KTH Computer Science
and Communication

Algorithms and statistical models for scaffolding contig assemblies and detecting structural variants using read pair data

KRISTOFFER SAHLIN

Doctoral Thesis
Stockholm, Sweden 2015

TRITA-CSC-A-2015:14
ISSN-1653-5723 KTH School of Computer Science and Communication
ISRN-KTH/CSC/A–15/14-SE SE-100 44 Stockholm
ISBN 978-91-7595-677-0 SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i datalogi torsdagen den 1 oktober 2015 klockan 10.00 i föreläsningssal Atrium, Nobels väg 12B, Stockholm.

Tryck: Universitetsservice US AB

Abstract

Advances in throughput from *Next Generation Sequencing* (NGS) methods has provided new ways to study molecular biology. The increased amount of data enables genome wide scale studies of structural variation, transcription, translation and genome composition. Not only is the scale of each experiment large; lowered cost and faster turn-around has also increased the frequency with which new experiments are conducted. With the data growth comes an increase in demand for efficient and robust algorithms — this is a great computational challenge. The design of computationally efficient algorithms are crucial to cope with the amount of data and it is relatively easy to verify an efficient algorithm by runtime and memory consumption. However, as NGS data comes with several artifacts together with the size the difficulty lies in verifying that the algorithm gives accurate results and are robust to different data sets.

This thesis focuses on modeling assumptions of mate-pair and paired-end reads when scaffolding contig assemblies or detecting variants. Both genome assembly and structural variation are difficult problems, partly because of a computationally complex nature of the problems, but also due to various noise and artifacts in input data. Constructing methods that addresses all artifacts and parameters in data is difficult, if not impossible, and end-to-end pipelines often come with several simplifications. Instead of tackling these difficult problems all at once, a large part of this thesis concentrates on smaller problems around scaffolding and structural variation detection. By identifying and modeling parts of the problem where simplifications has been made in other algorithms, we obtain an improved solution to the corresponding full problem.

The first paper shows an improved model to estimate gap sizes, hence contig placement, in the scaffolding problem. The second paper introduces a new scaffolder to scaffold large complex genomes and the third paper extends the scaffolding method to account for paired-end-contamination in mate-pair libraries. The fourth paper investigates detection of structural variants using fragment length information and corrects a commonly assumed null-hypothesis distribution used to detect structural variants.

Sammanfattning

Utvecklingen av sekvenseringsmetoder som kan producera mycket data på kort tid har banat nya vägar för forskning inom molekylärbiologi. Den ökade mängden data från dessa metoder möjliggör studier på en organisms hela arvsmassa i kontrast till tidigare metoder som begränsade studier till specifika regioner. Global analys av till exempel genomstruktur och strukturell variation kan underlätta bland annat evolutionsforskning samt utveckling av vaccin och personlig medicin.

Med denna datatillväxt kommer en ökad efterfrågan på effektiva och robusta algoritmer. Mängden data genererat från varje experiment är inte bara stor; sänkt kostnad och minskad tidsåtgång har också ökat frekvensen på antalet experiment som genomförs. Utformningen av beräkningseffektiva algoritmer är därför avgörande för att klara av mängden data. Det är relativt lätt att verifiera att en algorithm är effektiv genom att mäta körtid och minnesförbrukning. Det är emellertid svårare är att verifiera att algoritmen ger korrekta, eller nära korrekta, resultat och är robusta för olika datamängder. Denna svårighet kommer från storleken på datamängden tillsammans med de många och varierande artefakter som data kommer med.

Denna avhandling fokuserar på modelleringsantaganden för sekvenseringsmetoder som producerar parade läsningar för att länka contiger (contig scaffolding) eller upptäcka strukturella varianter. Både scaffolding och att hitta strukturella variationer är svåra problem, delvis på grund av den beräkningsmässiga komplexitet som dessa problem har, men också på grund av den mängd brus och artefakter som ofta finns i data. Att konstruera metoder som modellerar alla artefakter och brus i data är svårt, om inte omöjligt, och existerande helhetslösningar kommer ofta med flera förenklingar. I stället för att angripa dessa svåra problem på en gång, så fokuserar en stor del av denna avhandling på mindre problem runt scaffolding och strukturell variation. Genom att identifiera och modellera delar av problemet där förenklingar har gjorts i andra algoritmer, får vi en förbättrad lösning till de stora problemen.

