, in this thesis. It derives from a common question in genetic studies, and takes completeness, contiguity, and correctness into consideration. The comparison between PDR and other metrics on publicly available datasets showed its ability to integratively assess the quality of a genome assembly. In fact, this ability is guaranteed by its definition.

Beyond assemblers, many assembly strategies were also developed, e.g. mate-pair library, optical map, chromatin interaction, and genetic map. Among these strategies, genetic map is the most widely adopted one in breeding studies. But genetic map construction requires numerous computational resources and underperforms when draft assembly contains misassembly. To address these limitations, I propose a new assembly-improving method, CAST, in this thesis. It corrects and scaffolds a draft assembly by genetic information in progenies' NGS data, without the construction of a genetic map. In theory, it first splits the draft assembly at genetically incoherent positions, and then scaffolds contigs at genetically coherent positions. The evaluation on two public datasets suggests that CAST was able to significantly improve a draft assembly's contiguity and correctness.

List of Figures

List of Tables

Chapter 1

Overview

1.1 Introduction

1.1.1 Sequencing

Sequencing is a powerful tool for investigation of genomes. It is instrumental to revealing the underlying mechanisms of many biological processes. The Sanger sequencing technique, also known as First-Generation Sequencing, was presented in 1977 [Sanger et al., 1977]; however, it was very time-consuming and costly. In the late 1990s, some high-throughput sequencing techniques called Next-Generation Sequencing(NGS) were developed and commercialized. Subsequently, the cost of NGS has decreased dramatically as shown in Figure 1.1 [Wetterstrand, 2013].

In 2006, it cost 14 million dollars to sequence a human genome; in contrast, that only cost about 1000 dollars in 2016. This makes sequencing affordable for a lot of research projects and even some clinical applications. There are also several new sequencing techniques developed in recent years, with read length longer than ten thousand bases. However, up to now, these long-read length techniques are relatively expensive and error-prone.

1.1.2 Genome assembly

The reads from sequencers are small fragmented segments of a sample's genome. When a reference genome is absent, *de novo* assembly is usually performed to provide a global view of the sample's genome. Along with sequencing technologies, assembly methods have also evolved over the last ten years. Different representations were proposed to better describe and model information in sequencing data. Overlap-Layout-Consensus methods like Celera Assembler [Myers et al., 2000]

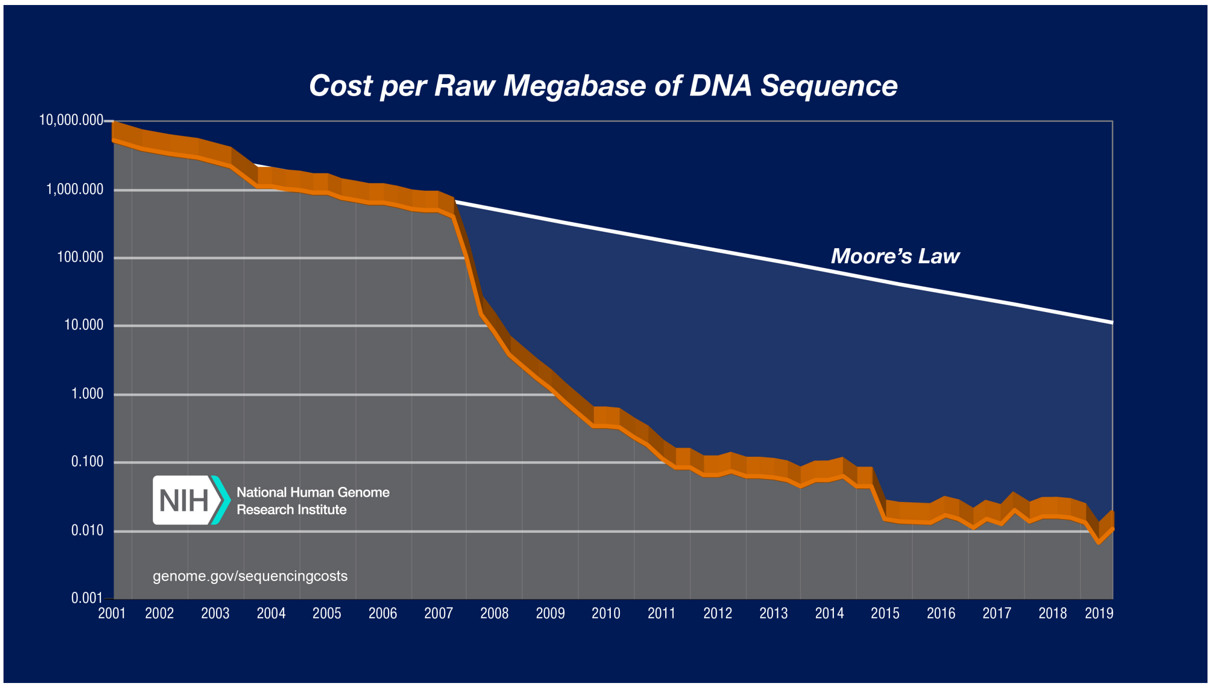


Figure 1.1: Cost per raw megabase of DNA sequence.

were commonly used in early assemblies. These methods usually perform a pairwise alignment on sequencing reads, and then arrange a layout of all reads. The consensus sequence from such a layout is outputted as assembly result. But these methods require a large memory as well as long runtime for high-throughput NGS sequencing data. To address this, de Bruijn graph was first introduced in EULER [Pevzner et al., 2001] and later adopted by Velvet [Zerbino and Birney, 2008] and Spades [Bankevich et al., 2012]. These de Bruijn graph-based methods use sliding window to decompose sequencing reads into substrings of length k (called k-mers). Then, k-mers are used to construct a de Bruijn graph where each node represents a distinct k-mer and each edge links two k-mers with an overlap of k-1 bases. Finally, an assembly is generated by Euler paths in subgraphs. Due to their low memory requirements, de Bruijn graph-based methods quickly became mainstream. Recently, some assemblers deploy even more elaborate algorithms to fit the characteristics of specific platforms; e.g. Canu [Koren et al., 2017] and miniasm [Li, 2016] were specially designed for long but noisy reads from Pacific Biosciences and Oxford Nanopore.

However, these assembly methods still often produce assemblies that are fragmented, incomplete, and even contain misassembled segments. This is mainly due to repeat regions in the genome. It is very difficult to uniquely determine the flanking sequence on either side of a repeat region when the repeat is too long to be spanned by a single read. Conservative assemblers leave these flanking sequences as separate contigs, resulting in fragmented assemblies. In contrast, aggressive assemblers try to resolve repeat regions based on subtle clues, at the cost of making misassemblies. Obviously, misassembly as well as poor connectivity hinders downstream analysis.

1.1.3 PDR

To evaluate assemblies from different assemblers, various metrics have been proposed for assessment of genome assembly. However, current popular genome assembly quality metrics evaluate contiguity, completeness, and correctness separately; it is not straightforward to get a correct picture of the overall quality of an assembly from these.

To address this problem, I consider the requirements of a good assembly: it should be highly similar to the real genome, especially in the aspect of genetic structure and property. Specifically, two genomes (or assemblies) are more similar to each other if the distance between any pair of positions in one genome is more similar to their distance in the other genome. This inspires me to propose a new metric PDR. It measures the quality of an assembly by the average ratio of the distance between any pair of positions in the reference genome to their distance in the draft assembly. It not only integrates contiguity, completeness, and correctness, but also makes good biological sense. Definitely, the computation over all pairs of positions in a large genome requires an efficient algorithm, otherwise the runtime is quadratic to the genome size. Thus, I also propose an approximation to compute PDR by integral. The error of this approximation is observed to be extremely small on a recent benchmark dataset. The result on this benchmark also shows that none of contiguity, completeness, or correctness has overwhelming impact on PDR. The PDR implementation based on integral is available at https://github.com/XLuyu/PDR.

1.1.4 CAST

With PDR as a reasonable metric, I have a deeper understanding about the limitation of tuning the aggressiveness of a genome assembler. In fact, there is a need of a better approach to further improve the quality of draft assembly, rather than to simply look for trade-off between contiguity and correctness by tuning the aggressiveness of a genome assembler.

In parent-progeny studies, as progeny sequence data are available, genetic map-anchoring approaches have an important budget advantage compared to other approaches (e.g. Hi-C experiments and optical maps) for improving genome assembly quality. However, when an appropriate genetic map is not available, the high time and memory demand of de novo genetic map construction severely hinders the application of genetic map-anchoring approaches on large genomes.

I recall the idea behind a genetic map, i.e. the law of genetic linkage: the closer two loci are in a genome, the more likely they are inherited together. Thus by abduction [1], given many progenies, if two positions in these progenies have highly correlated alleles, these two positions are likely close together in the parent genome. This inspires me that it is not necessary to explicitly construct a genetic map. Genetic information can be integrated into an assembly by checking whether its contigs follow the law of genetic linkage. In other words, when two contigs in a draft assembly have many positions with highly correlated alleles in many progenies, these two contigs can be scaffolded or even merged; conversely, when two halves of a contig in a draft assembly have many positions with uncorrelated alleles in many progenies, this contig can be split into two. Based on this idea, I propose CAST, a new correction and scaffolding tool for draft assembly. CAST uses progeny data but requires no genetic map. I have evaluated CAST on two datasets and have verified the correction and scaffolding made by CAST, using Hi-C data and synteny analysis against reference genomes. The results show that CAST improves the contiguity and correctness of draft genome assemblies. CAST is available at https://github.com/XLuyu/CAST.

1.2 Thesis organization

This thesis consists of 4 chapters. In Chapter 2, a new metric PDR is proposed to better assess assemblies. It is compared to existing metrics on a benchmark. Chapter 3 first provides a detailed review of several assembly-improving methods. Then, I propose a new method to correct and scaffold draft assembly, followed by its evaluation on two datasets. Finally, I conclude these works and propose some future work in chapter 4.

[1] A form of logical inference which starts with an observation or set of observations then seeks to find the simplest and most likely explanation for the observations

Chapter 2

PDR: a new genome assembly evaluation metric

2.1 Background

Over the past dozen years, NGS (Next-Generation Sequencing) has become a popular and powerful tool for genetic research. Its decreasing cost makes it affordable even for large genomes. Therefore, more and more species are sequenced and assembled. Meanwhile, assembly methods have also evolved along with sequencing technologies. Tens of assemblers have been developed with various strategies. Most of them are optimized for a specific sequencing technology or genome property. Thus, these assemblers outperform others on specific data for which they are optimized. Given the number of available assemblers, it has been a difficulty for people to select a suitable one. To provide some insight and guide such selection, many assembler benchmarks have been developed. Assemblathon [Earl et al., 2011, Bradnam et al., 2013] was first launched in the form of an assembly competition on given datasets. It compared assemblies from participating groups by tens of metrics. Salzberg et al. [2012] built up a benchmark GAGE on real datasets and tested 8 popular assemblers with a few metrics for multiple aspects. As genome assembly is shifting from short reads to long noisy reads in recent years, some comparison studies and benchmarks were proposed to assess assemblers for long noisy reads. In the same year, Šović et al. [2016] and Jayakumar and Sakakibara [2017] evaluated several assemblers for Pacbio SMRT data and Oxford Nanopore data, respectively. Beyond assemblers, assembly reconciliation tools were also compared and evaluated by Alhakami et al. [2017].

2.1.1 Contiguity

In principle, there is no standard metric set for genome assembly evaluation. Nevertheless, some metrics are commonly used in almost all of such works. A typical example is N50, which is defined as

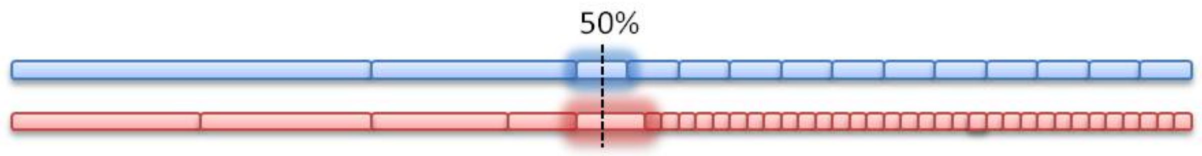


Figure 2.1: An example of NG50 (N50). Two assemblies are sorted and lined up. The 50th percentile of genome size (assembly size) is marked by black dot line. The NG50 (N50) length contigs are highlighted. The blue assembly is overall better than the red one, but its NG50 and N50 are lower.

the length of the longest scaffold (or contig) that at least 50% length of the assembly is contributed by scaffolds (or contigs) equal to or longer than it. This statistic has been widely used as well as criticized. An early challenge argued that the comparison could be unfair for assemblies with different sizes, especially when these assemblies come from an identical sample. NG50 was thus proposed, which is similar to N50 except the assembly size is replaced by an estimated genome size [Earl et al., 2011]. However, even NG50 could be misleading in some cases. Figure 2.1 shows an example where the blue assembly is overall better than the red one, but has lower NG50 and N50. QUAST [Gurevich et al., 2013] addressed this problem by offering NGx (Nx) plot to avoid domination of the 50th percentile.

However, this plot is not able to solve another problem in NG50: misassembly (a.k.a misjoin or structural error) resulting in larger contigs and thus inflated NG50. To fix this, when a reference genome is available, many benchmarks and evaluation tools used alignment block length to replace contig length in NG50 counting. These metrics include contig path NG50s [Earl et al., 2011], corrected Nx [Salzberg et al., 2012], normalized N50 [Mäkinen et al., 2012], and NGA50 [Gurevich et al., 2013]. Although they slightly differ in definition and implementation, their principle is identical: break erroneous contigs at misassembled points. This series of metrics are usually used in assessment of contiguity.

Besides these, there are some other contiguity metrics, despite they were only used in limited studies. For example, E-size was proposed in GAGE and defined as the expected contig length at which a random position locates. Similar to N50, this metric only reflects the distribution of contig lengths and can be inflated by misassemblies. With a reference genome, another metric U50 [Castro and Ng, 2017] identified unique, target-specific contigs by removing overlapping sequence, aiming at circumventing some limitations that are inherent to the N50 metric. In Assemblathon [Earl et al., 2011, Bradnam et al., 2013], CC50 was defined as a distinct idea. A pair of positions in a chromosome is correctly contiguous (CC) if they are identified in the same contig as well as right order. The distances between such 2 positions in the reference genome are gathered, and CC50 is the 50-th percentile of them. As its computation is time-consuming, it was only estimated by a sampling-based method.

2.1.2 Completeness

Completeness is another dimension in genome assembly evaluation. In some benchmarks, only contiguity was evaluated and completeness was considered as a part of it. In fact, completeness focuses more on the loss caused by an assembly, while contiguity reflects the reconstruction of local context. A universal metric of completeness is alignment coverage. It is very straightforward but sometimes poorly discriminative: not only good assemblies but also raw reads can achieve an alignment coverage above 90%. When a reference genome is not available, completeness can be assessed by CEGMA [Parra et al., 2007] and BUSCO [Simão et al., 2015]. They collect a set of conserved single-copy orthologs and test whether an assembly contains them. In other words, what they reflect is the completeness of gene space, not exact completeness of the assembled genome. So, they are actually sampling tests.

2.1.3 Correctness

One more aspect in genome assembly evaluation is correctness. The commonly used metrics are more straightforward than those for contiguity and less improved over time. These metrics usually include single base errors (mismatch), indels, and misassemblies. Misassembly is regarded as the most harmful type and widely used in metrics for assessment of correctness. To better profile this type, in QUAST, it is further categorized into: (a) relocation, a position whose flanking sequences are aligned to the same chromosome but away from each other; (b) inversion, a position whose flanking sequences are aligned to the same chromosome but on opposite strands; (c) translocation, a position whose flanking sequences are aligned to different chromosomes. Though these are enough to assess correctness, they are limited by the need of a reference genome, which is usually not available in *de novo* assembly. Moreover, these counts on the number of misassemblies do not reflect the size and effect of misassemblies.

Thus, REAPR [Hunt et al., 2013] and LAP [Ghodsi et al., 2013] were proposed to evaluate an assembly by checking the consistency between the assembly and the sample's sequencing reads in the absence of a reference genome. REAPR requires mapped pair-end reads with long insert size ($>$1000 bp) and utilized their pairing and mapping information as assessment evidence. Therefore, its performance is subject to the quality and insert size of reads. LAP defines quality as the conditional probability of observing the sequenced reads from the assembled sequence. But it is only applicable to comparing assemblies derived from the same read set. Similar to REAPR and LAP, FRCurve [Narzisi and Mishra, 2011] utilizes read layout information to detect misassembly features. Then, it plots a Feature-Response curve which shows the maximal total length of contigs (Y-axis) within a given number of misassembly feature count (X-axis). In fact, FRCurve not only evaluates correctness, but also contiguity. To give a quantitative metric, the plot is used to compute corrected N50. Unlike NGA50, the corrected N50 does not completely avoid the drawbacks of N50. A few misassemblies in large contigs can still heavily inflate corrected N50. Another problem of FRCurve is that all error features (e.g. alignment breakpoints, low depth, and abnormal read orientation) are equally weighted regardless of their effect in downstream analysis.

2.1.4 Need for an overall metric

There is a trade-off between contiguity and correctness. Specifically, some aggressive assemblies are better in contiguity at the cost of correctness, while others yield shorter contigs with fewer errors. Divergence mostly happens during contig extension at repeat regions. When a repeat region is too long to be spanned by a single read, assemblers have multiple choices to extend a contig but do not know which is the right one. Conservative assemblers stop extension when they lack enough evidence to make a decision, resulting in fragmented assemblies. In contrast, aggressive assemblers continue to extend the sequence along the way with subtle clues, at the risk of misassembly. Different assemblers have their own strategies to determine whether to continue extension; and if to continue, which choice is better.

In benchmarks and evaluation tools, each of contiguity, completeness, and correctness(C^3) is usually investigated by a few metrics to provide a multidimensional comparison and global profile of assemblies.

However, as criticized by Haiminen et al. [2011], providing tables full of different assembly metrics complicates the comparison between assemblies as each metric has its own front runner. Such overly detailed information does not really guide practice. It is still hard to tell which assembly is overall better. To address this, it is desirable to have an overall metric or score which integrates all three aspects, viz. contiguity, completeness, and correctness. Thus, I propose a new metric for genome assembly assessment to fill this gap. Pairwise Distance Reconstruction(PDR) integrates C^3 into one value by reasonable weights derived from a basic concern in genetic studies.

2.2 Methods

2.2.1 Definition

PDR is designed to answer the question: How accurately the distance of two positions on the genome can be obtained from the assembly? First, a position in a genome can be denoted by a coordinate (c, p) where c is the chromosome and p is the index on the chromosome (hereafter, chromosome and contig refer to sequences in the reference genome and the assembly respectively). Then, the distance between two positions $A = (c_A, p_A)$ and $B = (c_B, p_B)$ is defined as:

$$|AB| = \begin{cases} |p_B - p_A| & c_A = c_B \\ +\infty & c_A \neq c_B \end{cases} \tag{2.1}$$

Similarly, a position in an assembly is also denoted as a coordinate. For example, A and B in the reference genome map to $A' = (c_{A'}, p_{A'})$ and $B' = (c_{B'}, p_{B'})$ in the assembly, respectively. A quality score $S(A, B) \in [0, 1]$ can be defined to quantify the relative fold change of the distance between the reference genome and the assembly:

$$\begin{aligned} S(A,B) &= \begin{cases} \frac{|AB|}{|A'B'|} & |AB| \leq |A'B'| \\ \frac{|A'B'|}{|AB|} & |AB| > |A'B'| \end{cases} \\ &= \frac{min(|AB|, |A'B'|)}{max(|AB|, |A'B'|)} \end{aligned} \tag{2.2}$$

Note the convention that $\frac{0}{+\infty} = 0$ and $\frac{+\infty}{+\infty} = 1$ in Equation 2.2 for practical reason. Figure 2.2 shows

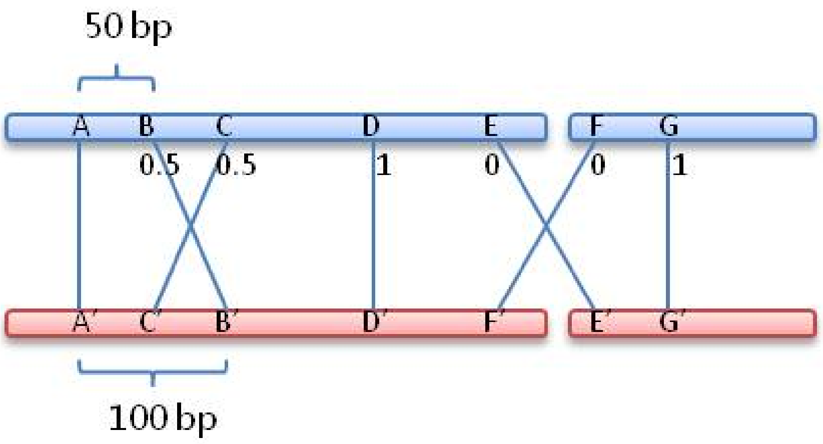


Figure 2.2: Quality score examples. The blue and red bars are reference chromosomes and assembly contigs, respectively. The lines between them are mappings for positions labeled by capitals. The score for each position to A is marked under blue bar, e.g. $S(A, B) = 0.5$, $S(A, D) = 1$.

a few examples of quality score. Definitely, if two positions locate on the same chromosome but map to different contigs, then the quality score is 0 as the assembly provides incorrect information, and vice versa (AE and AF in Figure 2.2). In contrast, if the distances in the reference genome and assembly are identical like AD or AG in Figure 2.2, the quality score is 1 as the information is perfectly reconstructed. Otherwise, if one of $|AB|$ and $|A'B'|$ is half of the other one (AB and AC in Figure 2.2), the quality score is 0.5 as downstream genetic analysis, e.g. genetic distance, will be biased by a factor of 0.5. Thus, this quality score reflects the accuracy of the information for a pair of positions in the assembly. Finally, the PDR of an assembly $\mathbb{A}$ can be simply defined as the average quality score of all pairs in the reference genome $\mathbb{G}$:

$$PDR(\mathbb{A}) = \frac{1}{|\mathbb{G}|^2} \sum_{A,B \in \mathbb{G}} S(A, B) \tag{2.3}$$

For easier reading and interpretation, PDR is preferred to be expressed as a percentage because $0 \leq PDR \leq 1$ always holds.

Note that if I define $S(A, B) = 1$ for all pairs located on the same chromosome and same contig, regardless of distance, then Equation 2.2 degenerates to:

$$S_d(A,B) = \begin{cases} 1 & c_A = c_B \textit{ and } c_{A'} = c_{B'} \\ 0 & c_A \neq c_B \textit{ and } c_{A'} = c_{B'} \\ 0 & c_A = c_B \textit{ and } c_{A'} \neq c_{B'} \\ 1 & c_A \neq c_B \textit{ and } c_{A'} \neq c_{B'} \end{cases} \tag{2.4}$$

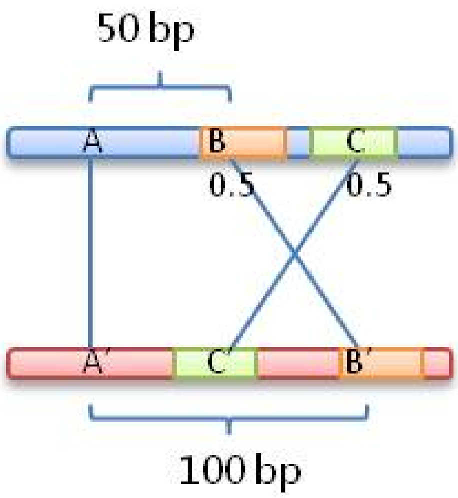
And the corresponding $PDR_d(\mathbb{A}) = \frac{1}{|\mathbb{G}|^2} \sum_{A,B \in \mathbb{G}} S_d(A,B)$ is exactly the probability that the assembly can successfully answer whether two randomly specified reference genome positions A and B locate on the same chromosome. This is a very basic question in genetic linkage studies. In further studies like genetic distance estimation, the distance on the same chromosome between A and B matters. And this is the reason I extend Equation 2.4 to Equation 2.2.

2.2.2 Relation to C^3

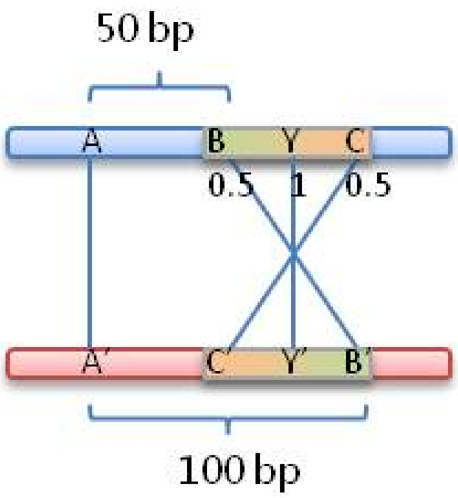
In Equation 2.3, it is assumed that every position in the reference genome can be mapped to the assembly. However, there is no such guarantee in practice. In fact, an assembly usually has an alignment coverage, denoted as α. Any pair involving at least one uncovered position has score 0. Thus by Equation 2.3, $PDR(\mathbb{A}) \leq \alpha^2$. In this way, completeness is integrated in PDR as the upper bound.

Contiguity is also a part of what PDR evaluates. In a fragmented assembly, widespread separated pairs (e.g. AE in Figure 2.2) score 0 and lower the PDR. In contrast, increase of contiguity essentially converts separated pairs to linked pairs (i.e. on the same chromosome like AB, AC, AD) and thus increases PDR.

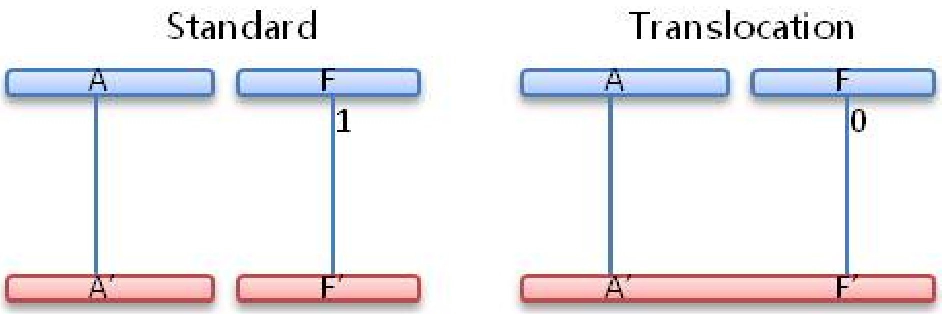
To analyze the effect of correctness, misassemblies including relocation, inversion, and translocation are considered. In Figure 2.3a, AB and AC are two typical examples of relocation. Compared to perfect pairs, such relocation pairs are penalized by a factor of fold change of the sequence length in the middle. Second, consider another simple case that B and C are boundary of an inversion, as shown in Figure 2.3b. In this case, like relocation, AB and AC are penalized. Note that the position Y at the exact middle of B and C has a score 1 for AY, because this position keeps the



(a) Relocation example. The green block is relocated at the middle between A and B due to misassembly. Not only the distance between A and C changes, but the distance between A and B also increases. $S(A, B)$ and $S(A, C)$ are therefore only 0.5.



(b) Inversion example. The block between B and C is inverse due to misassembly. $S(A, Y) = 1$ because Y is the center of inversion. All other positions between B and C are penalized by $0 \sim 0.5$, depending on their distance changes. (e.g. $S(A, B) = 0.5$ and $S(A, C) = 0.5$)



(c) Translocation example. A perfect assembly is shown on the left side, where $S(A, F) = 1$. Another assembly with translocation is shown on the right side. $S(A, F)$ reduces to 0 as A' and F' are incorrectly placed on the same contig in this assembly.

Figure 2.3: Examples of correctness. The blue and red bars are reference chromosomes and assembly contigs, respectively. The lines between them are mappings for positions labeled by capitals. The score for each position to A is marked under blue bar, e.g. $S(A, B) = 0.5$, $S(A, Y) = 1$

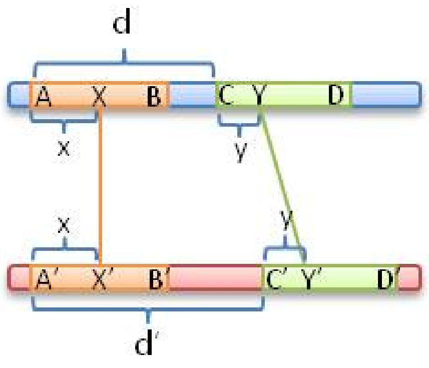
right distance to A. Thus, the penalty for inversion is similar to that of relocation but each position's penalty depends on its distance to the center of inversion. For translocation, Figure 2.3c shows a simple comparison. In the fragmented but correct assembly, $S(A, F) = 1$ as the information is consistent to the reference genome. However, in the other assembly with translocation, $S(A, F) = 0$ because the assembly draws a misleading conclusion about A and F. In fact, all pairs bridging the translocation point have a score of 0. This is an essential difference between PDR and other contiguity-only metrics like X50 series. Even in NGA50, misassemblies are just simply split and no penalty is applied on such misleading information in the assembly. Ignoring the risk of misassemblies, benchmarks and comparisons become unfair for conservative assemblers. PDR addresses this problem by integrating contiguity and correctness with reasonable implicit weights in the context of genetic studies.

2.2.3 Computation

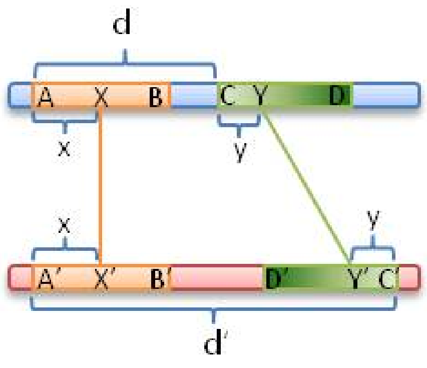
As PDR needs to compute the average of quality scores for all pairs of positions in a reference genome of size $|G|$, its computational complexity is $O(|G|^2)$. This complexity is impractically high for a large genome. Thus, I approximate it by an integral of mapping segment pairs. (The error is extremely small, as verified in Section 2.3.)

First, each chromosome in the reference genome is split into non-overlapping blocks by a fixed length l, except the last block which is allowed 50% more or less than l. Then, these blocks are mapped to the assembly by an aligner like BWA [Li and Durbin, 2009] or Minimap2 [Li, 2016]. Contiguous blocks are merged if they are mapped to contiguous positions in the assembly. The merged blocks are called mapping segments, denoted as the set $\mathbb{M}$. Compared to $|G|$, $|\mathbb{M}|$ is much smaller. Then, pairwise computation of all mapping segments is performed where each pair of mapping segments can be computed in constant time. For simplicity, the multi-mapping case is postponed.

First consider a simple example shown in Fig 2.4a, where two segments AB and CD only map to



(a) $I = 1$ when $C'D'$ and $A'B'$ are mapped on the same strand.



(b) $I = -1$ when $C'D'$ and $A'B'$ are mapped on different strands.

Figure 2.4: Mapping segments in pairwise computation. In each subfigure, two segments AB and CD map to $A'B'$ and $C'D'$, respectively. X and Y are enumerated on AB and CD, respectively.

$A'B'$ and $C'D'$. For any position X on AB, and Y on CD:

$$S(X,Y) = \frac{min(|XY|, |X'Y'|)}{max(|XY|, |X'Y'|)} = \frac{min(|d+y-x|, |d'+y-x|)}{max(|d+y-x|, |d'+y-x|)} \tag{2.5}$$

where $d = |AC|$, $d' = |A'C'|$, x and y are offsets from the start of segments. Then, the sum of all such XY pairs is the quality score for AB and CD:

$$\begin{aligned} S(AB,CD) &= \sum_{y=0}^{|CD|} \sum_{x=0}^{|AB|} \frac{min(|d+y-x|, |d'+y-x|)}{max(|d+y-x|, |d'+y-x|)} \\ &\approx \int_{y=0}^{l_y} \int_{x=0}^{l_x} \frac{min(|d+y-x|, |d'+y-x|)}{max(|d+y-x|, |d'+y-x|)} dxdy \end{aligned} \tag{2.6}$$

where $l_x = |AB|+1-\epsilon$, $l_y = |CD|+1$. During computation, the mapping segments are enumerated by reference coordinate order. Thus, without loss of generality, it can be assumed that $A < B < C < D$ (hereinafter, position variables on the same chromosome/contig are comparable by coordinate). Also, an additional infinitesimal ϵ in l_x avoids $d + y - x = 0$ with a negligible effect on the integral result. Based on these, $d + y - x > 0$ are guaranteed and the absolute-value function for it can be removed. Also, AB and CD are mapped on the same strand in this example. But this is not guaranteed in all mappings, e.g. inversion in Fig 2.4b. To simplify the computation, the strand on which AB maps is defined as the plus strand. By this way, $A' < B'$ is fixed. To indicate the strand CD maps on, an indicator I is introduced to Equation 2.6:

$$S(AB,CD) \approx \int_{y=0}^{l_y} \int_{x=0}^{l_x} \frac{min(d+y-x, |d'+Iy-x|)}{max(d+y-x, |d'+Iy-x|)} dxdy \tag{2.7}$$

where $I = 1$ when CD maps on the same strand as AB (Fig 2.4a), otherwise $I = -1$ (Fig 2.4b). In practice, a reference block can map to multiple positions in the assembly. This is mainly due to the repeat regions in the genome. A reasonable solution is to try every mapping position of each segment in the computation of Equation 2.7, and take the maximal quality score (In fact, it does not matter as discussed later in Section 2.3.4). Finally, PDR is approximated to:

$$PDR(\mathbb{A}) \approx \frac{1}{|\mathbb{G}|^2} \sum_{AB,CD \in \mathbb{M}} S(AB,CD) \tag{2.8}$$

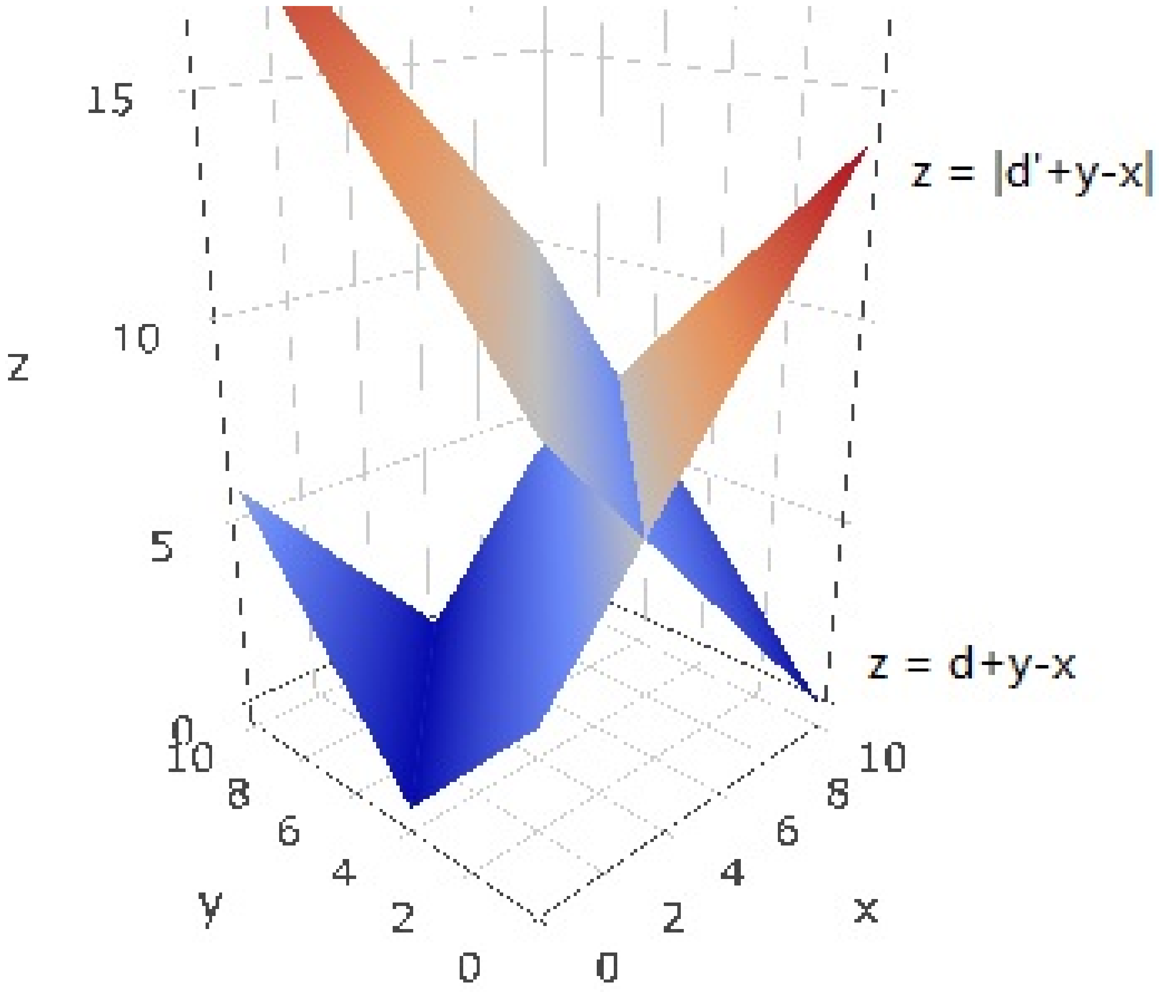
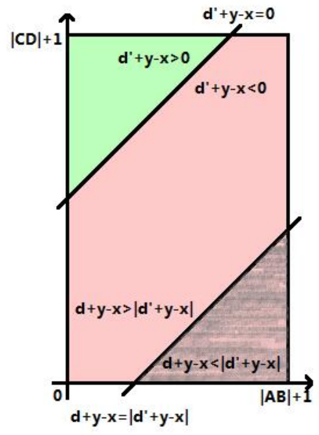


Figure 2.5: The surfaces of $d + y - x$ and $|d' + Iy - x|$ when $I = 1$, $d = 10$, $d' = -4$.

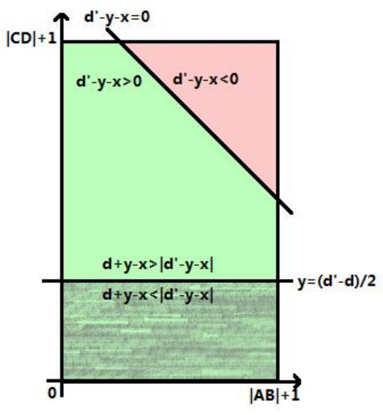
2.2.4 Integral

In Equation 2.7, the integrand contains absolute value, min, and max. Therefore, it is solved as a piecewise function. For a given y, the sub-function changes at $|d' + Iy - x| = 0$ and $d + y - x = |d' + Iy - x|$. To illustrate this, Fig 2.5 plots the surfaces of $10 + y - x$ and $| - 4 + y - x|$. The first boundary is always a line $d' + Iy - x = 0$, while the second boundary depends on the value of I.