Den första artikeln introducerar en förbättrad modell för att beräkna gap storlekar därmed contig placering i scaffolding. Den andra artikeln presenterar ett ny algorithm för att scaffolda stora genom snabbt. Vi utvidgar scaffolder metoden i det tredje arbetet med att modellera en vanlig artefact i sekvenseringsdata med heltalsprogrammering. Det fjärde arbetet korrigerar en vanlig antagen modell och således nollhypotes för att uptäcka structuralle varianter med hjälp av parade läsningar.

List of publications

I: Sahlin K, Street N, Lundeberg J, Arvestad L.
Improved gap size estimation for scaffolding algorithms.
Bioinformatics. 2012;28(17):2215-22.

II: Sahlin K, Vezzi F, Nystedt B, Lundeberg J, Arvestad, L.
BESST - Efficient scaffolding of large fragmented assemblies
BMC Bioinformatics. 2014;15(1):281.

III: Sahlin K, Chikhi R, Arvestad, L.
Genome scaffolding with PE-contaminated mate-pair libraries.
Manuscript, under review (accepted to WABI 2015). bioRxiv doi: 10.1101/025650

IV: Sahlin K, Frånberg M, Arvestad, L.
Correcting bias from stochastic insert size in read pair data — applications to structural variation detection and genome assembly
Manuscript. bioRxiv doi: 10.1101/023929

Acknowledgements

When I look back on the past 5 years and contemplate the roller coaster journey I have been on, I can't help but have a smile on my face. This journey in front of the computer has given me the tools to become an expert in the areas I have been researching albeit with an added hunch in my back and pain in my shoulders.

I would like to thank my supervisor Lars Arvestad who believed that I was PhD material although I didn't know what "implement" meant during the interview. It must have been my very original answer "I would google it" that caught your attention when asked how I would solve a particular programming problem. Your relaxed approach to supervision and excellent teaching has made the years easier to plow through. You have given me great feedback and inspired ideas for problem solving and trouble shooting as well as providing mental support when p-values from null distribution were non-uniform. I'm very thankful for the freedom I have been given to pursue my own projects — it has been crucial to develop my independence in research. I am thankful to have done my PhD around experienced peers such as Pekka Parvalainen, Hossein Farahani and Joel Sjöstrand. You guys prepared me for what PhD work really involves. Jens Lagergren, your frank and no nonsense academic advice has also given me a better understanding of academic life. A special thanks must be given to Mattias Frånberg for our collaboration in science, fitness and everyday life hacking. Thank you for taking the time to answer a million and one questions about programming and nowadays also statistics! You have inspired me in many ways and since all we know science and fitness these days, discussions are always easy. Måns and Ino are two other cool characters with whom I've learned a lot from and have had the pleasure to spend time with outside work.

Working in a buzzing environment (well, in comparison to other math/CS departments) like scilifelab has made the ups and downs of PhD life much easier. Thanks to all the awesome colleagues who joined me for a bit of banter over fika; the brothas Owais and Ikram whom I owe a fair amount of sweets to — the discussions (and cookies) are always a delight. My dear group members Mehmood and Hashim who are also fighting for their Phd's — grumbling with you about this last shitty period helps me through mine. Thanks too all the other cool people who I have met at ScilifeLab: Auwn, Viktor, Erik, Yrin, Linus, Luminita, Bengt, Lukas and Maria. It has been a pleasure fika with you.

During my PhD I have had the opportunity to travel to conferences, seminars, and workshops all over the world where I have partied hard with indecent creatures such as Johan "no sleep" Reimegård, Daniel Edsgärd and Mattias Frånberg — thanks for all the fun at the ISMB conferences. Some work and networking have also been on the agenda. Through these experiences I have met talented people in various stages of their careers and this has impacted me and my approach to research: Alexandru Tomescu, Rayan Chikhi, Leena Salmela, Paul Medvedev, Veli Mäkinen and Pall Melstedt are just a few of them. Special thanks to Alex and Rayan who I have collaborated closely with. The coding weeks, and beers that I shared with Alex really showed the essence of science collaboration.

Work is just a fraction of what defines me; thanks to mom for great support and encouragement throughout the years. During my PhD I was lucky to meet my best friend Carly. Thank you for giving me the (tough) love, support and energy through the difficult times and always providing fresh optimism. I especially appreciate the evenings spent on the couch in sweatpants swearing over science and the goddamn lack of results over take home food. Thank you for also proof reading my papers, thesis and even poking around in the acknowledgements ;)

Finally, these mentally though years wouldn't have been as fun if I hadn't balanced them with some physically requiring hobbies. During my years as a PhD student, I have been a sucker for dancing; thanks to Mehmood, Joel and Mattias for supporting me in competition, and to Ino the one and only science slacker[1] for joining me practicing popping at the local joint. I went on to become the next Haile Gebreselassie; thank to Mattias, Viktor Granholm and Daniel Edsgärd, Erik Sjölund and Joel Sjöstrand for our incline intervals and running competitions in Hagaparken. I finally went on with the mindset that gymnastics and calisthenics is the optimal form of training, and this is where I'm currently at. Thanks to Måns and Mattias for keeping the hype and competition alive!