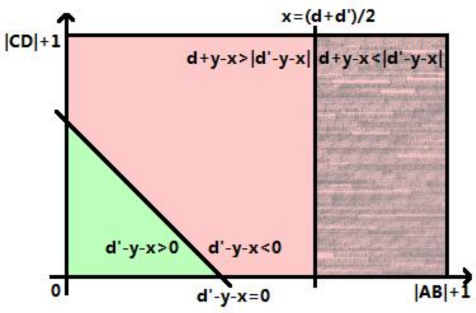
When $I = 1$, $d + y - x = |d' + Iy - x|$ has two solutions. For $d' + y - x \geq 0$, the solution is $d = d'$ and thus sub-function has no change. For $d' + y - x < 0$, the solution is $x = (d + d')/2 + y$. The



(a) $I = 1$



(b) $I = -1$ and $d < d'$



(c) $I = -1$ and $d \geq d'$

Figure 2.6: Illustration of sub-domain in piecewise integral. The sub-domains satisfying $d' + Iy - x < 0$ are green, otherwise red. Shadow masks sub-domains satisfying $|d' + Iy - x| > d + y - x$.

integral sub-domains are illustrated by Fig 2.6a and the Equation 2.7 is extended to:

$$
\begin{aligned}
S(AB, CD) \approx & \int_{y=-d'}^{l_y} \int_{x=0}^{d'+y} \frac{d'+y-x}{d+y-x} dxdy \\
& + \int_{y=-(d+d')/2}^{l_x-d'} \int_{x=d'+y}^{(d+d')/2+y} \frac{-(d'+y-x)}{d+y-x} dxdy \\
& + \int_{y=0}^{l_x-(d+d')/2} \int_{x=(d+d')/2+y}^{l_x} \frac{d+y-x}{-(d'+y-x)} dxdy
\end{aligned} \tag{2.9}
$$

When $I = -1$, the two solutions of $d + y - x = |d' + Iy - x|$ have different forms. For $d' - y - x \geq 0$, the solution is $y = (d' - d)/2$. For $d' - y - x < 0$, the solution is $x = (d + d')/2$. Note that $0 < (d' - d)/2$ and $(d + d')/2 < l_x$ are never satisfied simultaneously: If $d > d'$ then $0 > (d' - d)/2$; otherwise $d \leq d' \Rightarrow (d + d')/2 \geq d = |AC| \geq l_x$. This leads to different sub-domain layouts, as shown in Fig 2.6b and Fig 2.6c. The Equation 2.7 is thus extended to:

$$S(AB,CD) \approx \begin{cases} \int_{y=d'-l_x}^{l_y} \int_{x=d'-y}^{l_x} \frac{-(d'-y-x)}{d+y-x} dxdy \\ + \int_{y=(d'-d)/2}^{d'} \int_{x=0}^{d'-y} \frac{d'-y-x}{d+y-x} dxdy & d < d' \\ + \int_{y=0}^{(d'-d)/2} \int_{x=0}^{l_x} \frac{d+y-x}{d'-y-x} dxdy \\ \\ \int_{y=d'-l_x}^{l_y} \int_{x=d'-y}^{(d+d')/2} \frac{-(d'-y-x)}{d+y-x} dxdy \\ + \int_{y=0}^{d'} \int_{x=0}^{d'-y} \frac{d'-y-x}{d+y-x} dxdy & d \geq d' \\ + \int_{y=0}^{l_y} \int_{x=(d+d')/2}^{l_x} \frac{d+y-x}{-(d'-y-x)} dxdy \end{cases} \tag{2.10}$$

During integral computation, regardless of the value of I, all sub-domains need to be pruned by intersecting with $x \in [0, l_x]$ and $y \in [0, l_y]$. This may result in empty sub-domains as well as further sub-domains from trapezoidal sub-domains. Finally, the integral expression of all sub-domains are generalized; each sub-domain can be transformed to one of the following four generalized forms:

1. $\pm \int_w^v \int_q^p \frac{a+Ly-x}{b+Ky-x} dxdy$
2. $\pm \int_w^v \int_q^{p+y} \frac{a+y-x}{b+y-x} dxdy$
3. $\pm \int_w^v \int_q^{p-y} \frac{a-y-x}{b+y-x} dxdy$
4. $\pm \int_w^v \int_{q+y}^{p+y} \frac{a+y-x}{b+y-x} dxdy$

where $L, K \in \{-1, 1\}$. The leading $\pm$ is jointly determined by the sign of $d' + Iy - x$ and whether x's upper limit and lower limit are swapped during transformation. For simplicity, they are solved without considering $\pm$.

$$
\begin{aligned}
&\int_w^v \int_q^p \frac{a+Ly-x}{b+Ky-x}dxdy \\
=&\int_w^v (x-(Ly-Ky+a-b)\log(Ky+b-x))|_q^p dy \\
=&\frac{1}{2}\Big((p-q)(v-w)(2-K(K-L)) \\
&+v(Kv-Lv-2a+2b)(\log(-Kv-b+p)-\log(-Kv-b+q)) \\
&-w(Kw-Lw-2a+2b)(\log(-Kw-b+p)-\log(-Kw-b+q)) \\
&+(b-p)(-2Ka+Kb+Kp+Lb-Lp)(\log(2K(Kv+b-p))-\log(2K(Kw+b-p))) \\
&-(b-q)(-2Ka+Kb+Kq+Lb-Lq)(\log(2K(Kv+b-q))-\log(2K(Kw+b-q)))\Big)
\end{aligned} \tag{2.11}
$$

$$
\begin{aligned}
&\int_w^v \int_q^{p+y} \frac{a+y-x}{b+y-x}dxdy \\
=&\int_w^v (x-(a-b)\log(b-x+y))|_q^{p+y} dy \\
=&\frac{v^2}{2}-\frac{w^2}{2}-(w-v)(p-q) \\
&+(a-b)(b-q)(\log(b-q+v)-\log(b-q+w)) \\
&+(a-b)(v\log(-b+q-v)-w\log(-b+q-w)) \\
&+(w-v)(a-b)(\log(-b+p)+1)
\end{aligned} \tag{2.12}
$$

$$
\begin{aligned}
&\int_w^v \int_q^{p-y} \frac{a-y-x}{b+y-x}dxdy \\
=&\int_w^v (x+(-a+b+2y)\log(-b+x-y))|_q^{p-y} dy \\
=&\frac{(w-v)(w+v+b-p)}{2} \\
&+v(-a+b+v)(\log(-b+p-2v)-\log(-b+q-v)) \\
&-w(-a+b+w)(\log(-b+p-2w)-\log(-b+q-w)) \\
&+(a-q)(b-q)(\log(b-q+v)-\log(b-q+w)) \\
&+\frac{1}{4}(b-p)(-2a+b+p)(\log(2b-2p+4v)-\log(2b-2p+4w))
\end{aligned} \tag{2.13}
$$

$$\begin{aligned}
&\int_w^v \int_{q+y}^{p+y} \frac{a+y-x}{b+y-x} dxdy \\
=&\int_w^v (x-(a-b)\log(b-x+y))|_{q+y}^{p+y} dy \\
=&(v-w)(p-q-(a-b)\log(-b+p)+(a-b)\log(-b+q))
\end{aligned} \tag{2.14}$$

2.3 Experiment

2.3.1 Dataset

To profile and further investigate PDR's properties, three datasets from the QUAST-LG benchmark (available on http://cab.spbu.ru/software/quast-lg/) were selected to compute PDR for all assemblies in each dataset. The basic information are summarized in Table 2.1. As shown in the table, these datasets have different genome sizes and were assembled by multiple assemblers. So, they have enough diversity to test PDR's robustness. In each dataset, besides assemblies from assemblers, Mikheenko et al. [2018] also proposed a "UpperBound" assembly inferred from the given sequencing data and a reference genome to indicate the theoretically best assembly.

PDR was then compared with metrics in the original QUAST reports. From the original reports, genome fraction (a.k.a alignment coverage) was selected as key metrics for completeness, while contiguity is reflected by NG50 and NGA50. In terms of correctness, misassembly count is a common representative. Meanwhile, for the yeast dataset and the worm dataset, an accurate PDR defined by Equation 2.3 was also computed by brute force to verify the error introduced by the approximation in

Table 2.1: The basic information of datasets.

Dataset	Genome Size (bp)	Sequencing Platform	Assemblers
S. cerevisiae (Yeast)	12.1M	Illumina pair-ends and PacBio SMRT	UpperBound, Canu, FALCON, Flye, MaSuRCA, Miniasm
C. elegans (Worm)	100.3M	Illumina pair-ends and PacBio SMRT	UpperBound, Canu, FALCON, Flye, MaSuRCA, Miniasm
Human HG001	3.1G	Illumina pair-end and Oxford Nanopore	UpperBound, Canu, Flye

Equation 2.8.

The PDR computation for all assemblies was carried out on a computing node with two six-core Intel Xeon E5-2620 v3 2.4GHz CPUs and 64G random-access memory. Whereas, the brute force for the accurate PDR was executed on a distributed system comprising of 36 nodes.

2.3.2 Aligner and block size

Before considering other aspects, aligner and block size l were first investigated to find the best settings in practice. The yeast dataset includes 6 assemblies and was used to compute PDR based on BWA and Minimap2, varying the block size from 30 to 100 bases. Figure 2.7 shows the result. Although BWA converged faster than Minimap2 on all assemblies, Minimap2 also converged with block size increase. After convergence, the differences between assemblies were much larger than those between aligners. Thus, aligner is not a key factor in PDR computation, as long as the block size is large enough.

As BWA had better accuracy and converges earlier then Minimap2, the following evaluations were

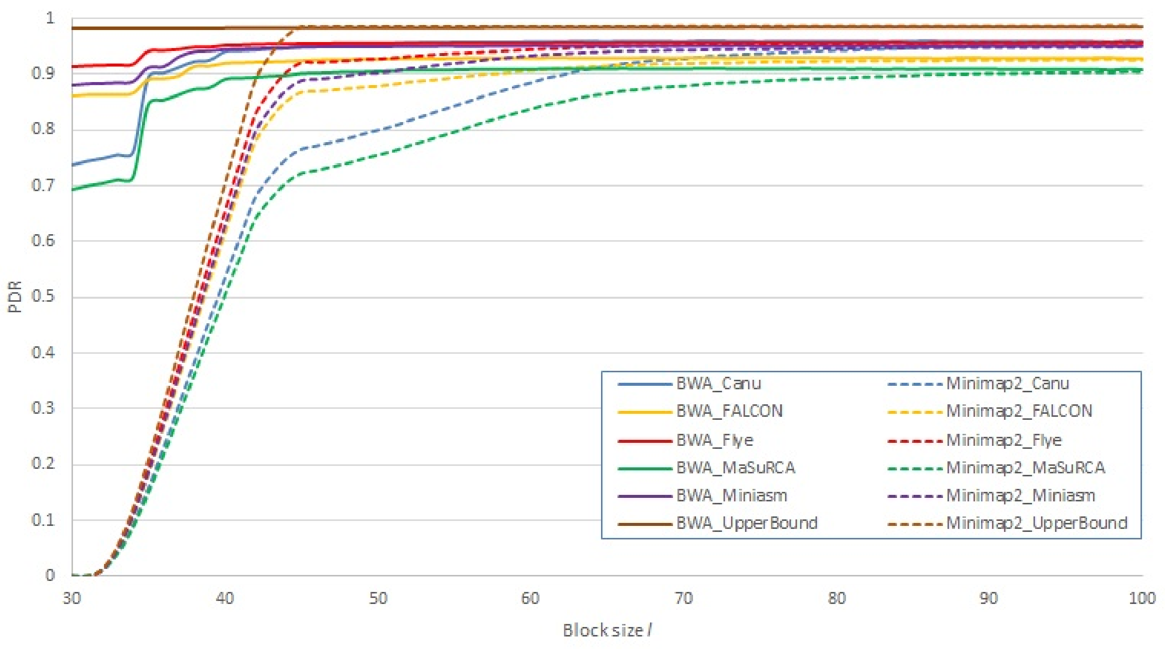


Figure 2.7: PDR on different block sizes and aligners. All 6 assemblies (distinguished by color here) from the yeast dataset were used to compute PDR. During PDR computation, BWA (solid lines) and Minimap2 (dashed lines) were used as aligner. Block size was tuned along X-axis.

carried out based on BWA. And the block size l was set to 1000 for the sake of runtime.

2.3.3 Approximation

Since an approximation is used for fast computation of PDR, its acceleration effect and introduced error need to be measured. Table 2.2 and Table 2.3 show a comparison between approximated PDR and accurate PDR on the yeast dataset and the worm dataset. The human dataset was not used because the brute force computation of PDR could not finish within one day.

According to the |PDR-aPDR| row in these tables, it is obvious that the approximation in Equation 2.8 had an extremely small error, which is below 1E-9. In contrast, the runtime differences were very large. The computation for PDR for each assembly in the yeast dataset and the worm dataset finished

Table 2.2: Approximation and acceleration effect on the yeast dataset. The "PDR" in this table was computed by the integral approximation on a single computing node, while the "aPDR" is the accurate PDR computed by brute force on 36 computing nodes. (Aligner's runtime was not included)

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
PDR	98.74%	93.64%	91.44%	94.49%	88.49%	93.27%
aPDR	98.74%	93.64%	91.44%	94.49%	88.49%	93.27%
|PDR-aPDR|	6.5937E-11	2.3E-11	1.98E-10	2.9E-11	1.437E-10	2.5E-11
PDR runtime	1s	1s	1s	1s	1s	1s
aPDR runtime	201s	185s	178s	172s	90s	172s

Table 2.3: Approximation and acceleration effect on the worm dataset. The "PDR" in this table was computed by the integral approximation on a single computing node, while the "aPDR" is the accurate PDR computed by brute force on 36 computing nodes. (Aligner's runtime was not included)

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
PDR	87.81%	85.15%	82.23%	84.33%	82.72%	83.46%
aPDR	87.81%	85.15%	82.23%	84.33%	82.72%	83.46%
|PDR-aPDR|	8.44802E-12	3.6E-12	2.69E-11	2.3E-11	4.396E-12	1.6E-11
PDR runtime	1s	1s	1s	1s	1s	1s
aPDR runtime	9916s	7048s	4517	6010s	2632s	4012s

within 1 second. In fact, for a human assembly, it was able to finish within 10 minutes. However, even on a distributed platform with 36 nodes, thousands of seconds were required to compute accurate PDR for any assembly in the worm dataset. It would cost more than one day if this was run on a single node like approximated PDR, and even much longer for the human dataset. This verified that the approximation accelerates the PDR computation with negligible accuracy loss.

2.3.4 Repeat handling

As described in Section 2.2, when facing a multi-map segment caused by a repeat region, my PDR implementation tries every mapping position of each segment and intuitively takes the maximal quality score. To investigate the effect of this multi-map resolution, a revised PDR value, which takes minimal quality score during multi-map resolving, was computed for all three datasets.

As shown in Table 2.4, Table 2.5, and Table 2.6, the percentages of multi-map blocks are all below 5%

Table 2.4: PDR values in the yeast dataset under different repeat resolutions. The "Multi-map block" indicates the percentage of blocks mapping to multiple positions in an assembly. The "PDR" was computed by taking maximal quality score for multi-map segments. In contrast, the "PDRmin" took minimal quality score.

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
Multi-map block	0.10%	1.23%	0.81%	0.83%	1.24%	0.86%
PDR	98.74%	93.64%	91.44%	94.49%	88.49%	93.27%
PDRmin	98.74%	93.64%	91.44%	94.49%	88.49%	93.27%
|PDR-PDRmin|	0	0	0	0	0	0

Table 2.5: PDR values in the worm dataset under different repeat resolutions. The "Multi-map block" indicates the percentage of blocks mapping to multiple positions in an assembly. The "PDR" was computed by taking maximal quality score for multi-map segments. In contrast, the "PDRmin" took minimal quality score.

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
Multi-map block	0.06%	0.28%	0.45%	0.45%	0.35%	0.29%
PDR	87.81%	85.15%	82.23%	84.33%	82.72%	83.46%
PDRmin	87.81%	85.15%	82.23%	84.33%	82.72%	83.46%
|PDR-PDRmin|	0	0	0	0	0	0

Table 2.6: PDR values in the human dataset under different repeat resolutions. The "PDR" was computed by taking maximal quality score for multi-map segments. In contrast, the "PDRmin" took minimal quality score.

Metrics	UpperBound	Canu	Flye
Multi-map block	0.46%	3.09%	1.20%
PDR	94.24%	82.73%	79.64%
PDRmin	94.24%	82.73%	79.64%
|PDR-PDRmin|	0	2.6E-14	1.1E-14

because the block size is long enough to span most repeat regions. More importantly, no difference between two resolutions (PDR and PRDmin in the tables) were found in the yeast dataset and the worm dataset; only negligible difference was observed in the human dataset. So, in fact, it does not matter whether we take max or min to resolve multi-map segments.

2.3.5 Correlation to C^3

Table 2.7 and Table 2.8 show the PDR results on the yeast dataset and the worm dataset respectively, accompanied by selected metrics from the original reports.

On both datasets, the traditional metrics did not totally agree to each other. However, they were consistent across datasets in some degree. Therefore, the character of each assembler can be inferred from these metrics. For example, FALCON introduced the least number of misassemblies, but obtained the lowest genome fractions. So it is a typical conservative assembler. It keeps contigs

Table 2.7: Metrics on yeast dataset. For each metric, the best values are colored blue while the worst values are colored red.

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
Genome fraction	99.92%	98.77%	96.07%	98.04%	97.41%	97.31%
# misassemblies	0	35	19	24	60	35
NG50	776,910	776,810	762,979	776,728	432,306	737,373
NGA50	776,910	668,909	694,355	676,772	345,836	663,236
PDR	98.74%	93.64%	91.44%	94.49%	88.49%	93.27%

Table 2.8: Metrics on worm dataset. For each metric, the best values are colored blue while the worst values are colored red.

Metrics	UpperBound	Canu	FALCON	Flye	MaSuRCA	Miniasm
Genome fraction	99.95%	99.54%	98.67%	99.31%	99.18%	99.41%
# misassemblies	0	147	94	122	138	262
NG50	3,507,402	3,634,244	2,013,998	2,321,891	1,435,395	2,105,818
NGA50	3,507,402	1,292,248	1,176,205	1,305,538	1,016,420	1,214,817
PDR	87.81%	85.15%	82.23%	84.33%	82.72%	83.46%

separated when the repeats are too similar to each other. In contrast, Canu could be a representative of aggressive assemblers. It achieved good completeness and contiguity at the cost of correctness. Without an integrated metric, it may be hard to choose between a cleaner assembly and a more complete assembly. PDR makes them comparable.

In the yeast dataset, PDR indicates that a good balance between completeness and correctness was achieved by Flye, where contiguity was also served. This also proves that neither completeness nor correctness is able to dominate PDR. In terms of contiguity, Canu has higher NG50 while FALCON has higher NGA50. This is because NGA50 partially eliminates the influence from misassembly. Although Canu and FALCON beat Flye in NG50 and NGA50 respectively, Flye obtained higher PDR than these two. Thus, contiguity is also not a dominating factor. But these two observations do not imply they are unconsidered. For example, MaSuRCA reasonably got the lowest PDR because it underperformed others in term of correctness and contiguity, despite its moderate completeness. In the worm dataset, FALCON had the lowest PDR because of poor completeness and unsatisfying contiguity, though it consistently made the fewest misassemblies.

To quantitatively investigate the correlation between PDR and other metrics, the Pearson Correlation Coefficient was calculated for each pair of metrics within these two datasets. Table 2.9 and Table 2.10 show the results in matrix form.

On the yeast dataset, it can be observed that PDR always kept good correlation (>0.75) with all other metrics. Given that genome fraction was poorly correlated to NG50, NGA50, and the number of misassemblies, PDR played a role as a bridge between completeness, contiguity and correctness.

Table 2.9: Pearson correlation between PDR and other metrics on yeast dataset. Cells are colored by their value. Dark red indicates high correlation. "# misassembly" is negated as less misassembly is preferred.

	Genome fraction	PDR	# misassemblies	NG50	NGA50
Genome fraction	1	0.764352278	0.411333157	0.247482277	0.342096991
PDR	0.764352278	1	0.860966054	0.730571659	0.834831891
# misassemblies	0.411333157	0.860966054	1	0.791050105	0.907051293
NG50	0.247482277	0.730571659	0.791050105	1	0.966809906
NGA50	0.342096991	0.834831891	0.907051293	0.966809906	1

Table 2.10: Pearson correlation between PDR and other metrics on worm dataset. Cells are colored by their value. Dark red indicates high correlation. "# misassembly" is negated as less misassembly is preferred.

	Genome fraction	PDR	# misassemblies	NG50	NGA50
Genome fraction	1	0.908765273	0.24103203	0.708747995	0.729736332
PDR	0.908765273	1	0.571649643	0.844333425	0.890795898
# misassemblies	0.24103203	0.571649643	1	0.411291529	0.725016538
NG50	0.708747995	0.844333425	0.411291529	1	0.625937718
NGA50	0.729736332	0.890795898	0.725016538	0.625937718	1

On the worm dataset, PDR had even higher correlation to other metrics except the number of misassemblies. It is worth noting that the correlation between PDR and the number of misassemblies was less stable. As a correctness metric, the number of misassemblies weights every misassembly equally, regardless the impact of each single misassembly. This makes it only partially responsive to and reflective of assembly quality. And this is exactly what PDR addresses.

2.3.6 Non-reference genome

In practice, a reference genome is sometimes absent. So, PDR performance is also studied using a close species' genome as the reference. Table 2.11 shows the metrics of the human dataset, computed based on a human reference genome and a chimpanzee genome (RefSeq assembly accession: GCF_002880755.1).