[1] dubbed by Joel

Contents

Chapter 1

Introduction

This thesis focuses on two bioinformatic problems — *scaffolding* and *structural variant detection* — that on a high level both involves inferring genomic structure. Although these problems are not bound to specific input, we will in this thesis assume *sequencing data* as input. Chapter 1 gives an overview of what sequencing data is and why it exists and introduces *sequence assembly* which leads the foundation for work presented in this thesis. Chapter 2 introduces the *scaffolding problem* and gives an overview of previous and current work on this problem. Limitations and issues with current work as well as potential future work on the scaffolding problem is discussed in Chapter 3. Chapter 4 switches topic and gives a lightweight introduction to the problem of detecting *structural variation.* Finally, Chapter 5 briefly summarizes the contents of the included papers and Chapter 6 concludes the thesis.

1.1 Sequencing

The *genome* is an organism's complete set of DNA, including both genes and non-coding sequences. As genes only constitute a fraction of the DNA in an organism, studying the complete genome sequence can give more comprehensive information about the organism's function, evolution, and interaction with bacteria or viruses. Having the complete genome sequence of an organism facilitates research in, *e.g.*,; medicine to develop of vaccines, antibiotics and personalized medical treatments; agricultural research such as resistance to insects and parasites; and evolution. These benefits together with new sequencing techniques facilitating genome wide scale analysis has led to a increased interest in producing complete genomic sequences of organisms.

However, there is no technology yet that can provide us with the complete genomic sequence. Current sequencing technologies output only fragments, referred to as *reads*, that constitutes a small piece of sequence from the genome. The sequencing methods that we will focus on in this thesis, commonly referred to as *Next*

Generation Sequencing (NGS) produces millions to billions of reads from an organism, where the reads may overlap each other. NGS methods can also produce *read pairs*. Read pairs are two reads that are separated by a distance that is approximately known on the genome. The distance between two fragments is referred to as the *fragment length* (sometimes also *insert size*) and the fragment length of read pairs produced in a NGS experiment will vary around a target fragment length set in the lab. The fragment length of all read pairs in an NGS experiment gives rise to a *fragment length distribution*. The work in this thesis concentrates on applications using read pairs as input, where the fragment length distribution is in focus.

1.2 Sequence assembly

> *"Imagine several copies of a book cut by scissors into 10 million small pieces. Each copy is cut in an individual way so that a piece from one copy may overlap a piece from another copy. Assuming that 1 million pieces are lost and the remaining 9 million are splashed with ink, try recover the original text. After doing this you'll get a feeling of what the DNA sequencing problem is like."*
>
> — Pavel Pevzner, *Computational Molecular Biology – An algorithmic approach*

The total length of read sequences produced from an NGS experiment is c times longer than the genome of the organism, where c is called the *sequencing coverage*, *i.e.*, the average number of times a position in the genome occurs in the reads. The position of each read on the genome is unknown to us, but if $c > 1$, it provides us with the fact that there exist reads that share sequence on the genome. Reads that overlap are found by investigating the sequence similarity. If two reads share a sequence, they can be merged to form a longer contiguous sequence. Such a contiguous sequence made from several merged reads is called *contig*. In practice c is usually between 10 to 200 in NGS sequence assembly projects[1] to ensure that most positions on the genome is covered and that in each position there exists reads that are overlapping enough for the overlap to be unique on the genome — allowing unambiguous merging of the reads. Creating as long and correct contigs as possible by finding overlapping reads is the core problem in sequence assembly. As we will solely focus sequence data from whole genomes we will refer to the problem of sequencing assembly of a genome as *genome assembly*. This problem is difficult due to

[1] The sufficient coverage needed to have enough overlap of reads depends on read length — longer read length requires less coverage. What "enough" is depends on the structure of the genome and its repetitive sequence lengths.

- **Data size:** The amount of data requires that the method of finding overlaps is algorithmically efficient. The simple solution to compare all possible pairs of reads is too slow.

- **Repeats:** A repeat is a sequence that occurs more than once in the genome. The sequence of a genome has more frequent, and longer, stretches of repetitive sequence compared to a randomly generated string of the same size with a four letter alphabet. Reads that contain sequence from repetitive regions will therefore share this sequence with reads from other copies of the repeat. If the repeat is frequent this will create large clusters of reads that appear to overlap, with only some of them truly overlapping on the genome.

- **Read errors:** Sequencing technologies have artifacts, they can modify the sequence by substituting, inserting, or deleting bases in the reads. This will break the perfect matching of true overlapping reads, and may even, although not as usual, create false overlaps of non overlapping reads.

- **Uneven coverage:** NGS sequencing artifacts gives rise to a non-uniform probability over the genome from which a read is generated. The probability distribution is difficult, if not impossible, to model and depends on a number of factors such as the content of nucleotides Guanine and Cytosine (GC-content), library preparation, amount of DNA available for sequencing, and amplification.

Problem representation

The genome assembly problem is usually represented as a graph where nodes represent sequences in reads and edges describes a relationship between two sequences, *e.g.*, similarity. The two most commonly used graph representations are overlap graphs and De Bruijn graphs (DBG).

Overlap graphs

In overlap graphs, reads are nodes and edges are constructed by overlapping regions in reads. The overlaps does not need to have identical sequences as the reads can contain errors. The main bottleneck with this representation is the computational time needed to construct such a representation, where the speed depends on the allowed difference between overlaps. The naive approach to compare n reads of length m pairwise with, *e.g.*, Smith-Waterman has a runtime complexity of $O(n^2m^2)$ — m^2 for the Smith-Waterman algorithm and $\binom{n}{2}$ for the number of pairwise comparisons. For n around a billion and m around 100, sophisticated data structures with sub quadratic construction time are needed. Although there has been work on such constructions [74, 104, 53], most methods use De Bruijn graphs for NGS data today.

De Bruijn graphs

Let a k-mer denote a string with k letters. A DBG of order k is a directed graph where a node represents a k-mer and an edge from node u to node v represents an overlap of the $k-1$ suffix of u and the $k-1$ prefix of v. For example, two 3-mers ACT and CTG share the edge CT. In genome assembly the DBG is constructed from k-mers of reads. Each k-mer constitutes k consecutive bases in the read, which makes read of length r have a multi set of $r-k+1$ k-mers[2] (where k-mers are not necessarily unique).

A DBG is constructed by adding nodes for each k-mer in the reads, and edges for overlaps of prefix or suffix of the $k-1$ base pairs in a k-mer, see Figure 1.1 for example constructions. Note that the construction step can be performed in $O(nm)$ time using a hash table as we only need to look if the k-mer and its $k-1$ suffix and prefix is present or not. This is a major scale down from trying all possible overlaps between reads. The downside of this construction is that we loose information of overlaps longer than $k-1$ base pairs as only substrings of length k is present in the graph.

The genome assembly problem under the DB graph representation is to find a set of paths in the graph such that the contiguous sequence (contig) spelled by consecutive k-mers in each path is present in the genome. This graph structure is used by a majority of the genome assemblers today, *e.g.*, in [4, 60, 127, 105, 20].

Reconstructing the genome from a DBG

As data is represented as a graph, a natural approach is to turn to a graph traversal algorithm that visits all vertices (Hamiltonian path) or edges (Eulerian walk) in G. Assume that error free reads with length $> k$ cover all bases of the genome and each consecutive pair of reads on the genome overlaps at least k bases and no strings of more than $k-1$ bases occurs more than once in the genome. Then there exist one unique linear simple path in G and the genome is spelled by the letters in the path. This never happens in practice. The problem of perfect reconstruction gets complex fast. [13] (based on Ukkonen's condition [111]) showed that a *unique* perfect reconstruction is not possible as long as G contains interleaved or triple repeats. A triple repeat is a repetitive sequence that occurs more than two times and interleaved repeats means that at least one repetitive sequence A occurs between two repetitive sequences B, B' different from A. These repeat structures almost always occurs in practice with the reads lengths of NGS methods. But the uniqueness of a reconstruction is not the only problem. Sequencing artifacts impedes finding even a single reconstruction as read errors creates spurious paths and coverge bias gives low-coverage or unsequenced regions of the genome — breaking the graph.

DBG assemblers therefore use heuristics to traverse the graph, removing nodes and edges based on properties of the data and using the abundance of k-mers and paired reads to chose what paths to traverse and when to break the graph. The

[2] Ignoring the reverse complement of a read.