Among these metrics, NG50 was least changed because it only takes reference size into consideration, regardless of its content. Besides, genome fraction reasonably kept its order as well. However, the

Table 2.11: Evaluation metrics for 3 human assemblies from HG001 dataset. Given human or chimpanzee genome as reference, each metric was computed for these 3 assemblies. The best values are colored blue while the worst values are colored red.

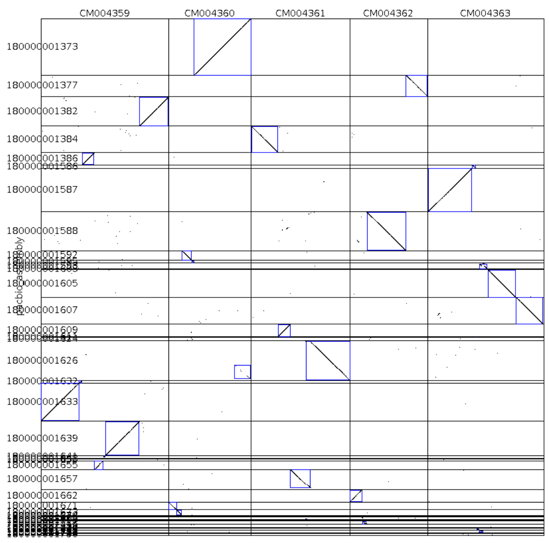
Reference	Metric	UpperBound	Canu	Flye
Human	Genome fraction	99.07%	92.25%	91.91%
	# misassemblies	0	853	673
	NG50	7,862,149	3,241,232	3,767,461
	NGA50	7,862,149	2,744,681	3,172,168
	PDR	94.24%	82.73%	79.64%
Chimpanzee	Genome fraction	88.43%	85.13%	72.85%
	# misassemblies	5632	3630	3029
	NG50	8,113,773	3,311,085	3,810,280
	NGA50	2,109,898	1,377,976	922,952
	PDR	75.39%	68.82%	64.46%

number of misassemblies became unreliable as many differences between chimpanzee and human were counted into this metric. NGA50 was thus influenced. It is originally designed to obtain pure contiguity metric by eliminating correctness impact from NG50. But under a close species' genome, some contigs were broken into smaller pieces due to reference differences. For example, in Table 2.11, Canu had lower NGA50 under human genome, while Flye became lower under chimpanzee genome. Therefore, NGA50 and misassemblies count are not effective for comparing assemblies based on close genome. Fortunately, most structural differences between close species only involve short fragments. As a result, PDR was not caused to have large fluctuations by these differences. Meanwhile, its sensitivity to significant misassemblies was retained for quality comparison of assemblies.

2.3.7 Discrimination on similar assemblies

When assemblies come from different sequencing platforms or assemblers, they usually differ from each other in many aspects due to technical characteristics. But for those similar assemblies, e.g. one is refined or derived from another, the traditional metrics usually lack of discrimination power.

This can be illustrated by an *A. Thaliana* example. The *A. Thaliana* reference genome was downloaded


(a) MUMmer plot: reference genome vs. Draft assembly

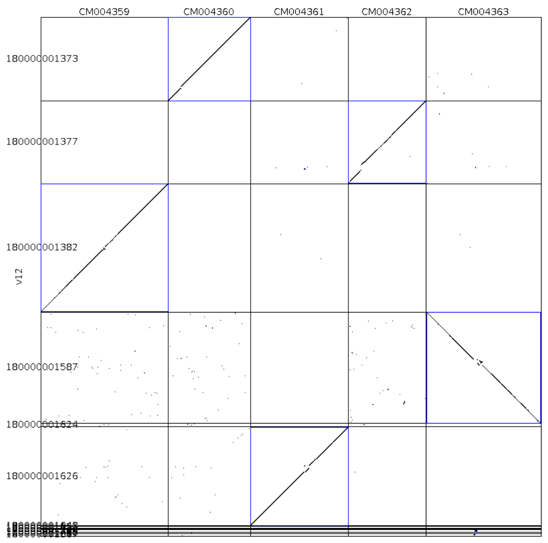

(b) MUMmer plot: reference genome vs. refined assembly

Figure 2.8: MUMmer plots of the draft assembly and the refined assembly against the reference genome. In each MUMmer plot, X-axis is the reference genome while Y-axis is an assembly. Each diagonal black line (which may degrade to a dot) indicates an alignment between corresponding reference segment on X-axis and assembly segment on Y-axis. Blue box is for easy interpretation.

Table 2.12: Metrics on *A. thaliana* draft assembly and its refined assembly. Change rates are calculated by (Refined-Draft)/Draft*100%

Assembly	Draft	Refined	Change rate
Genome Fraction (%)	98.797	98.795	-0.002%
Misassembly Count	2224	2184	-1.8%
NG50	7,853K	22,731K	+189.46%
NGA50	778K	784K	+0.77%
PDR	84.67%	98.02%	+15.77%

from NCBI (GenBank accession number: GCA_001651475.1). A draft assembly was obtained from Pacbio public sample data (https://github.com/PacificBiosciences/DevNet/wiki/Arabidopsis-P5C3). It was refined to a more contiguous assembly by scaffolding (method detailed in Chapter 3). We used SyMap [Soderlund et al., 2006] which invokes MUMmer [Marçais et al., 2018] to plot alignments between the reference genome and the draft assembly as well as the refined assembly. The alignments are shown in Figure 2.8.

Intuitively, the chromosome-level refined assembly is much better than the fragmented draft assembly.

However, this is not reflected by traditional metrics. Table 2.12 lists the typical metrics on these two assemblies. After scaffolding, NGA50 only increased less than 1%, and misassembly count only decreased less than 2%. More than that, genome fraction even counter-intuitively decreased 0.002%. Obviously, all these did not match the sense we obtained from Figure 2.8: the scaffolding indeed improved the assembly. Only NG50 reflects this improvement. However, NG50 is the least reliable metric among them. Because it would be very large even when all contigs were disorderly concatenated together. In other words, with only NG50, one can not tell whether the improvement on contiguity is achieved at a huge cost to correctness. Compared to these typical metrics, PDR profiled and quantified the improvement by a more reasonable value change. A value of 98% for the refined assembly also suggests that there is almost no room for further improvement on the refined assembly.

2.4 Discussion

In this chapter, I have proposed a new metric PDR to integrate all three aspects in assembly assessment, i.e. contiguity, completeness, and correctness. In fact, PDR does not provide a new aspect, but instead weights these three aspects in the context of genetic studies. Each contig is weighted by its value in downstream analysis, while each misassembly is penalized by its misleading impact. As a reasonable weighted sum, the overall score is able to guide the selection of assemblies from various library construction strategies, sequencing technologies, and assemblers. Meanwhile, I have also proposed an efficient implementation of PDR. My implementation runs in minutes instead of the hours needed by a brute-force implementation, and the results show that the introduced error is usually extremely low and thus negligible.

CHAPTER 3

CAST: refining a parent assembly using many progenies' reads

3.1 Background

Although a lot of efforts have been made to improve genome assembly algorithms for better performance, limited improvement can be made on assemblers to overcome repeat regions because short sequencing reads by themselves do not provide enough information for this purpose. Instead of optimizing assembly algorithms, researchers have tried to adopt new assembly strategies.

3.1.1 Mate-pair library

The most commonly-used strategy is mate-pair sequencing. By generating long-insert paired-end libraries, two mates of a read are able to flank a repeat region. Therefore, many assemblers, for example SPADES [Bankevich et al., 2012] and SOAPdenovo2 [Luo et al., 2012], support mate-pair library as scaffolding evidence. For a large genome, multiple mate-pair libraries with various insert sizes may be prepared to provide hierarchical resolutions. Even so, the typical insert sizes of mate-pair libraries are usually not larger than 20kb. This limits its application on repeat regions longer than this value, which is not rare in highly repetitive genomes like plants'. Moreover, with the commercialization of long read sequencing technologies (e.g. Oxford Nanopore and Pacbio SMRT), mate-pair sequencing is gradually being replaced.

3.1.2 Optical map

To better deal with long repeat regions, informations from other sources were integrated to form a backbone (also called a skeleton), to which contigs are anchored or mapped. Based on the backbone, contigs are assigned with order and orientation to yield a more informative assembly. A choice of information source is optical maps. Neto et al. [2011] described the application of optical mapping in *de novo* assembly, which places contigs by mapping the *in silico* restriction map of contigs to the optical map. The direction and position of an anchored contig can thus be determined. Although an optical map is able to provide chromosome-scale information to improve assembly, the relatively high cost of experiments makes it unattractive for projects with limited budget.

3.1.3 Chromatin interaction

Besides optical map, chromatin interaction data can also be used to improve assembly. Contact frequency between a pair of loci is negatively correlated with distance on a chromosome sequence. Burton et al. [2013] exploited this to generate chromosome-scale *de novo* assemblies of human, mouse and *Drosophila* genomes by NGS and Hi-C data. This idea was further developed in assembly pipelines like 3D-DNA [Dudchenko et al., 2017] and SALSA [Ghurye et al., 2017]. These pipelines not only merge contigs but also try to split suspicious misassembled contigs based on discordant contact frequency between adjacent regions. Although these pipelines showed substantial effect on assembly improvement, their accuracy are limited by the low resolution of Hi-C data. The complexity of library preparation as well as data analysis is another factor that further restricts the use of this assembly strategy.

3.1.4 Genetic map

Another information source is genetic maps, which depict the relative locations of genes and other markers by exploiting the idea of linkage: the closer two genes are to each other, the greater probability that they will be inherited together. As a traditional method, genetic maps are well studied even before sequencing became common, and thus are easy to obtain for some model species. For example, Mascher et al. [2013] proposed a pipeline to anchor contigs onto an existing genetic map; and in the absence of an available genetic map, to first also infer a genetic map from population data. Before placement of contigs, another pipeline RPGC [Hahn et al., 2014] first polished an initial assembly. It iteratively identified and corrected split alleles and collapsed alleles (i.e. repeats that have wrong copy numbers in assembly) by abnormal allele segregation. This improved the precision of genotyping. Therefore, the later genetic-map construction and contigs anchoring benefited from this. Rather than loose coupling of assembly and genotyping, Nossa et al. [2014] integrated SNP identification into the assembly process for better genotyping with low sequencing depth. First, reads were assembled by a de Bruijn graph with highly conservative criteria. K-mer pairs with only one mismatch to each other were tracked and termed SNPmer. Samples' genotypes at each SNPmer were then inferred by a probabilistic model. Finally, SNP markers were ordered by hierarchical clustering.

These pipelines have been successfully applied in a few studies. In some of these studies, descendants are specially sequenced for parents' genome assembly. More commonly, in many crop-breeding research projects, the sequencing data of descendants are already available as part of these projects and thus can be directly used to help the assembly of parent strains. However, there are still some limitations of genetic maps. In cases where the organism is polyploid[1], some traditional genetic-map algorithms are not able to handle complex heterozygote[2]. Another limitation is that if the organism has hundreds of thousands of markers, genetic-map construction becomes very time consuming, even infeasible. Besides, these pipelines only try to anchor and order the contigs; they do not correct any misassembly inconsistent with the genetic map. In addition, these pipelines are not fully automated as they utilized existing tools by a few customized scripts. Even worse, some of their components have become unavailable due to lack of maintenance.

3.2 Method

3.2.1 Overview

To address the problems in existing genetic map-based methods, I present a new method, CAST (Correction And Scaffolding Tool), to improve draft assembly by sequencing data of a progeny population. It exploits the fact that the closer two alleles are, the more likely they are passed on to

[1]The state of a cell or organism having more than two paired (homologous) sets of chromosomes.

[2]An individual who has more than one different form of a particular gene.

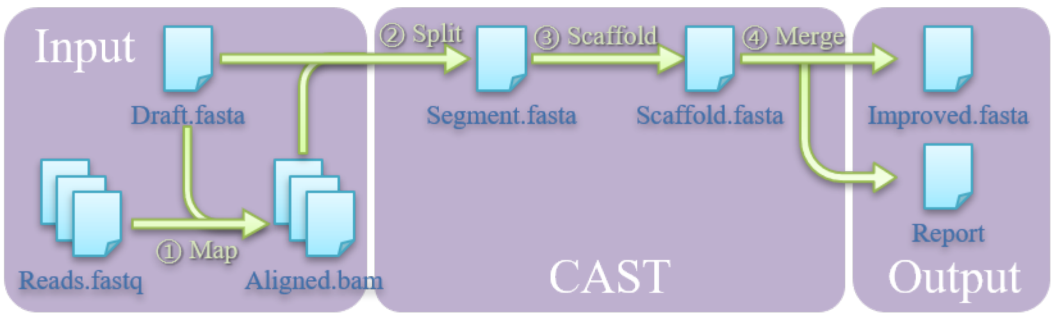


Figure 3.1: The pipeline of CAST. CAST takes a draft assembly and a batch of progeny alignments as input. It first splits contigs in the draft assembly at incoherent sites, and then scaffolds them by coherent sites. Later, contigs are merged by overlap. Finally, it produces an improved assembly as well as a coherence report.

the next generation together unless recombination happens. This idea is common in genetic-map construction. But instead of mapping genomic sequences to a genetic map, CAST inspects genetic coherence along contigs in an initial draft assembly. Contigs are split by adjacent sites incoherent to each other, and then merged by coherent sites. In this way, the draft assembly is improved without time-consuming construction of a genetic map.

Figure 3.1 shows the pipeline of CAST. Progeny reads are previously mapped to the parent draft assembly by an alignment tool, e.g. BWA [Li and Durbin, 2009]. Then, CAST takes the draft assembly and the progeny alignments in BAM format as input, and goes through each contig in the draft assembly to inspect whether adjacent sites are coherent in genotype. It splits contigs at incoherent sites, and links unsplit contigs together with split contigs by genotype coherence at ends of contigs. For each linked chain of contigs, CAST tries to merge neighbouring contigs by overlap. Finally, it outputs an improved assembly in fasta format as well as a contig-linkage report.

3.2.2 Notation and definition

For a given position (i.e. a single base on a reference genome) i on some contig, the counts of 5 possible alleles (A, C, G, T, and gap) from aligned reads of progeny j form a 5 dimensional vector c_{ij}. The genotype of progeny j at position i is defined as a vector comprising of these 5 allele ratios, i.e. $r_{ij} = c_{ij}/\sum c_{ij}$. Figure 3.2 shows two examples. Compared to the traditional character-based genotype representation, this quantitative definition is able to better describe heterozygote, even potentially in polyploidy. To profile the correlation among progenies, CAST calculates a distance

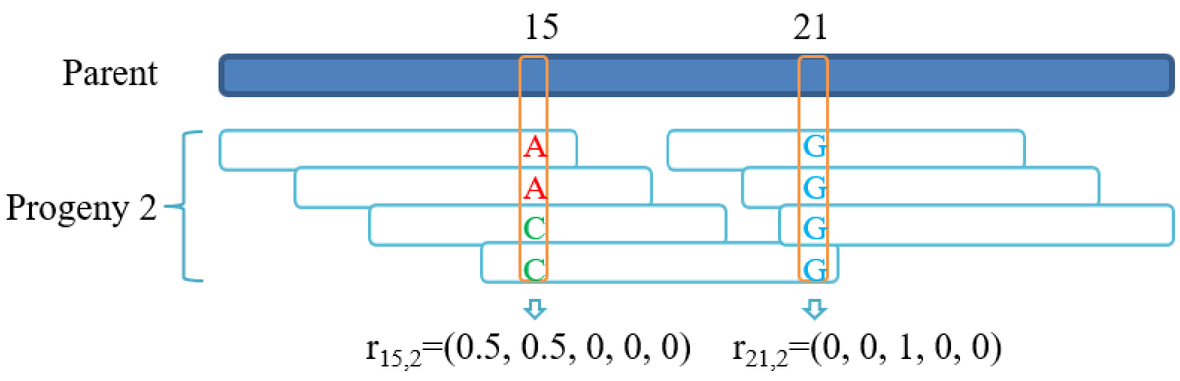


Figure 3.2: Examples of genotyping for one progeny. First, reads of this progeny (cyan bars) are mapped to the parent's contigs (dark blue bar). Then, each column in the alignment is inspected. Orange boxes show two examples. The genotypes of these two columns are corresponding ratios, in order, of base A, C, G, T, and gap.

matrix D_i for each position i; the k-th element of the j-th row is the half Manhattan distance between the genotypes of progeny j and progeny k:

$$d_{jk} = \frac{\tau(r_{ij}, r_{ik})}{2} \tag{3.1}$$

where τ is the Manhattan distance. In such a matrix, each element $0 \leq d_{jk} \leq 1$ indicates the dissimilarity between progeny j and progeny k. Obviously, there is no difference among progenies on most positions. Such positions (called non-polymorphic positions) contain no genetic information and thus they are not considered. Figure 3.3 gives a few examples on polymorphic positions. For simplicity, the examples hereafter are double haploid, so that genotypes are homozygous. Note that D_{18} and D_{10} have similar matrices and thus are coherent even if their genotypes are totally different. Specifically, d_{jk} is a similarity matrix reflecting whether progeny j and progeny k are likely to have inherited their genotype at position i from the same parent (i.e. their father or their mother). Theoretically, these distance matrices are enough to profile the correlation of progenies. But noise from sequencing errors and erroneous alignments may prejudice genotype and thus single positions cannot be directly used to make decision. CAST solves this problem by combining multiple consecutive polymorphic positions to dilute effects from unreliable positions. Specifically, CAST defines a haplotype length H (0.1% of genome length by default) and average all polymorphic positions D within a range of H. For each polymorphic position i, the left (right) haplotype matrix $L_i(R_i)$ is computed by Equation 3.2

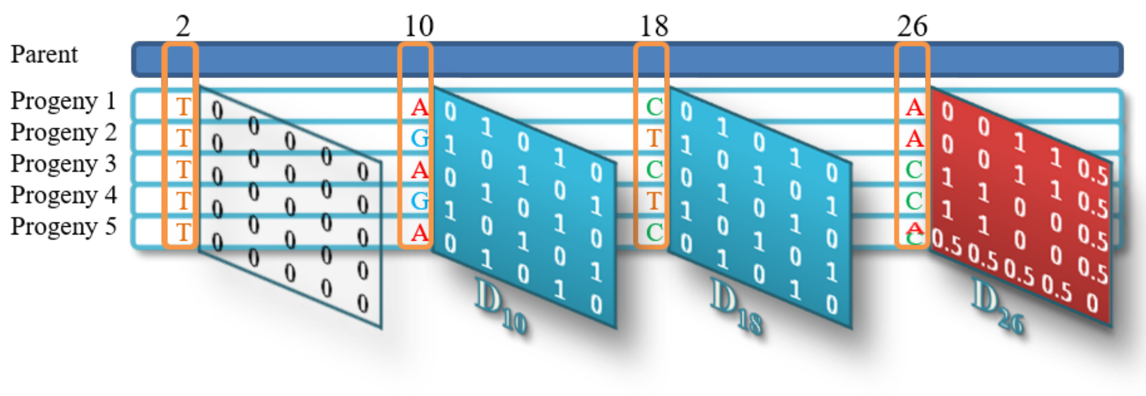


Figure 3.3: Examples of distance matrices. The dark blue bar represents a parent contig, while each cyan bar here is not a read but genotype information of one progeny from its reads. At position 2, all progenies have same genotype (0,0,0,1,0), thus D_2 is a zero matrix (transparent). At position 10, three progenies have (1,0,0,0,0) while another two progenies have (0,0,1,0,0). At position 18, three progenies have (0,1,0,0,0) while another two progenies have (0,0,0,1,0). D_{10} and D_{18} have the same color because they are similar. At position 26, progeny 5 has genotype (0.5,0.5,0,0,0). D_{26} is differentially colored as it has different pattern from D_{18} and D_{10}.

(Equation 3.3). Figure 3.4 shows an example of these computations.

$$L_i = Avg\{D_x | i - H \leq x \leq i\} \tag{3.2}$$

$$R_i = Avg\{D_x | i \leq x \leq i + H\} \tag{3.3}$$

Given two haplotype matrices L_p and R_q, their sum of absolute difference is defined as:

$$\delta(L_p, R_q) = \sum |L_{p_{ij}} - R_{q_{ij}}| \tag{3.4}$$

$\delta(L_p, R_q)$ is likely to be small when there is genetic linkage between positions q and p. Otherwise, it should be much larger.