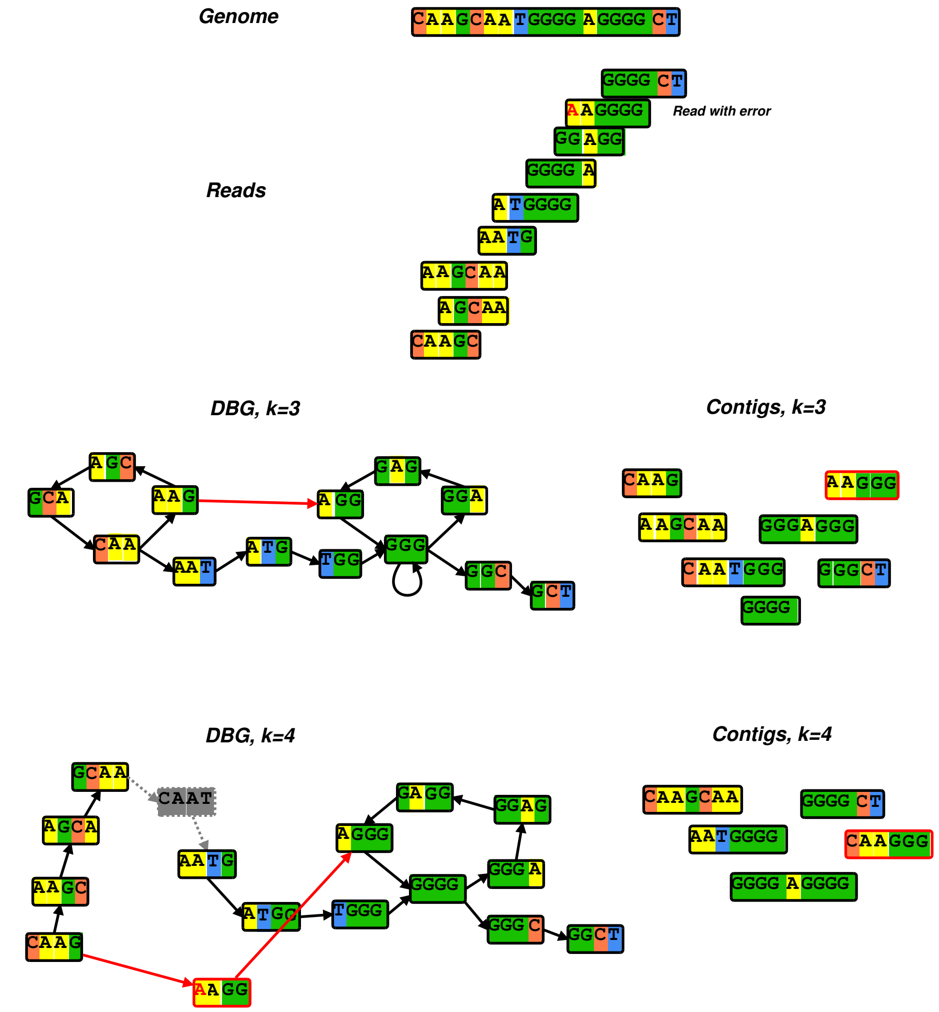


Figure 1.1: De Bruijn graph construction and contig assembly of reads from a sequence. One of the reads countains an error and it affects the features of the DBG. Contigs are formed by starting at all nodes with no, or more than one, incoming edges and stopping at all nodes with no, or more than one, outgoing edges. The resulting sets of contigs differs with k.

basic strategy is to find as long paths as possible. A simplistic example to obtain contigs is shown in Figure 1.1 by starting at all nodes with no, or more than one, incoming edges and stopping at all nodes with no, or more than one, outgoing edges.

Chapter 2

Scaffolding

2.1 Introduction

Scaffolding is the problem of ordering and orienting contigs in such a way that their arrangement matches the arrangement they have on the genome. A *scaffold* denotes a set of contigs that are positioned, ordered and oriented. We assume that contigs are placed in a linear order within a scaffold[1]. The positions assigned to contigs may induce *gaps* between contigs representing unknown sequence.

The most commonly used source of information for scaffolding contigs is read pairs. A read pair consists of two reads (also referred to as *mates*) that have distance x between them, called *fragment length.* Millions to billions of read pairs are produced in a single NGS experiment and the set of all read pairs is called the *read pair library.* In an experiment, a target size is set for the fragment length of the library, but it cannot be controlled perfectly. Therefore, x varies between read pairs in library and the distance for any read pair is known up to the distribution $f(x)$. Read pairs that have mates mapping to different contigs provides information to arrange the contigs into scaffolds. In practice, scaffolding can be applied directly on the DBG/string graph or as a separate step in a genome assembly pipeline after creating contigs. We will discuss scaffolding as a problem separated from the assembly step assuming contigs and read pairs as input. This problem can be (and has so far only been) formulated as a graph problem with contigs as nodes and mates mapping to different contigs as edges between nodes. Depending on the algorithm and implementation, the graph could be either a directed graph with one node per contig or an undirected graph with two nodes per contig.

[1] This is not always biologically sound as two or more contigs might occupy the same genomic positions due to polyploid organisms with high heterozygozity, or contigs can overlap due to, *e.g.*, assembly errors.

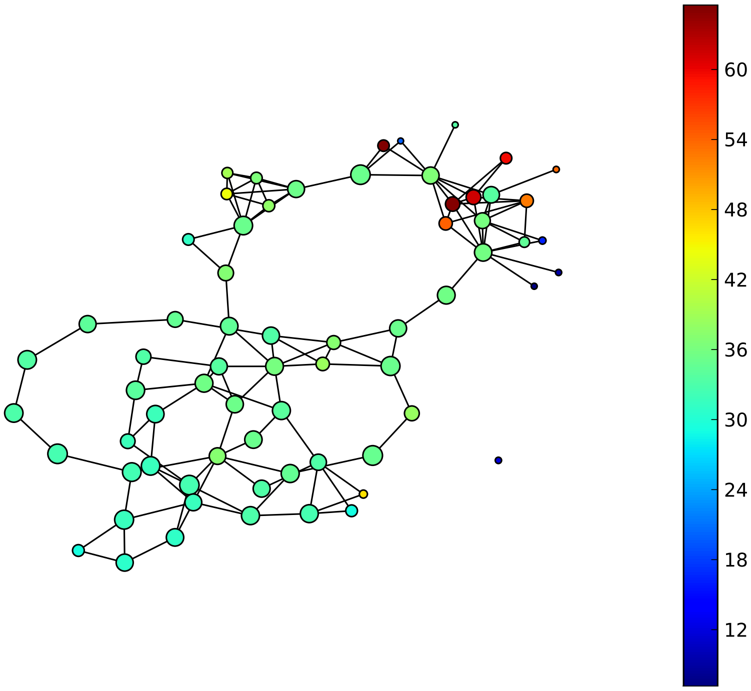
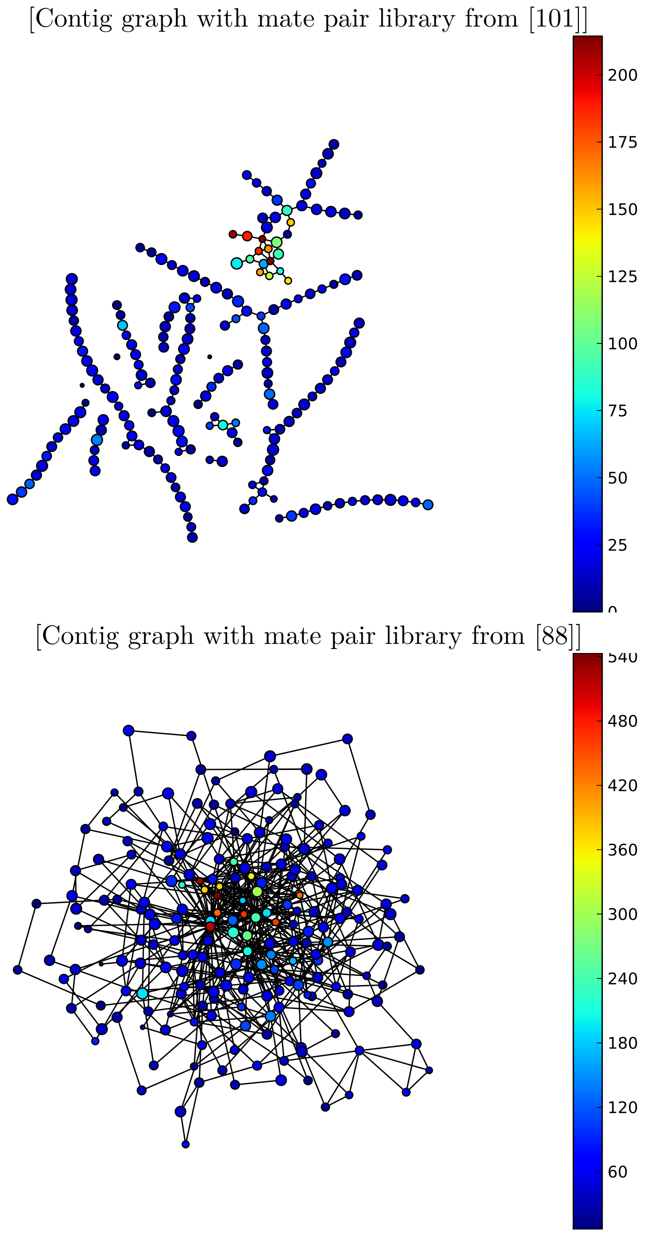