3.2.3 Correction

CAST utilizes sliding windows to compute haplotype matrices L_i and R_i for each polymorphic position i. During the scan, CAST checks whether each pair of adjacent polymorphic positions v and w are coherent. A sampling-based permutation test (see Section 3.2.5) is performed on L_v and R_w, for the null hypothesis that positions v and position w are not close to each other and thus not

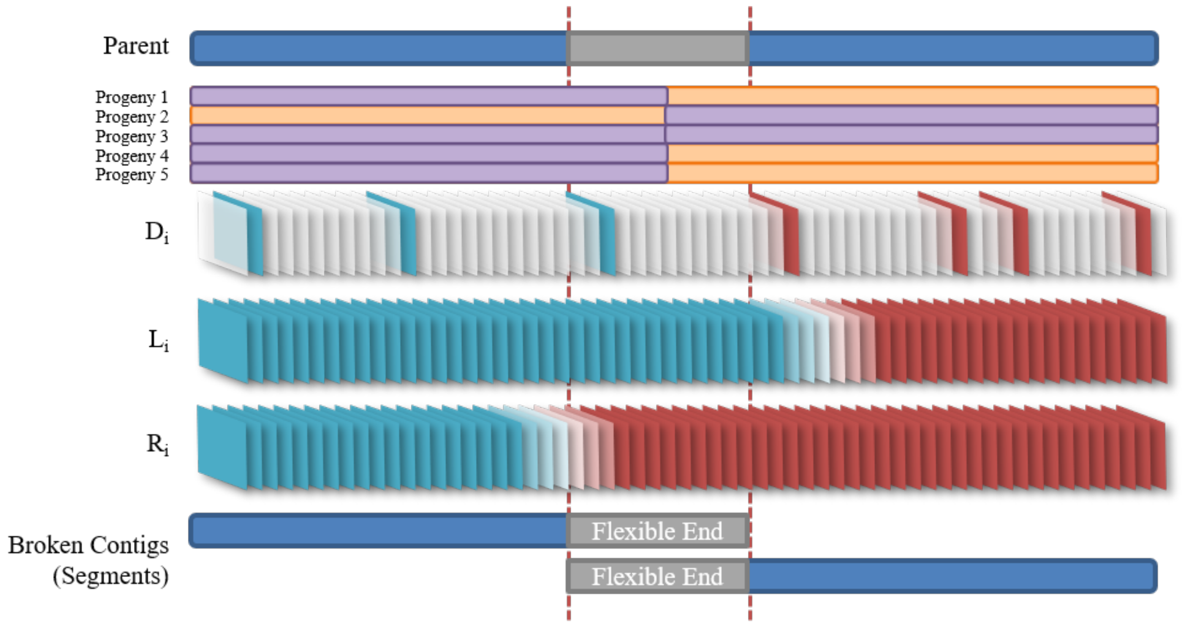
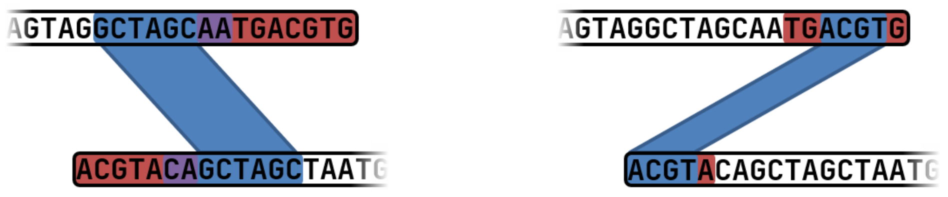


Figure 3.4: Examples of haplotype matrices and splitting. The top layer is a parent contig, on which the grey region harbors a misassembly. The tracks in the second layer shows inheritance of five progenies. Orange segments come from one parent; purple segments come from the other parent. The third layer is distance matrices. Grey matrices are zero matrices due to non-polymorphism, while cyan and red matrices are from different chromosomes (misassembled into this contig). Fourth and fifth layers show L_i and R_i respectively. Around the misassembled region, haplotype matrices are converted gradually as indicated by gradient colors. Finally, the parent contig is split into 2 segments as shown in bottom layer.

bound by genetic linkage. The permutation test shuffles the label of progenies and then compute empirical distribution of $\delta(L_v, R_w)$. If p-value of the test is larger than a predefined significant level P, a misassembly between p_v and p_w is assumed. In such a case, the contig is split into two "broken" contigs and each broken contig retains a copy of the sequence from p_v to p_w for potential merging, named flexible ends (see grey regions in Figure 3.4). Flexible ends are free to be partially or totally discarded during merging because the assumed misassembly may happen at an arbitrary position between p_v to p_w. Note that a contig may be split more than once to generate several broken contigs. Hereafter, original contigs without splitting and broken contigs are all called segment for easy reference. Each segment may have zero, one, or two flexible ends depending on the splittings.

3.2.4 Scaffolding and merging

The genetic information of each segment can be characterized by its left-most haplotype matrix L_1 and right-most haplotype matrix R_w, which are termed joints in the following description. CAST computes δ for all pairs of joints from different segments. Two joints are considered to be close to



(a) Alignment with loss=4 **(b)** Alignment without loss

Figure 3.5: Two potential alignments between a pair of segments. Red regions indicate flexible ends while blue regions are alignments. (a) shows an alignment where aligned regions are out of flexible ends on both segments. The regions between aligned regions and flexible ends count as loss (marked purple). So, loss=4 for this case. (b) is another alignment where the aligned regions harbour within flexible ends and thus loss=0.

each other in the underlying genome if their joint matrices A and B has a p-value smaller than P in permutation test described above. CAST reports these two joints with their δ value as a link.

It is common for a joint to have more than one link. To determine which is the best one to perform merging with, CAST first sorts segments by their lengths, and then places them one by one from long to short. For a given segment to place, it first forms a new scaffold. If this segment has no link on its both joints, the placement is done. Otherwise, for its any joint with at least one link, the link with the minimum δ is chosen to be a candidate. If any candidate links to a left-most or right-most joint of some scaffold, then the new scaffold is merged with that one.

During merging, BLAST is called to find all significant alignments ($E - value < 10$ by default) between the two involved segments in scaffolds. For each alignment, the loss of this alignment is defined as the length of sequence between flexible end and aligned region (purple regions in Figure 3.5), which is the unique sequence that will be discarded if merging is performed by this alignment. In this definition, flexible ends are not counted. Because they are duplicated in splitting and can be totally discarded without information loss. Therefore, if an aligned region overlaps with a flexible end on a segment, then the alignment has no loss on this segment. For example in Figure 3.5a, if merging is performed by this alignment, the sequence after the aligned region in the top contig and the sequence before the aligned region in the bottom contig will be discarded. But the loss only comes from purple regions, which is 4. In Figure 3.5b, the sequences need to be discarded are all parts of flexible ends, so the loss is 0. Based on this definition of loss, alignments with a loss greater than their aligned region length are filtered out. If no alignment is left, two segments are merged by 50 N's in middle (called gapped merging). Otherwise, they are simply merged by overlapping aligned regions in the alignment with minimum loss. On each segment, the overhanging end beyond the aligned region is truncated. For the sake of efficiency, sequence-merge operations are postponed in my implementation and only applied after segments are all placed.

Sometimes, candidates may link to a joint in the middle of some scaffold. Such candidates are usually ignored, except when two candidates of the same segment link to two joints previously gapped-merged. This means the segment is supposed to place in the gap. In this case, the previous gapped merging is reverted; and then the candidates are applied for merging. Note that this is reasonable because the segments are placed from long to short, in this way the scaffolds are constructed from backbone to detail. The advantages of this approach is to ensure the chromosome-level scaffolding first, and then endeavour to resolve local conflicts which may have no perfect solution.

3.2.5 Implementation details

In all sequencing platforms, sequencing errors are prevalent. They may bias genotyping and produce incoherent distance matrices. Especially, when sequencing depth is low, the genotype may be totally converted to an incorrect one. To avoid this, CAST ignores bases with phred quality score less than 20. Although this cannot clean all sequencing errors, the survived sequencing errors are sparse enough to be diluted by neighbouring distance matrices.

Besides platform noise, spontaneous mutations also influence distance matrices. When a new mutation in a single progeny converts a non-polymorphic position to a polymorphic one, the generated distance matrix has one non-zero row and one non-zero column. Obviously, this matrix is incoherent to neighboring matrices, causing erroneous splitting around this position. CAST identifies these mutations by checking whether one progeny's genotype is roughly equally far from others. Specifically, CAST inspects each row of a distance matrix after removing its principal diagonal. If the difference between the lowest and highest elements in any row is less than a threshold γ (0.2 for diploid by default), the position corresponding to this distance matrix is regarded as a spontaneous mutation and ignored. With this rule, non-polymorphic positions are also filtered.

Another interference in genotyping is due to dispersed repeats on different chromosomes. The reads from such repeats may map to arbitrary copies. Consequently, genotypes from the same sample but different chromosomes pollute each other, and distance matrices also interpenetrate. Thus, repeat regions should be excluded from scans. To this end, reads are inspected and called suspicious if any of the following criteria is satisfied:

- map quality (i.e. MAPQ field in BAM) is less than 30
- two mates of a paired read map to different contigs
- alternative mapping exists
- total length of indel, clip, and mismatch exceeds 10% of read length

Then, a position is identified as repeat if more than 10% of reads spanning it are suspicious in any progeny. A position is also identified as repeat if its sequencing depth is larger than 1.8 times average. Such positions are ignored.

The permutation test between adjacent haplotype matrices are computationally intensive. In fact, there is no need to enumerate all permutations when the number of progenies is large. CAST uses Monte Carlo to obtain a $\delta(A, B)$ null distribution by M (10000 by default) permutations of progeny labels. Each permutation keeps A untouched and shuffles progeny labels of B. Then, A and shuffled B are used to compute distance matrices, on which δ is calculated later. Finally, the $(P * M)$th smallest value among M δ values is the threshold for significant level P.

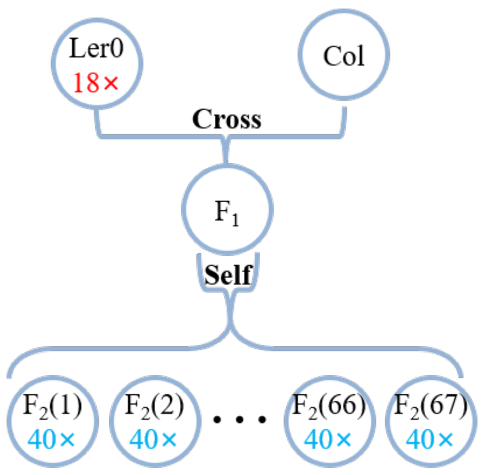
When the number of polymorphic positions is also large, the permutation test is still time-consuming even with Monte Carlo. To solve this problem, CAST first performs a preliminary Monte Carlo where $M = 1$. δ values are collected from all pairs of adjacent polymorphic positions and form a distribution. This distribution is used instead in the test.

3.3 Evaluation

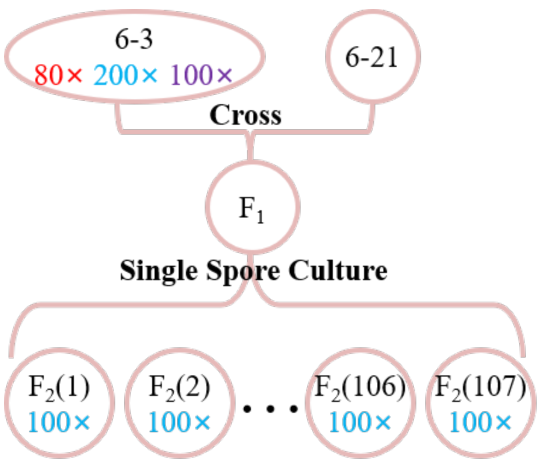
3.3.1 Datasets

To evaluate the improvement of genome assembly processed by CAST, I used two datasets with different experiment materials and designs.

The first dataset comes from a parent–progeny project on *Arabidopsis thaliana* [Yang et al., 2015]. Figure 3.6a illustrates the relationship between strains and data. The authors crossed two typical *A. thaliana* strains Ler0 and Col to get F_1; and then a single F_1 seed was selfed to generate multiple F_2



(a) *A. thaliana* dataset



(b) *F. velutipes* dataset

Figure 3.6: Illustration of datasets. Each circle represents an individual. The red and blue numbers in circles indicate the sequencing depth of Pacbio and Illumina platform, respectively. Purple number indicates the sequencing depth of Hi-C experiment. (a) In *A. thaliana* dataset, Ler0 was crossed with Col to get $F1$. $F1$ was then selfed to generate $F2$. Ler0 and 67 $F2$ individuals were sequenced by Pacbio and Illumina platform respectively. (b) In *F. velutipes* dataset, 6-3 was crossed with 6-21 to get $F1$. 107 spores generated by $F1$ were single-spore cultured as $F2$.

individuals, among which 67 individuals were sequenced by Illumina platform at depth of $\sim 40\times$. I randomly downloaded 31 progenies out of 67 from NCBI (BioProject database accession number: PRJNA178613, see Table 3.1 for details) for evaluation. The parent Ler0's Pacbio assembly was obtained from Pacbio public sample data (https://github.com/PacificBiosciences/DevNet/wiki/Arabidopsis-P5C3).

The reference genome in evaluation was downloaded from NCBI (GenBank accession number: GCA_001651475.1).

The other dataset is from our ongoing *Flammulina velutipes* project to be published later. Figure 3.6b illustrates the relationship between strains and data. A monokaryon strain 6-3 was crossed with another monokaryon strain 6-21 to generate a hybrid strain F_1. Spores of F_1 were collected and cultured as monosporous strains, of which 107 strains were sequenced later by Illumina platform at depth of $\sim 100\times$. A random subset of 30 strains were used for evaluation. The parent strain 6-3 was sequenced by Illumina platform at depth of $\sim 200\times$ and Pacbio platform at depth of $\sim 80\times$. I evaluated improvement on both 6-3's Illumina assembly and Pacbio assembly. Illumina assembly was performed using Spades, while Canu and Pilon [Walker et al., 2014] were used in Pacbio assembly. A BAC-based chromosome-level assembly was downloaded from NCBI genome database (GenBank accession number: GCA_000633125.1) as the reference genome in evaluation. In addition, 6-3 was

Table 3.1: A. Thaliana dataset accession numbers.

BioSample	Run	Platform	Organism
SAMN01797620	SRR611079	ILLUMINA	Arabidopsis thaliana
SAMN01797660	SRR611084	ILLUMINA	Arabidopsis thaliana
SAMN01797661	SRR611085	ILLUMINA	Arabidopsis thaliana
SAMN01797630	SRR611092	ILLUMINA	Arabidopsis thaliana
SAMN01797631	SRR611093	ILLUMINA	Arabidopsis thaliana
SAMN01797632	SRR611094	ILLUMINA	Arabidopsis thaliana
SAMN01797633	SRR611095	ILLUMINA	Arabidopsis thaliana
SAMN01797634	SRR611096	ILLUMINA	Arabidopsis thaliana
SAMN01797635	SRR611097	ILLUMINA	Arabidopsis thaliana
SAMN01797636	SRR611098	ILLUMINA	Arabidopsis thaliana
SAMN01797637	SRR611099	ILLUMINA	Arabidopsis thaliana
SAMN01797638	SRR611100	ILLUMINA	Arabidopsis thaliana
SAMN01797639	SRR611101	ILLUMINA	Arabidopsis thaliana
SAMN01797640	SRR611102	ILLUMINA	Arabidopsis thaliana
SAMN01797641	SRR611103	ILLUMINA	Arabidopsis thaliana
SAMN01797642	SRR611104	ILLUMINA	Arabidopsis thaliana
SAMN01797643	SRR611105	ILLUMINA	Arabidopsis thaliana
SAMN01797644	SRR611106	ILLUMINA	Arabidopsis thaliana
SAMN01797645	SRR611107	ILLUMINA	Arabidopsis thaliana
SAMN01797646	SRR611108	ILLUMINA	Arabidopsis thaliana
SAMN01797647	SRR611109	ILLUMINA	Arabidopsis thaliana
SAMN01797648	SRR611110	ILLUMINA	Arabidopsis thaliana
SAMN01797649	SRR611111	ILLUMINA	Arabidopsis thaliana
SAMN01797650	SRR611112	ILLUMINA	Arabidopsis thaliana
SAMN01797654	SRR611113	ILLUMINA	Arabidopsis thaliana
SAMN01797655	SRR611114	ILLUMINA	Arabidopsis thaliana
SAMN01797656	SRR611115	ILLUMINA	Arabidopsis thaliana
SAMN01797657	SRR611116	ILLUMINA	Arabidopsis thaliana
SAMN01797658	SRR611117	ILLUMINA	Arabidopsis thaliana
SAMN01797659	SRR611118	ILLUMINA	Arabidopsis thaliana
SAMN01797659	SRR616982	ILLUMINA	Arabidopsis thaliana

also sequenced by Hi-C experiments at depth of $\sim 100\times$ to be used in comparison of CAST and Hi-C scaffolding tools. All data are publicly available (Accession numbers are listed in Table 3.2).

3.3.2 Measurements

The three main dimensions for genome assembly assessment are contiguity, completeness, and correctness. Reliable assemblies are expected to be good on all these dimensions. However, contiguity and correctness are usually dichotomous to some degree. Before assembly, initial reads are fragmented

Table 3.2: F. velutipes dataset accession numbers.

BioSample	Run	Platform	Organism
SAMN10175083	SRR7963008	PACBIO_SMRT	Flammulina velutipes
SAMN10174861	SRR7962694	ILLUMINA	Flammulina velutipes
SAMN10174860	SRR7962695	ILLUMINA	Flammulina velutipes
SAMN10174853	SRR7962696	ILLUMINA	Flammulina velutipes
SAMN10174852	SRR7962697	ILLUMINA	Flammulina velutipes
SAMN10174855	SRR7962698	ILLUMINA	Flammulina velutipes
SAMN10174854	SRR7962699	ILLUMINA	Flammulina velutipes
SAMN10174857	SRR7962700	ILLUMINA	Flammulina velutipes
SAMN10174856	SRR7962701	ILLUMINA	Flammulina velutipes
SAMN10174859	SRR7962702	ILLUMINA	Flammulina velutipes
SAMN10174858	SRR7962703	ILLUMINA	Flammulina velutipes
SAMN10174851	SRR7962704	ILLUMINA	Flammulina velutipes
SAMN10174850	SRR7962705	ILLUMINA	Flammulina velutipes
SAMN10174845	SRR7962706	ILLUMINA	Flammulina velutipes
SAMN10174844	SRR7962707	ILLUMINA	Flammulina velutipes
SAMN10174843	SRR7962708	ILLUMINA	Flammulina velutipes
SAMN10174842	SRR7962709	ILLUMINA	Flammulina velutipes
SAMN10174849	SRR7962710	ILLUMINA	Flammulina velutipes
SAMN10174848	SRR7962711	ILLUMINA	Flammulina velutipes
SAMN10174847	SRR7962712	ILLUMINA	Flammulina velutipes
SAMN10174846	SRR7962713	ILLUMINA	Flammulina velutipes
SAMN10174866	SRR7962714	ILLUMINA	Flammulina velutipes
SAMN10174867	SRR7962715	ILLUMINA	Flammulina velutipes
SAMN10174868	SRR7962716	ILLUMINA	Flammulina velutipes
SAMN10174869	SRR7962717	ILLUMINA	Flammulina velutipes
SAMN10174862	SRR7962718	ILLUMINA	Flammulina velutipes
SAMN10174863	SRR7962719	ILLUMINA	Flammulina velutipes
SAMN10174864	SRR7962720	ILLUMINA	Flammulina velutipes
SAMN10174865	SRR7962721	ILLUMINA	Flammulina velutipes
SAMN10175083	SRR7963007	ILLUMINA	Flammulina velutipes
SAMN10175084	SRR7963009	ILLUMINA	Flammulina velutipes
SAMN10175083	SRR7968349	ILLUMINA	Flammulina velutipes

but with no misassembly. During assembly, reads are iteratively merged into larger fragments and thus sometimes misassemblies are introduced. A lot of criteria are applied in various assemblers to mine clues and reduce the probability of misassembly. Even so, the probability of misassembly dramatically increases when merging around repeats. Some assemblers stop merging to avoid this hazard while others take the risk and attempt for better contiguity.

A few analysis tools as well as some proposed metrics have been used to assess assembly quality.

Three types of misassembly are usually considered:

- relocation: flanking sequences are aligned to the same reference chromosome but with $>$1kb offset
- translocation: flanking sequences are aligned to different reference chromosomes
- inversion: flanking sequences are aligned to the same reference chromosome but on opposite strands

In our evaluation, I focus less on relocation as it is common to have repeat collapse around long tandem repeats. In such a region, the repeat copies are highly similar to each other and thus pileup in an assembler, yielding an underestimated copy number in an assembly. When collapsed repeat regions are aligned to a reference genome, their flanking sequences could be aligned to positions far away.

For contiguity, a well-known measure is NG50 [Earl et al., 2011] defined as the largest contig length l satisfying:

$$\sum_{L_i \geq l} Li \geq 50\% * G \tag{3.5}$$

where L_i is the length of i-th contig and G is the total length of the reference genome. This value reflects the contiguity of longer contigs, ignoring small fragments. However, it simply trusts every contig and does not take misassembly into consideration. Thus one can ridiculously achieve high NG50 by concatenating all contigs.

To fix this problem, Gurevich et al. [2013] proposed NGA50 which has a similar definition to NG50 but uses aligned blocks in length counting instead of whole contig. In common implementation, contigs are first aligned to a reference genome and then broken into aligned blocks without misassembly. Although NGA50 is more reasonable than NG50, it is not perfect. In particular, it does not tolerate collapsed repeats, which is a common misassembly but almost harmless to downstream analysis.

In this work, I choose a commonly used tool QUAST [Gurevich et al., 2013] to compute metrics including NG50, NGA50 and misassembly counts. It was run with option "–skip-unaligned-mis-contigs" to suppress a default mechanism which ignores contigs $>$50% unaligned. In addition, PDR introduced in Chapter 2 was also computed for each assembly, to provide an overall assessment.

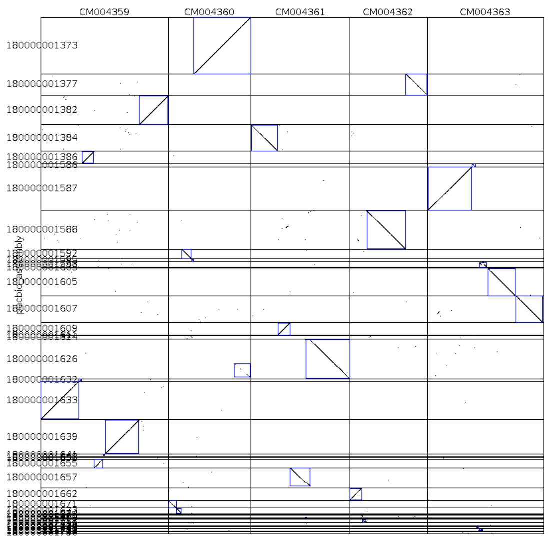
In the evaluations on both datasets, I mainly compared draft assembly with CAST-improved assembly to show the improvement made by CAST. CAST is also supposed to be compared with other genetic-map based methods. However, they are all unavailable now. As commented in Fierst [2015], POPSEQ and RPGC are actually proof-of-principle recipes rather than softwares. POPSEQ's authors only provided a tool for a marker anchoring step as a part of the whole pipeline. RPGC consists of more than 40 commands with several sample-specific parameters. Also, its authors didn't release a key python script used in their final step. Unlike POPSEQ and RPGC, JointAssembly looks like a complete program, but its documentation is incomplete. Worse, it is no longer being maintained. Since all genetic-based methods are unavailable, I compared CAST with 2 Hi-C based method, SALSA and 3D-DNA, on *F. velutipes* dataset where Hi-C data is available.

3.3.3 *A. thaliana* Pacbio assembly

CAST was first tested using the publicly available *A. thaliana* dataset described in Section 3.3.1. The results are shown in Table 3.3.

Table 3.3: Improvement on *A. thaliana* Pacbio assembly. For each statistic, theoretical best values are colored blue; theoretical worst values are colored red.

Assembly	Draft	CAST
# contigs	545	513
Genome fraction (%)	98.797	98.795
Total length	130,858K	130,647K
Total aligned length	118,974K	118,896K
Largest contig	13,211K	29,558K
Largest alignment	4,362K	4,362K
NG50	7,853K	22,731K
NGA50	778K	784K
# relocations	1142	1126
# translocations	1033	1016
# inversions	49	42
PDR	84.67%	98.02%


(a) MUMmer plot: reference genome vs. Draft assembly

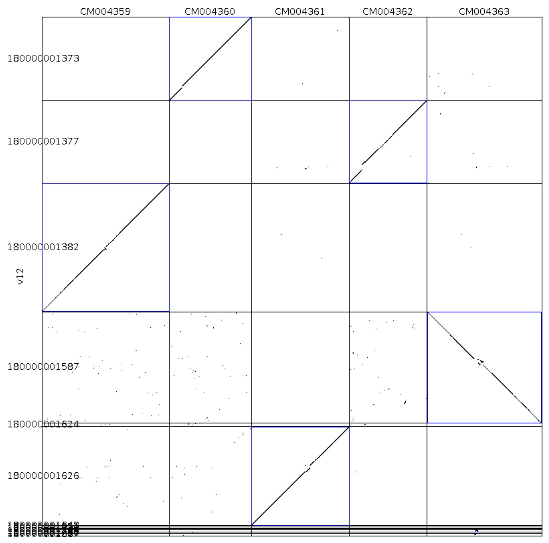

(b) MUMmer plot: reference genome vs. CAST-improved assembly

Figure 3.7: MUMmer plots of the draft assembly and the CAST-improved assembly against the reference genome. In each MUMmer plot, X-axis is the reference genome while Y-axis is a assembly. Each diagonal black line (which may degrade to a dot) indicates an alignment between corresponding reference segment on X-axis and assembly segment on Y-axis. Blue box is for easy interpretation.

The draft assembly (i.e. the assembly to be improved) only yielded slightly better genome fraction, total length, and total aligned length. In fact, this difference came from repeat collapse in CAST merging, which usually does not mean a loss of informative sequence. However, the CAST assembly outperformed the draft assembly in other metrics. Especially, largest contig, NG50, and PDR significantly increased. Also, all three types of misassemblies decreased. This means CAST improved the draft assembly's contiguity and correctness simultaneously. In addition, PDR reveals that CAST has made a big improvement on the draft assembly to a refined assembly of near perfection. This improvement is clearly visible in Figure 3.7. Figure 3.7 is generated by SyMap [Soderlund et al., 2006], which invokes MUMmer to plot alignments between the reference genome and the draft assembly as well as the CAST-improved assembly. It can be seen that the draft assembly was relatively fragmented. However, the CAST-improved assembly presented all five chromosomes, though there were some local breakpoints. The comparison between these two figures illustrates the CAST improvement, which may not be successfully reflected by some common metrics.

This dataset is also used to investigate the effects of sequencing depth and progeny count. To evaluate CAST performance with different sequencing depths, each progeny sequencing data was randomly down-sampled. For progenies whose original data were small, I kept them unchanged during

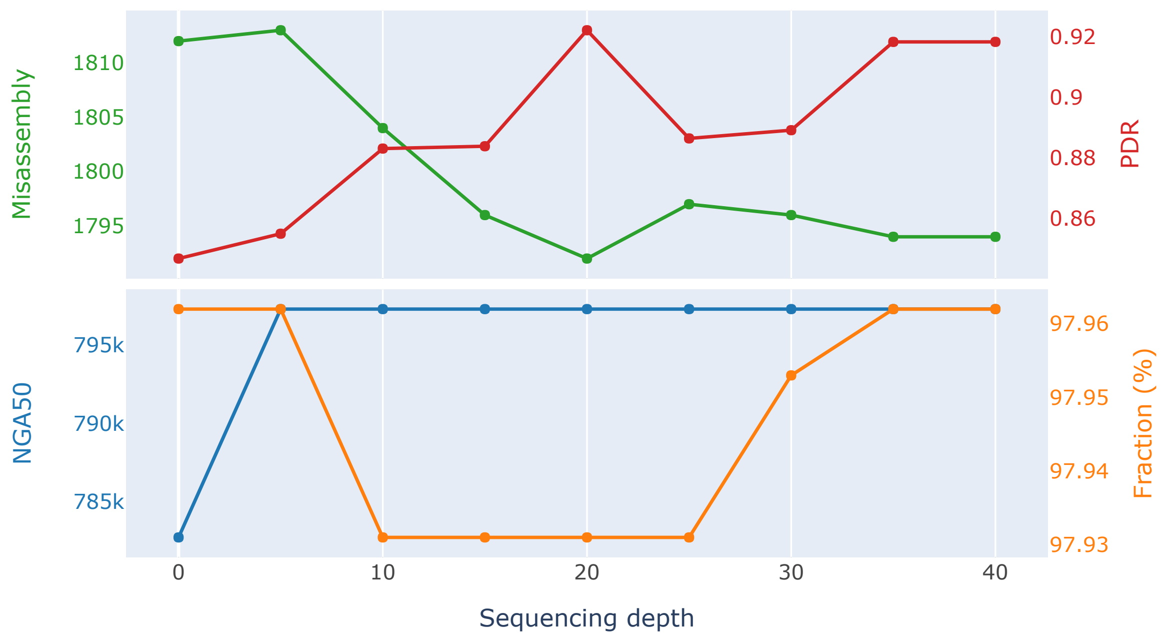


Figure 3.8: The effect of sequencing depth. All 31 progenies were used. The sequencing depth of each progeny was tuned along X-axis by random downsampling. Specially, 0 on X-axis indicates the draft assembly. To show the trend, all four Y-axes do not start from 0. Misassembly (green line) refers to the total misassembly count including relocation, inversion, and translocation.

evaluation for depths beyond their own. Figure 3.8 shows the effect of sequencing depth with all 31 progenies. It can be seen that contiguity profiled by NGA50 converged at sequencing depth above 10-fold. Similarly, the correctness represented by the total misassembly count converged after 20-fold. As a proxy for completeness, genome fraction decreased 0.03% from 10-fold to 30-fold. The reason of this decrease could be randomness in heterozygous positions, which is reduced with increased sequencing depth. In fact, the decrease is not critical since PDR is majorly correlated to misassembly count. Overall, as suggested by PDR, 20-fold depth is enough for improvement, while 35-fold depth produces stable result.

The effect of progeny count is shown in Figure 3.9. Unlike the effect of sequencing depth, contiguity and completeness both converged early from 8 onward. However, limited by correctness, PDR converged from 13 onward. This implies about 15 progenies are enough to make significant improvement. The slight increase of NG50 at 27 progenies also suggested that more progenies can further improve assembly. By contrast, the typical number of progenies needed in genetic-map construction is 50 or more.

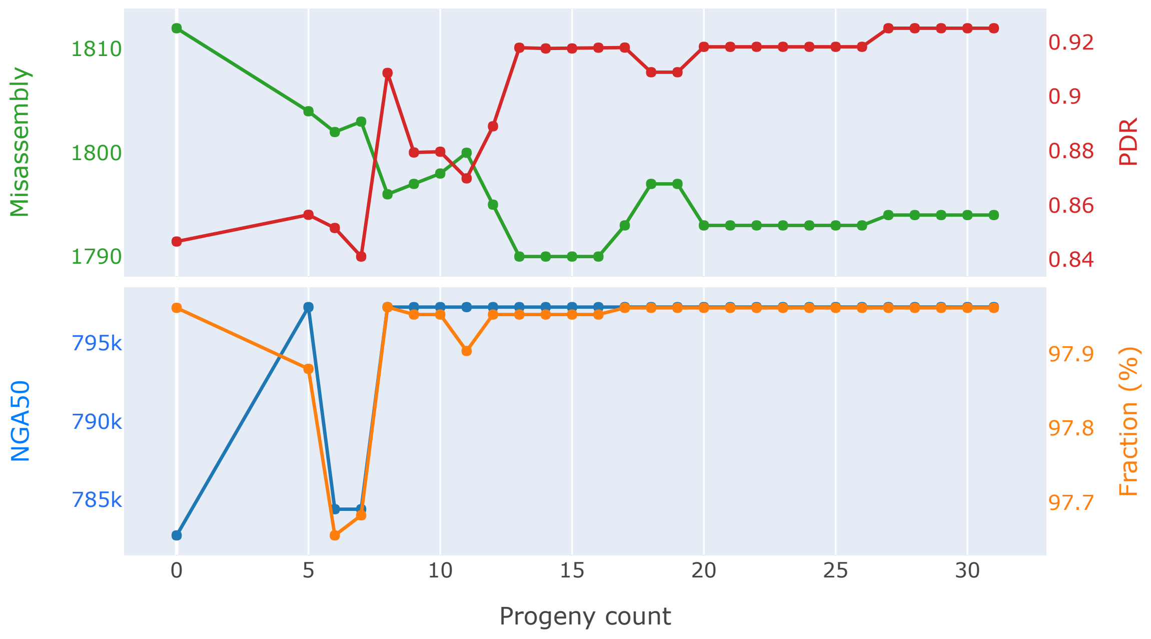


Figure 3.9: The effect of progeny count. The progeny count was tuned along X-axis while original sequencing depths were retained. Specially, 0 on X-axis indicates the draft assembly. To show the trend, all four Y-axes do not start from 0. Misassembly (green line) refers to the total misassembly count which includes relocation, inversion, and translocation.

3.3.4 *F. velutipes* Pacbio assembly

To compare CAST with Hi-C scaffolding tools, I run SALSA and 3D-DNA with 6-3 strain Hi-C data to improve the Pacbio draft assembly. Meanwhile, CAST is also run on the same draft assembly with 31 progeny datasets. Then, all improved assemblies as well as the draft assembly are evaluated by QUAST and PDR against the reference genome.

Table 3.4 shows the metrics for the draft and the improved assemblies. It is worth noting that both SALSA and 3D-DNA yielded the largest contig with length of more than 10Mb. However, the largest reference chromosome is less than 5Mb, which means SALSA and 3D-DNA mistakenly merged at least three chromosomes. This was also reflected in the increase of translocations. In contrast to these two tools, CAST not only reduced relocations and translocations, but also improved NG50 though NGA50 remained unchanged. In other aspects, there are only small differences across all assemblies.

Note that the reference genome is not the exact 6-3 strain. This is the reason that all assemblies only covered $\sim 70\%$ of the reference genome. Also, the reference genome was assembled by Roche 454

Table 3.4: Improvement on *F. velutipes* Pacbio assembly. For each statistic, theoretical best values are colored blue; theoretical worst values are colored red. And the values in purple are abnormal.

Assembly	Draft	CAST	SALSA	3D-DNA
# contigs	32	24	23	114
Genome fraction (%)	71.228	71.229	71.219	71.139
Total length	38,339K	38,246K	38,343K	38,391K
Total aligned length	27,251K	27,193K	27,250K	27,233K
Largest contig	4,487K	4,487K	10,168K	19,474K
Largest alignment	375K	375K	375K	260K
NG50	3,046K	3,174K	4,487K	19,474K
NGA50	30K	30K	30K	25K
# relocations	913	911	917	945
# translocations	1598	1588	1604	1611
# inversions	30	30	29	32
PDR	48.80%	48.96%	46.19%	30.40%

platform. The length of reads produced by Roche 454 platform is about 400 bp, providing limited resolving ability to repeats. Thus, the reference genome may also harbour misassemblies. These two factors are very likely the main sources of the misassembly counts in the evaluation. However, as they contributed roughly same number of misassemblies in the evaluation of each assembly, the comparison between these assemblies still made sense.

To further investigate the modifications made by CAST, I used SyMap to plot synteny of the draft assembly and the CAST assembly against the reference genome. The synteny analysis is shown in Figure 3.10. It can be seen that CAST merged partial or all blocks on chromosomes CM002705, CM002704, CM002700, CM002699, CM002698, CM002697, and CM002696. Meanwhile, it also extended the first block on CM002702. In total, the modifications covered 8 out of 11 chromosomes. Although CAST did not improve NG50 and NGA50 too much, it definitely contributed a lot to contiguity.

In Figure 3.11, the synteny analysis also provides an overview of the correctness of the CAST assembly. The reference genome is aligned on the CAST assembly to label where contigs come from. From

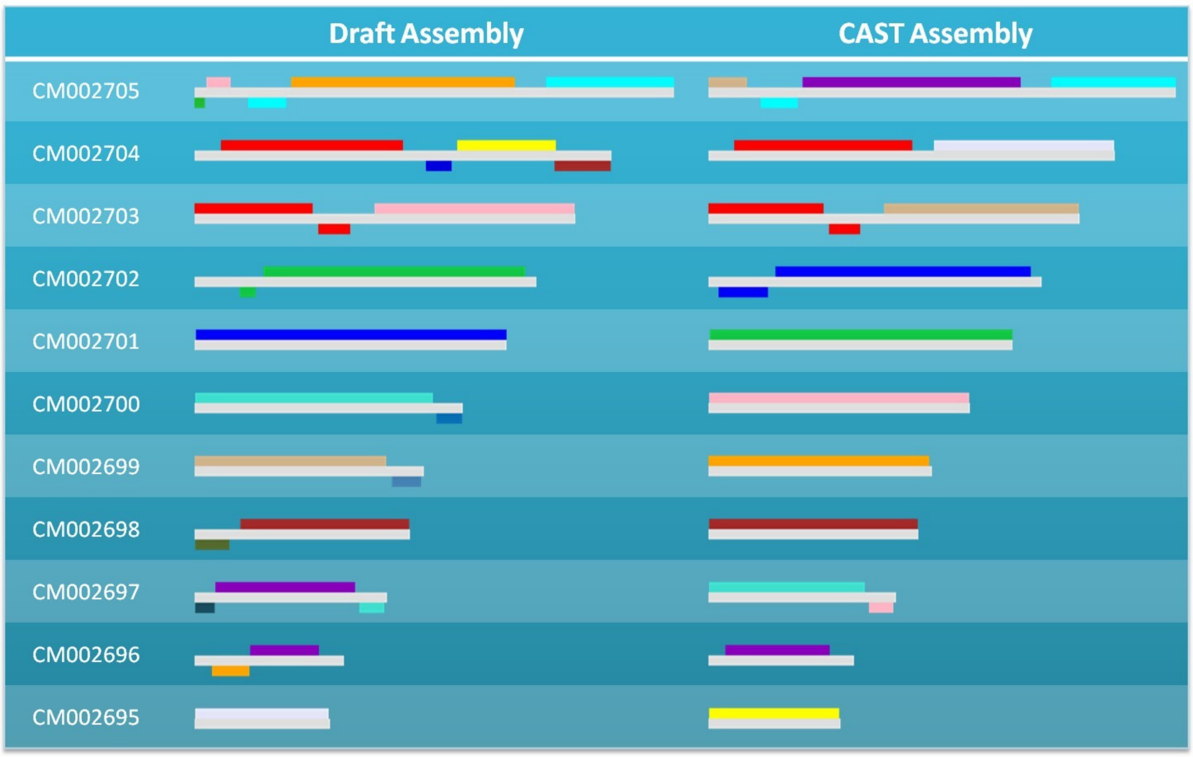


Figure 3.10: Assemblies aligned on the reference genome. Each row indicates a chromosome in the reference genome. Grey bars in the same row represent the same chromosome. Each column is an assembly. The colored bars attached to a grey bar are aligned blocks in this assembly to the reference chromosome. Within an assembly (i.e. a column in the table), bars with identical color are from the same contig.