Figure 2.1: Contig graph for ALLPATHS-LG contig assembly of *Staphyloccocus aureus* (GAGE data). Color represents coverage from reads aligned back to the contigs and contig sizes are represented by the diameter of nodes (logarithmic scale) Although this is an illustration of a high quality assembly on a smaller genome, the graph captures some interesting features typical to contig graphs; Firstly, contig graph structure is relatively linear compared to arbitrary graphs, *i.e.*, the probability of two random nodes being connected is low. Secondly, contigs with higher coverage are usually repeats, thus involved in more complex regions as seen in the upper right part of the graph. Thirdly, in practice smaller contigs often shows a lower coverage (see blue small nodes), potentially due to difficulties aligning reads that partly overlaps with the contig or due to misassemblies in the contigs.



[Contig graph with mate pair library from [101]]

[Contig graph with mate pair library from [88]]

Figure 2.2: Contig graphs of the same ALLPATHS-LG contig assembly on *Rhodobacter sphaeroides* with two different read pair libraries. (a) Contig graph from a read pair library with approximate distribution N(3700,200) (b) Contig graph from a read pair library with approximate distribution N(2500,1500)

Contig graphs in practice

The graph structure of a contig graph depends on the amount of collapsed repetitive sequences in the contigs[2]. Contigs containing collapsed repetitive sequence will generally have increased coverage (see Figure 2.1) and they are often filtered out before scaffolding. Another factor that locally dictates the density of the graph is the fragment length distribution $f(x)$ in relation to the length of the contigs, studied in [31]. Increased average fragment length μ and standard deviation σ contributes to increased density; larger μ will contribute to read pairs spanning over more contigs, thus, increasing the number of neighbors for nodes. Similarly, increased variation in fragment length will allow both close and distant contigs to be linked — increasing the graph density. See Figure 2.2 for an example of contig graphs with two different read pair libraries. The density can be adjusted by requiring a minimum number on the read pairs that links two contigs. Generally, contig graphs are sparse for most read pair libraries and contig assemblies as two arbitrary contigs most likely are not linked.

2.2 Notation

We assume a set of contigs $\mathcal{C} = \{c_1, c_2, \ldots\}$ and a number of read pairs $\mathcal{R} = \{(r_1^1, r_1^2), (r_2^1, r_2^2), \ldots\}$ that have been aligned to the contigs. A read pair have a fragment length x following a fragment length distribution $f(x), \quad a < x < b$ with mean μ and standard deviation σ. The contig graph $\mathcal{G}$ is induced as follows. Each contig gives rise to precisely two vertices $c_{i,L}$ and $c_{i,R}$ in $\mathcal{G}$ where $c_{i,L}$ and $c_{i,R}$ denotes it's 5' and 3' end respectively. In a read pair, if r_i^1 aligns to precisely one contig c_k and r_i^2 aligns to precisely one contig c_m, with $k \neq m$, the read pair induces a relative orientation and an approximate distance between c_k and c_m. This relationship is represented as a *link* l.

2.3 Methods

The scaffolding problem (SP) defined by [42] is a combinatorial optimization formulation that is widely adopted. Let $\Phi : \mathbf{V} \rightarrow \mathbf{N}$ be an ordering, orientation and distance mapping of vertices in $\mathcal{G}$ to the natural numbers that preserves contig lengths. That is Φ induces scaffolds by giving contigs positions. We say that l_i is *concordant* if it has correct orientation and $x_i \in [i_{min}, i_{max}]$ in a given contig arrangement instance ϕ. SP is given as

[2] For computational simplifications, most scaffolders requires unique alignments of both mates to include the read pair as information for scaffolding. Therefore, read pairs from repetitive sequence that occurs multiple times in the contigs should in theory not create edges and therefore not increase the connectivity of the graph. However, repetitive sequence that has been collapsed into a single representation will create edges, as the alignments to this sequence will appear unique.