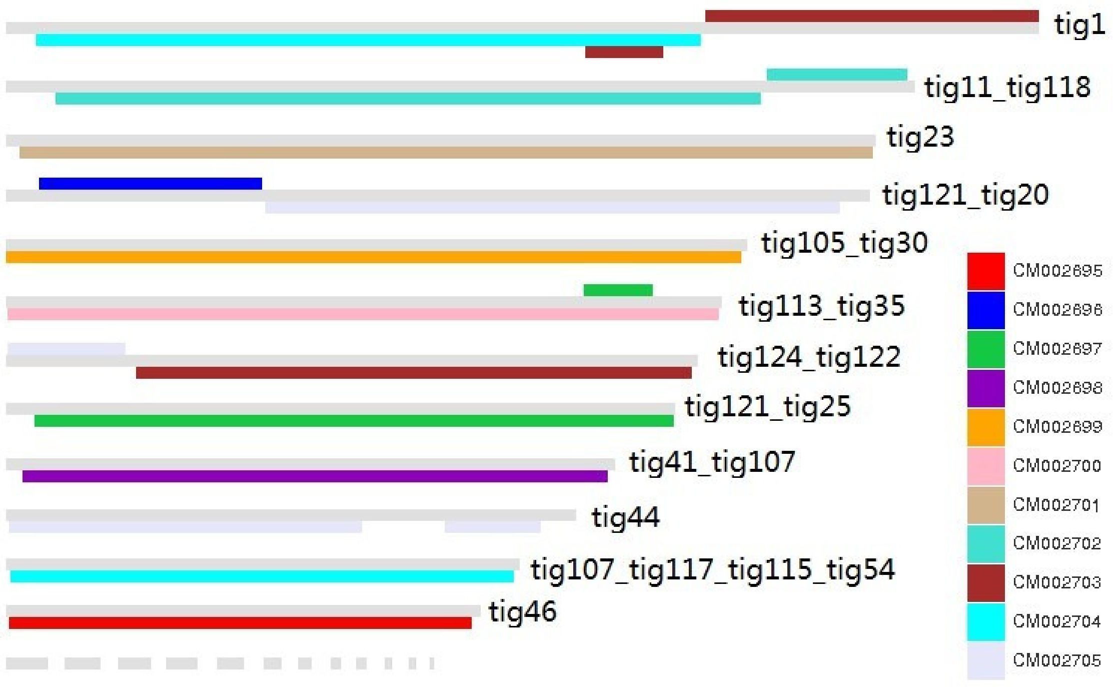
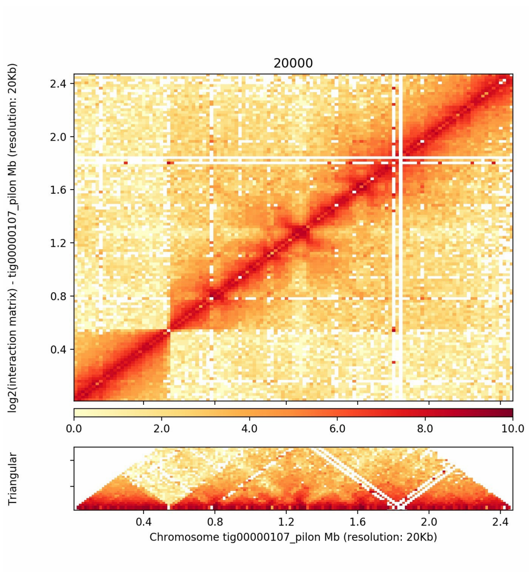
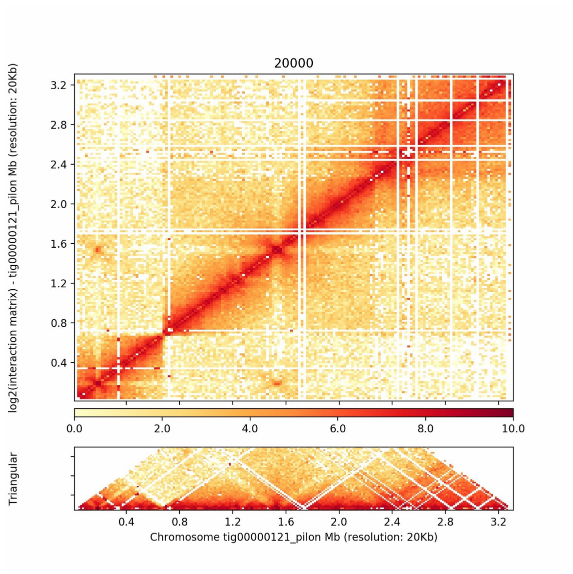


Figure 3.11: The reference genome aligned on CAST assembly. Each grey bar represents a contig in the CAST assembly, and the colored bars on it are aligned blocks from the reference genome. Each contig in the CAST assembly is named by draft contigs which form it. Small contigs without alignment are compacted.



(a) tig107



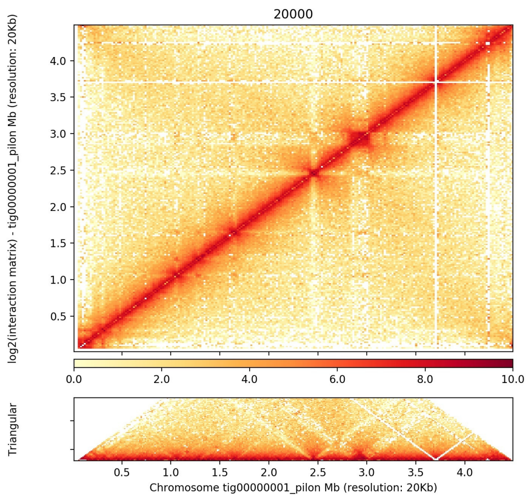
(b) tig121

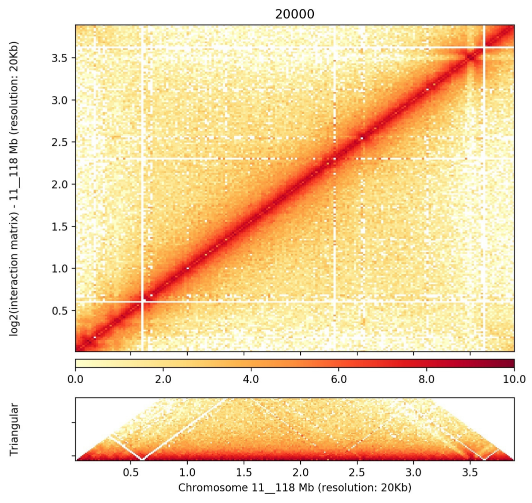
Figure 3.12: Chromatin interaction heatmaps for tig107 and tig121. Darker color indicates higher interaction. This is usually expected along the diagonal in a square heatmap, which represents interactions between close positions.

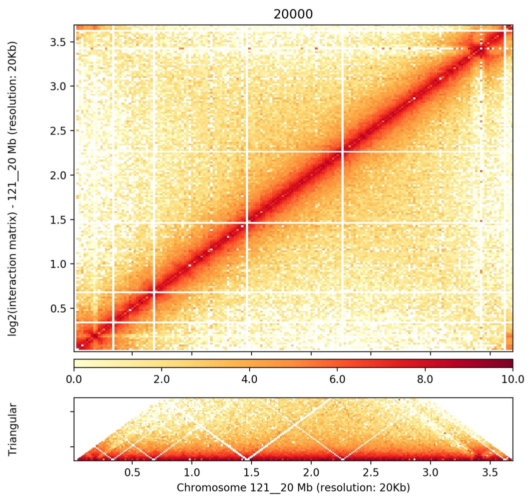
contigs' name of the CAST assembly in this figure, it can be observed that tig107 and tig121 in the draft assembly were split and merged with others. To verify whether these modifications are reasonable, Hi-C data were mapped to the CAST assembly and interactions between regions are counted by HiC-Pro [Servant et al., 2015]. Then, I visualized the interaction data on each chromosome by HiCPlotter [Akdemir and Chin, 2015]. Based on the fact that interaction frequency between regions is negatively correlated with the distance within a chromosome, the count for interaction spanning a translocation point should be much lower than neighbouring points. Thus, discontinuous coloring is expected in the heatmaps for misassembled contigs.

Figure 3.12 shows the chromatin interactions heatmap for tig107 and tig121. The misassembled points can be easily identified at ∼0.5Mb in tig107 and ∼0.7Mb in tig121. These positions are exactly where CAST split tig107 and tig121. So, these two splits are supported and consistent with Hi-C analysis.

After this split verification, I also used Hi-C data to inspect merging made by CAST. In Figure 3.11, the new contigs totally aligned to single chromosomes are considered as reliable (e.g. tig23, tig105_tig30). But there are 4 contigs aligned to more than one chromosome (i.e. tig1, tig121_tig20, tig124_tig122) or aligned to different strands of a chromosome (tig11_tig118). Figure 3.13 shows the smoothly


(a) tig1


(b) tig11_118


(c) tig121_20

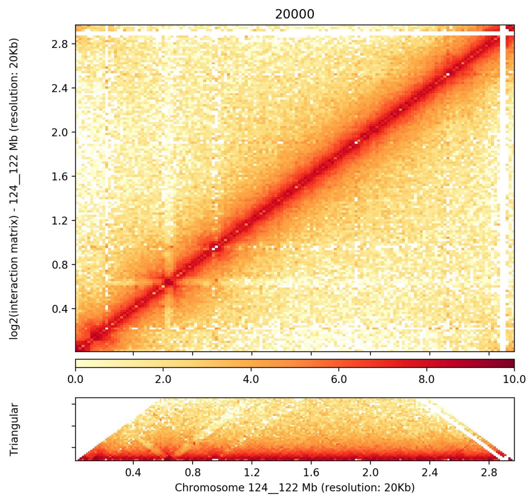
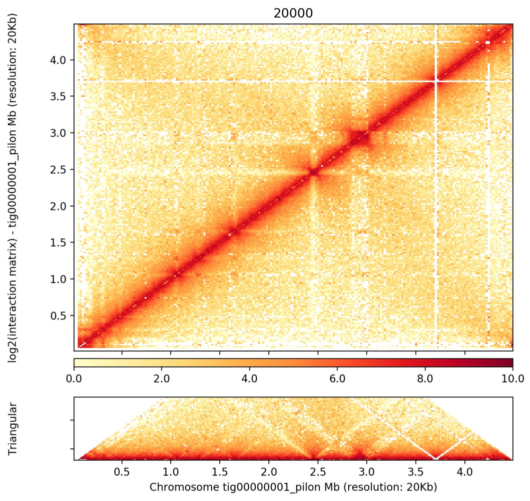
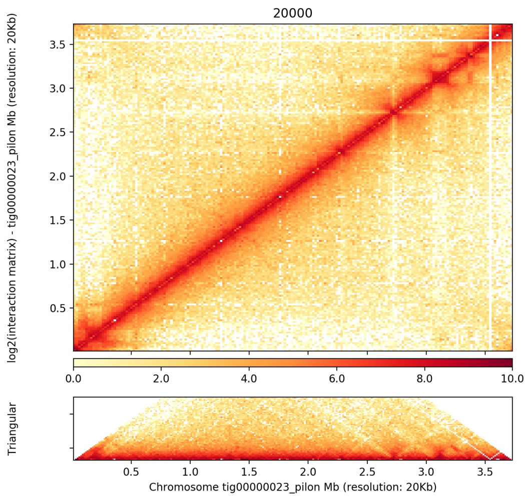

(d) tig124_122

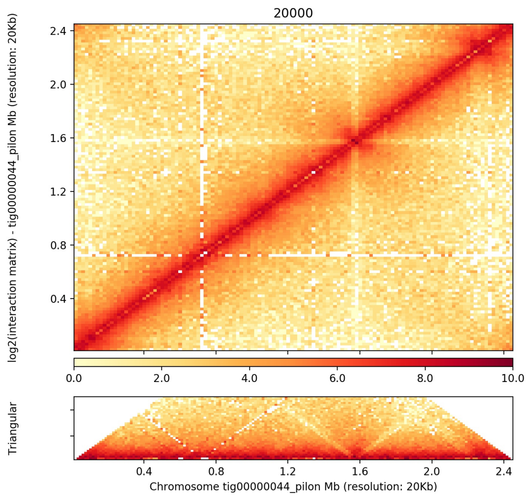
Figure 3.13: Chromatin interaction heatmaps for 4 suspected contigs. Darker color indicates higher interaction. This is usually expected along the diagonal in a square heatmap, which represents interactions between close positions.

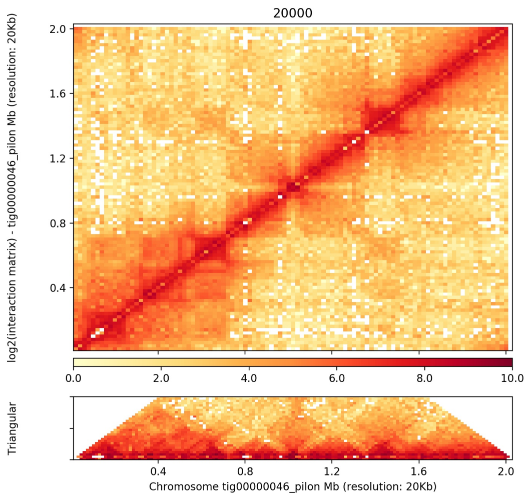
colored heatmaps of these 4 contigs. No significant break is observed in these heatmaps; thus the merging on these contigs are reliable and proper. In fact, no translocation can be observed in all contig heatmaps (see Figure 3.14).

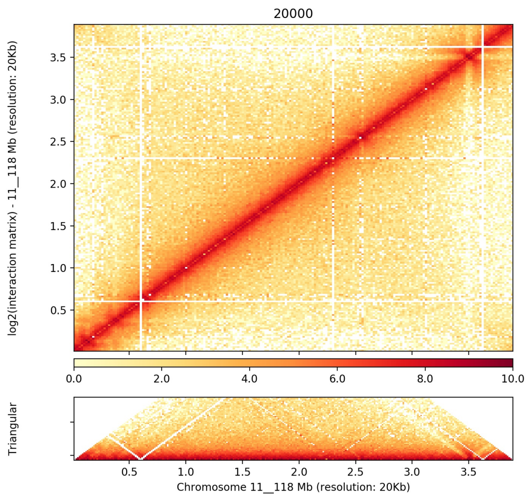
To verify the sequences lost in merging are repeats and therefore harmless, I investigate these sequences one by one as well as the overlaps used for merging. Each sequence was queried against CAST-improved assembly by BLAST. An alignment with >90% identity and >50% length of query

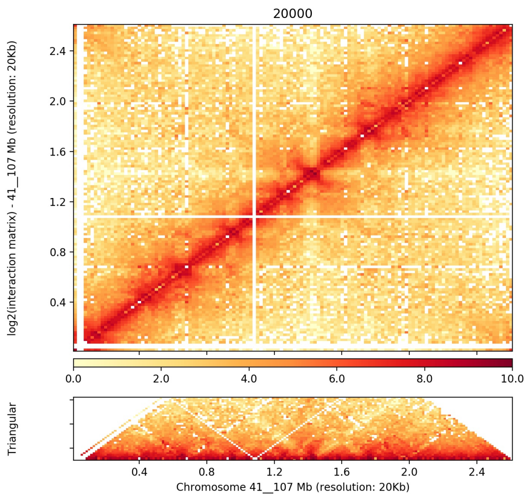

20000
log2(interaction matrix) - tig00000001_pilon Mb (resolution: 20Kb)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Chromosome tig00000001_pilon Mb (resolution: 20Kb)


20000
log2(interaction matrix) - tig00000023_pilon Mb (resolution: 20Kb)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.5 1.0 1.5 2.0 2.5 3.0 3.5
Chromosome tig00000023_pilon Mb (resolution: 20Kb)


20000
log2(interaction matrix) - tig00000044_pilon Mb (resolution: 20Kb)
2.4
2.0
1.6
1.2
0.8
0.4
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.4 0.8 1.2 1.6 2.0 2.4
Chromosome tig00000044_pilon Mb (resolution: 20Kb)


20000
log2(interaction matrix) - tig00000046_pilon Mb (resolution: 20Kb)
2.0
1.6
1.2
0.8
0.4
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.4 0.8 1.2 1.6 2.0
Chromosome tig00000046_pilon Mb (resolution: 20Kb)


20000
log2(interaction matrix) - 11__118 Mb (resolution: 20Kb)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.5 1.0 1.5 2.0 2.5 3.0 3.5
Chromosome 11__118 Mb (resolution: 20Kb)


20000
log2(interaction matrix) - 41__107 Mb (resolution: 20Kb)
2.4
2.0
1.6
1.2
0.8
0.4
0.0 2.0 4.0 6.0 8.0 10.0
Triangular
0.4 0.8 1.2 1.6 2.0 2.4
Chromosome 41__107 Mb (resolution: 20Kb)

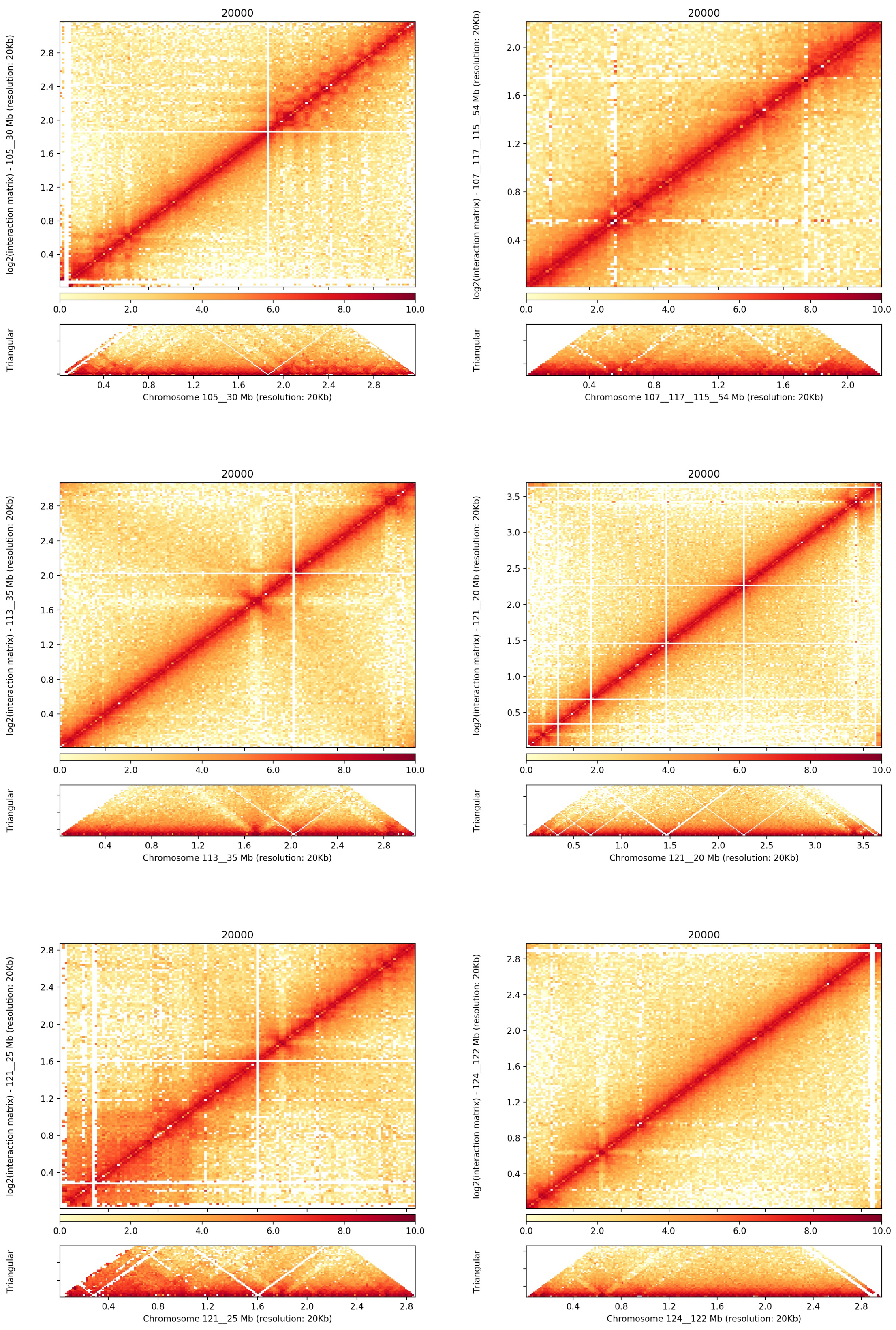


Figure 3.14: Chromatin interaction heatmaps for all contigs > 500K in the CAST-improved assembly. Darker color indicates higher interaction. This is usually expected along the diagonal in a square heatmap, which represents interactions between close positions.

Table 3.5: The length of overlap and sequence loss during merging. Left loss and right loss refer to the loss on different contigs, while overlap shows the length of overlap. The numbers in parenthesis indicate the occurrence count of the corresponding sequence in the CAST-improved assembly.

Merging	Left Loss (Occur)	Overlap (Occur)	Right Loss (Occur)
11 and 118	0	14057(2)	0
25 and 121	0	7849(18)	0
107 and 117	0	16974(16)	0
117 and 115	0	13970(2)	0
115 and 054	0	2175(18)	0
124 and 122	0	17290(13)	0
105 and 30	3984(19)	1738(48)	4687(19)
20 and 121	0	21467(2)	0
35 and 113	0	1638(19)	0

sequence was counted as an occurrence. Table 3.5 summarizes the occurrence counts. In the table, it can be seen that all sequences occur more than once. The lost sequences are definitely repeats, and their other copies still exist in the CAST-improved assembly. The overlap sequences are either too long to be covered by a single read, repeated over 10 times, or both. This result supports that CAST improves assembly by resolving and merging repeat regions with almost harmless pruning of hangover repeats.

In summary, all corrections and merging made by CAST on this dataset were supported by synteny analysis and Hi-C heatmaps. Compared with Hi-C scaffolding tools, CAST better identified misassemblies. As synteny analysis showed, CAST rearranged the draft genome correctly, and took the final step from scaffold-level assembly to chromosome-level assembly. However, the quality metrics reported by QUAST was not sufficient to reflect the significance of those rearrangements. Because misassemblies were only counted in quantity, regardless of their influence. The advantages of CAST will be more apparent with a better metric weighted by misassembly's influence. Therefore, I pursued this aspect in Chapter 2.

3.3.5 *F. velutipes* Illumina assembly

Although CAST was initially designed to improve draft assemblies obtained by long reads, I also evaluated its robustness on draft assemblies obtained by short reads. The draft assembly here is the Illumina assembly from Spades. All improved assemblies and draft assembly were evaluated against the draft Pacbio assembly (described in Section 3.3.1) instead of the downloaded reference genome, so that the difference between strain 6-3 and the reference genome can be eliminated. 3D-DNA was not included in the comparison because it could not finish within 48 hours.

Table 3.6 shows the result on Illumina assembly. Unlike the evaluation on Pacbio assembly, SALSA significantly improved contiguity metrics. Largest alignment and NGA50 were tripled and doubled, respectively. But I also noticed that the largest contig increased by more than 50 times while NG50 incredibly surged over 400 folds. These rates of increase are disproportionate for NG50 against NGA50 and the largest contig against the largest alignment. These disagreements suggest SALSA made a lot of successful merging but also much more misassemblies. This inference is supported by

Table 3.6: Improvement on *F. velutipes* Illumina assembly. For each statistic, theoretical best values are colored blue; theoretical worst values are colored red.

Assembly	Draft assembly	CAST	SALSA
# contigs	2837	2769	1695
Genome fraction (%)	90.96	90.86	90.955
Total length	35,455K	35,418K	36,026K
Total aligned length	34,929K	34,891K	34,928K
Largest contig	466K	466K	25,269K
Largest alignment	466K	466K	1,251K
NG50	60K	66K	25,269K
NGA50	60K	62K	113K
# relocations	3	24	452
# translocations	10	9	149
# inversions	0	1	111
PDR	73.26%	73.37%	38.24%

a large increase of misassembly counts as well as a significant decrease of PDR. SALSA introduced 250 translocations and inversions. Given that the number of contigs is reduced by 1142, at least 1142 contigs were merged with others, while translocations and inversions were introduced in 20% of these merging. Overall, SALSA is an aggressive scaffolding tool which improves contiguity at a non-negligible cost to correctness. In contrast, CAST is more conservative as it only slightly improved contiguity metrics but was harmless to correctness if I ignore relocations caused by repeat collapse. It even corrected 1 out of 10 translocations.

3.3.6 Run time evaluation

Without considering BLAST merging, the time complexity of CAST is $O(P(GC+GP+N^2P))$, where P, G, C, and N are progeny count, genome size, sequencing depth, and contig count respectively. The progeny count has a non-negligible influence on the run time. Thus, I evaluated CAST run time by tuning the number of progenies in *A. thaliana* dataset. I carried out all the experiments on the same computer, which has two six-core Intel Xeon E5-2620 v3 2.4GHz CPUs and 64G random-access memory, installed with the CentOS 7 operating system. Run time and the total file size are plotted against progeny count in Figure 3.15. Although run time is quadratic to progeny count in theory, it is approximately linear to total file size in practice. This is because files loading and reads preprocessing

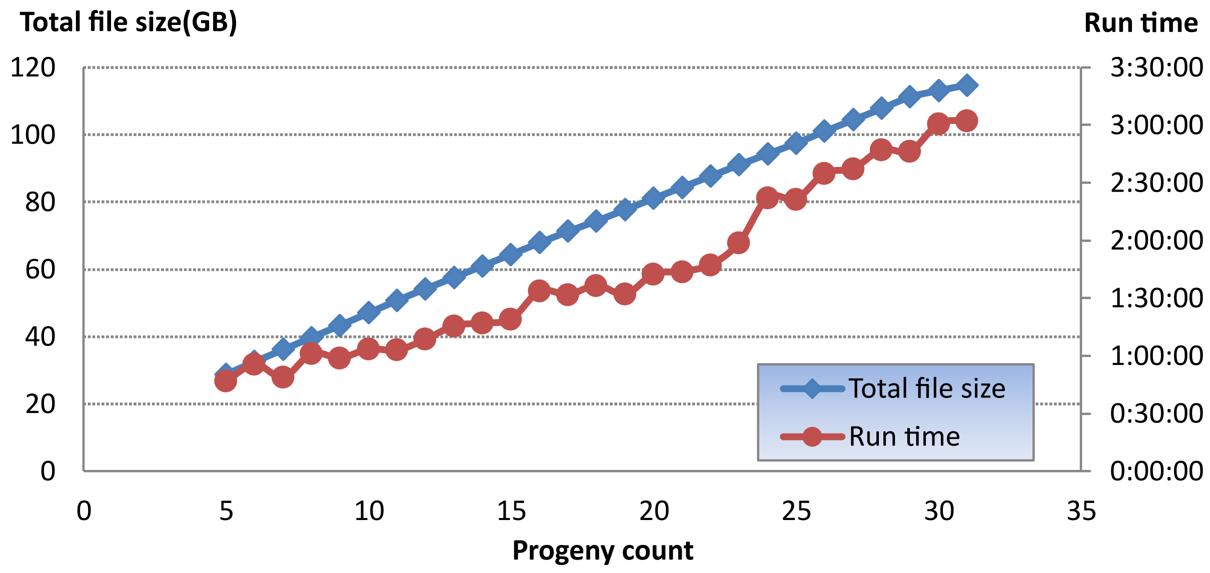


Figure 3.15: Total file size vs. run time. Total file size was tuned by using different progeny count.

dominate the run time when progeny count is not too large. In normal use, CAST only requires tens of progenies at most. So the overall run time is roughly proportional to total file size. Empirically, it processed 100GB of data within 3 hours.

3.4 Discussion

CAST is designed to improve assembly by progeny sequencing data. The idea behind CAST is to utilize haplotype in homologous recombination to span repeat regions and identify misassembly. Similar to traditional anchoring methods based on genetic map, CAST also integrates genetic information. But they differ on their backbones, as CAST uses a draft assembly as backbone to move away from dependence on construction of a genetic map. For large genomes, constructing a genetic map requires extremely long run time as well as large memory. CAST uses buffers during scan to reduce memory usage, and its run time is theoretically linear to genome size or marker size. It also requires fewer samples to achieve better improvement. Compared with Hi-C methods, CAST is more sensitive and suffers less from technical noise.

To some degree, the performance of CAST depends on a draft assembly's quality. For a draft assembly obtained by short reads or assembled by a conservative assembler, CAST will improve contiguity more than correctness as the contigs are relatively reliable. In contrast, CAST may identify more misassemblies but contribute little to contiguity for a draft assembly obtained by long reads or assembled by an aggressive assembler.

As observed from several evaluations, CAST often yields slightly shorter total length and total aligned length, compared to the draft assembly. This is because CAST occasionally introduces or enlarges repeat collapse during merging. However, repeat collapse is a common problem in genome assembly and it is almost harmless as it does not effect most downstream applications like linkage analysis. Thus, it is reasonable to trade for contiguity and correctness at some cost of repeat collapse.

Chapter 4

Concluding remarks

4.1 Conclusion

Next-generation sequencing provides a powerful tool for many research topics. In sequencing projects, a high-quality reference genome is the first step to understanding an organism. Many downstream applications (e.g. genetic variant calling, transcriptome analysis, and epigenomic analysis) all highly rely on the quality of the reference genome.

With sequencing data, people usually run a few assemblers separately, compare the assemblies produced, and then select the best one for downstream analysis. However, there are too many ways to define "best". Depending on downstream analysis, people may focus on contiguity, correctness, completeness or some combination of them. In practice, tens of metrics are used and each of them assesses assemblies on a specific aspect. In most cases, all of them are expected to be as high as possible. However, due to strategic decisions made during a genome assembly, one of them is sometimes sacrificed for another. Thus, PDR has been defined in this thesis to provide an integrative metric, accompanied by an efficient approximated implementation of it. It is informative in guiding the selection from various assemblers having their own advantages.

Based on a reasonable metric, people may want to further improve their draft assemblies. In this thesis, CAST has been proposed to improve a draft assembly by using sequencing data of a progeny population. Such data is usually available in breeding research projects. For those projects in which population sequencing is not intended, CAST is also an option to obtain a better assembly when budget is big enough. Individual sequencing has an upper bound on assembly quality due to technical limitation. CAST breaks through this by integrating genetic information from progeny and exploiting the law of genetic linkage: DNA sequences are likely to be inherited together during the meiosis phase of sexual reproduction, if they are close together on a chromosome.

4.2 Future work

PDR is theoretically defined on a perfect alignment between reference and assembly. However, in practice, it is nearly impossible to acquire the perfect alignment. The quality of alignment depends on aligners and their parameters. In Chapter 2, the fluctuation introduced by some popular aligners have been qualitatively verified to be small enough in assembly comparison. Next, the quantitative correlation between PDR and alignment statistics has also been investigated. Furthermore, a new PDR implementation may be considered to avoid influence from alignment.

Although our experiment results have showed the merit of CAST, there are still some other aspects to be investigated or verified. For example, sequencing cost could constraint its wide adoption in practice. For projects which focus on plant breeding and genetic topics, progeny sequencing are usually intended for downstream analysis and thus also available during genome assembly. In Chapter 3, it has been verified that CAST improves assembly quality by integrating genetic information. But for some other projects where progeny sequencing is not necessary, CAST introduces extra efforts and cost. In such cases, people may want to know whether or in what degree CAST is better than other assembly strategies. So, further investigation is required to show whether CAST is able to outperform other strategies under the same amount of budget.

Theoretically, CAST is capable of handling polyploidy data. However, such species are usually plants with large genome. Because of limited budget, it is unpractical to sequence tens of plant progenies by high sequencing depth. Therefore, CAST has not been tested on polyploidy data. With decreasing sequencing cost, it is promising to obtain suitable data for this purpose within a few years. Then, CAST will be verified on some polyploidy data.

Bibliography

Kadir Caner Akdemir and Lynda Chin. Hicplotter integrates genomic data with interaction matrices. Genome biology, 16(1):198, 2015.

Hind Alhakami, Hamid Mirebrahim, and Stefano Lonardi. A comparative evaluation of genome assembly reconciliation tools. Genome biology, 18(1):93, 2017.

Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. Journal of computational biology, 19(5):455–477, 2012.

Keith R Bradnam, Joseph N Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi Chou, Jacques Corbeil, Cristian Del Fabbro, T Roderick Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno A Fonseca, Ganeshkumar Ganapathy, Richard A Gibbs, Sante Gnerre, Élénie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph B Hiatt, Isaac Y Ho, Jason Howard, Martin Hunt, Shaun D Jackman, David B Jaffe, Erich D Jarvis, Huaiyang Jiang, Sergey Kazakov, Paul J Kersey, Jacob O Kitzman, James R Knight, Sergey Koren, Tak-Wah Lam, Dominique Lavenier, François Laviolette, Yingrui Li, Zhenyu Li, Binghang Liu, Yue Liu, Ruibang Luo, Iain MacCallum, Matthew D MacManes, Nicolas Maillet, Sergey Melnikov, Delphine Naquin, Zemin Ning, Thomas D Otto, Benedict Paten, Octávio S Paulo, Adam M Phillippy, Francisco Pina-Martins, Michael Place, Dariusz Przybylski, Xiang Qin, Carson Qu, Filipe J Ribeiro, Stephen Richards, Daniel S Rokhsar, J Graham Ruby, Simone Scalabrin, Michael C Schatz, David C Schwartz, Alexey Sergushichev, Ted Sharpe, Timothy I Shaw, Jay Shendure, Yujian Shi, Jared T Simpson, Henry Song, Fedor Tsarev, Francesco Vezzi, Riccardo Vicedomini, Bruno M Vieira, Jun Wang, Kim C Worley, Shuangye Yin, Siu-Ming Yiu, Jianying Yuan, Guojie Zhang, Hao Zhang, Shiguo Zhou, and Ian F Korf. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. GigaScience, 2(1):2047–217X–2–10,

2013. doi: 10.1186/2047-217X-2-10. URL http://dx.doi.org/10.1186/2047-217X-2-10.

Joshua N Burton, Andrew Adey, Rupali P Patwardhan, Ruolan Qiu, Jacob O Kitzman, and Jay Shendure. Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. Nature biotechnology, 31(12):1119, 2013.

Christina J Castro and Terry Fei Fan Ng. U50: A new metric for measuring assembly output based on non-overlapping, target-specific contigs. Journal of Computational Biology, 24(11):1071–1080, 2017.

Olga Dudchenko, Sanjit S Batra, Arina D Omer, Sarah K Nyquist, Marie Hoeger, Neva C Durand, Muhammad S Shamim, Ido Machol, Eric S Lander, Aviva Presser Aiden, et al. De novo assembly of the aedes aegypti genome using hi-c yields chromosome-length scaffolds. Science, 356(6333): 92–95, 2017.

Dent Earl, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken Yu, Vince Buffalo, Daniel R Zerbino, Mark Diekhans, et al. Assemblathon 1: a competitive assessment of de novo short read assembly methods. Genome research, 21(12):2224–2241, 2011.

Janna L Fierst. Using linkage maps to correct and scaffold de novo genome assemblies: methods, challenges, and computational tools. Frontiers in genetics, 6:220, 2015.

Mohammadreza Ghodsi, Christopher M Hill, Irina Astrovskaya, Henry Lin, Dan D Sommer, Sergey Koren, and Mihai Pop. De novo likelihood-based measures for comparing genome assemblies. BMC research notes, 6(1):334, 2013.

Jay Ghurye, Mihai Pop, Sergey Koren, Derek Bickhart, and Chen-Shan Chin. Scaffolding of long read assemblies using long range contact information. BMC genomics, 18(1):527, 2017.

Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. Quast: quality assessment tool for genome assemblies. Bioinformatics, 29(8):1072–1075, 2013.

Matthew W Hahn, Simo V Zhang, and Leonie C Moyle. Sequencing, assembling, and correcting draft genomes using recombinant populations. G3: Genes, Genomes, Genetics, pages g3–114, 2014.

Niina Haiminen, David N Kuhn, Laxmi Parida, and Isidore Rigoutsos. Evaluation of methods for de novo genome assembly from high-throughput sequencing reads reveals dependencies that affect the quality of the results. PloS one, 6(9):e24182, 2011.

Martin Hunt, Taisei Kikuchi, Mandy Sanders, Chris Newbold, Matthew Berriman, and Thomas D Otto. Reapr: a universal tool for genome assembly evaluation. Genome biology, 14(5):R47, 2013.

Vasanthan Jayakumar and Yasubumi Sakakibara. Comprehensive evaluation of non-hybrid genome assembly tools for third-generation pacbio long-read sequence data. Briefings in Bioinformatics, page bbx147, 2017. doi: 10.1093/bib/bbx147. URL http://dx.doi.org/10.1093/bib/bbx147.

Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome research, 27(5):722–736, 2017.

Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. Bioinformatics, 32(14):2103–2110, 2016.

Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. Bioinformatics, 25(14):1754–1760, 2009.

Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. Gigascience, 1(1):18, 2012.

Veli Mäkinen, Leena Salmela, and Johannes Ylinen. Normalized n50 assembly metric using gap-restricted co-linear chaining. BMC bioinformatics, 13(1):255, 2012.

Guillaume Marçais, Arthur L Delcher, Adam M Phillippy, Rachel Coston, Steven L Salzberg, and Aleksey Zimin. Mummer4: a fast and versatile genome alignment system. PLoS computational biology, 14(1):e1005944, 2018.

Martin Mascher, Gary J Muehlbauer, Daniel S Rokhsar, Jarrod Chapman, Jeremy Schmutz, Kerrie Barry, María Muñoz-Amatriaín, Timothy J Close, Roger P Wise, Alan H Schulman, et al. Anchoring and ordering ngs contig assemblies by population sequencing (popseq). The Plant Journal, 76(4): 718–727, 2013.

Alla Mikheenko, Andrey Prjibelski, Vladislav Saveliev, Dmitry Antipov, and Alexey Gurevich. Versatile genome assembly evaluation with quast-lg. Bioinformatics, 34(13):i142–i150, 2018.

Eugene W Myers, Granger G Sutton, Art L Delcher, Ian M Dew, Dan P Fasulo, Michael J Flanigan, Saul A Kravitz, Clark M Mobarry, Knut HJ Reinert, Karin A Remington, et al. A whole-genome assembly of drosophila. Science, 287(5461):2196–2204, 2000.

Giuseppe Narzisi and Bud Mishra. Comparing de novo genome assembly: the long and short of it. PloS one, 6(4):e19175, 2011.

M Neto, G Skorski, D Thevenot, and E Loukiadis. Optical maps: methodology and applications in microbiology. EuroReference, 5:38–46, 2011.

Carlos W Nossa, Paul Havlak, Jia-Xing Yue, Jie Lv, Kimberly Y Vincent, H Jane Brockmann, and Nicholas H Putnam. Joint assembly and genetic mapping of the atlantic horseshoe crab genome reveals ancient whole genome duplication. GigaScience, 3(1):9, 2014.

Genis Parra, Keith Bradnam, and Ian Korf. Cegma: a pipeline to accurately annotate core genes in eukaryotic genomes. Bioinformatics, 23(9):1061–1067, 2007.

Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An eulerian path approach to dna fragment assembly. Proceedings of the national academy of sciences, 98(17):9748–9753, 2001.

Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J Treangen, Michael C Schatz, Arthur L Delcher, Michael Roberts, et al. Gage: A critical evaluation of genome assemblies and assembly algorithms. Genome research, 22(3):557–567, 2012.

Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. Proceedings of the national academy of sciences, 74(12):5463–5467, 1977.

Nicolas Servant, Nelle Varoquaux, Bryan R Lajoie, Eric Viara, Chong-Jian Chen, Jean-Philippe Vert, Edith Heard, Job Dekker, and Emmanuel Barillot. Hic-pro: an optimized and flexible pipeline for hi-c data processing. Genome biology, 16(1):259, 2015.

Felipe A Simão, Robert M Waterhouse, Panagiotis Ioannidis, Evgenia V Kriventseva, and Evgeny M Zdobnov. Busco: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics, 31(19):3210–3212, 2015.

Carol Soderlund, William Nelson, Austin Shoemaker, and Andrew Paterson. Symap: A system for discovering and viewing syntenic regions of fpc maps. Genome research, 16(9):1159–1168, 2006.

Ivan Sović, Krešimir Križanović, Karolj Skala, and Mile Šikić. Evaluation of hybrid and non-hybrid methods for de novo assembly of nanopore reads. Bioinformatics, 32(17):2582–2589, 2016.

Bruce J Walker, Thomas Abeel, Terrance Shea, Margaret Priest, Amr Abouelliel, Sharadha Sakthikumar, Christina A Cuomo, Qiandong Zeng, Jennifer Wortman, Sarah K Young, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. PloS one, 9(11):e112963, 2014.

Kris A Wetterstrand. Dna sequencing costs: data from the nhgri genome sequencing program (gsp), 2013.

Sihai Yang, Long Wang, Ju Huang, Xiaohui Zhang, Yang Yuan, Jian-Qun Chen, Laurence D Hurst, and Dacheng Tian. Parent–progeny sequencing indicates higher mutation rates in heterozygotes. Nature, 523(7561):463, 2015.

Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome research, 18(5):821–829, 2008.