Scaffolding Problem (SP). *Find the instance ϕ_{max} that maximizes the number of concordant links in $\mathcal{G}$.*

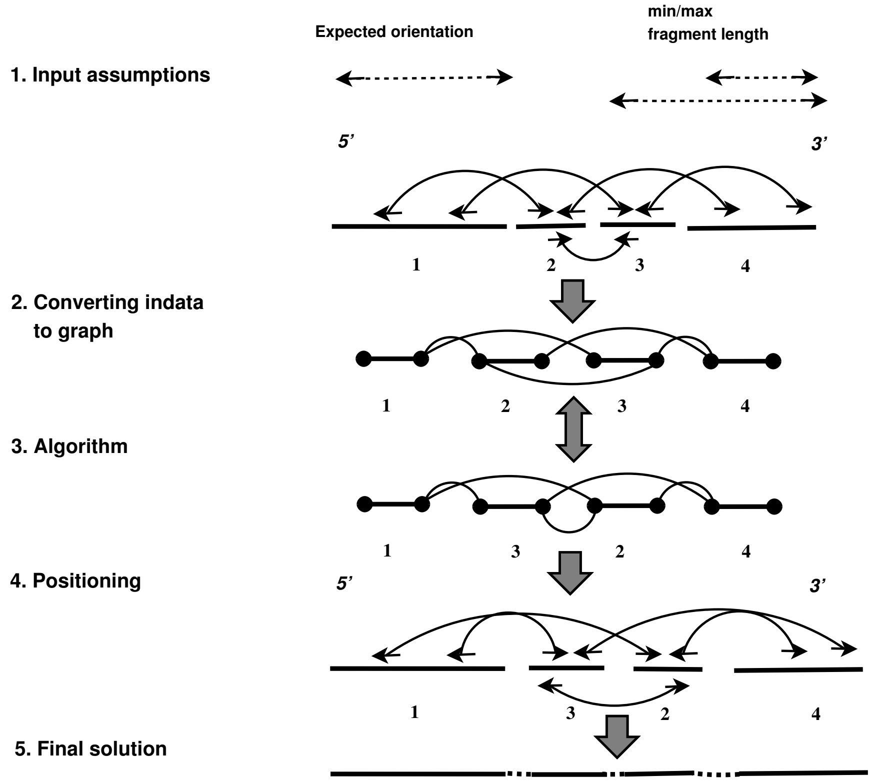


Figure 2.3: Example of a scaffolding workflow. 1) Assumptions about data are made. 2) Data is parsed and represented as a graph. 3) An algorithm is employed, *e.g.*, finding an ordering which maximizing the number of links with expected orientation. 5) Positioning of contigs in a linear order within the scaffold. 5) A scaffold containing contig sequence and unknown sequence (gaps) usually represented with the letter "N".

By reduction to the Bandwidth problem [42] showed that SP is NP-hard under this formulation. Although there exists other proposed formulations of scaffolding [17][3], SP has been used as foundation for a number of scaffolding algorithms, where the common denominator being to maximize the number of concordant read

[3] They introduce two parameters σ_p and σ_c that denotes the maximum number of linear and circular chromosomes allowed in a solution. These parameters makes sense theoretically prevent a scaffolding solution to contain less scaffolds/contigs than the number of chromosomes. However, in practice this rarely happens.

pairs. As methods have different boundaries on i_{min} and i_{max}, we will relax the definition of concordant to be dependent on the method under consideration, see Figure 2.3 for an example workflow in a scaffolder.

A common operation on $\mathcal{G}$ is to bundle together read pairs suggesting the same order orientation and approximate location of two contigs as a weighted edge e [42]. The weight is related to the support of the edge but differs among methods (*e.g.*, the number of links or the number of links weighted by mapping probabilities). We let $\mathbf{E}$ denote the set of edges respectively in $\mathcal{G}$.

Exact solution with dynamic programming

Although SP is NP-hard under general graph instances it does not mean that it is computational infeasible to solve an instance of a contig graph exactly. For higher quality bacterial genomes the problem instance might be small enough to permit an exact algorithm for SP, see Figure 2.1. Also, the sparsity of general contig graphs could be utilized as exact algorithms with polynomial time complexity (in the treewidth of the graph) are commonly employed on sparse graphs where $\mathbf{V}$ is large but treewidth is bounded [8]. For example, for several special graph structures there exists exact polynomial time algorithms for the bandwidth problem [39] that has similarity to the formulation of SP.

As fragment lengths, contig lengths and repeats determine how connected $\mathcal{G}$ will be (assuming no spurious edges) assumptions specific to these properties can be made. For instance, all correct contigs larger than i_{max} do not have any correct aligned read pairs spanning over them (assuming there are no overlapping contigs), which is information that can be used to decompose the problem. Also more explicitly to the topology of $\mathcal{G}$, articulation points can be used to split the problem into smaller instances. An articulation point (also referred to as a cut vertex) of a graph is a vertex that disconnects the graph if removed. By finding such points, one can individually solve the contig ordering and orientation of subgraphs induced by removing the articulation points. With more articulation points, subgraphs becomes smaller and tractable to solve.

Proposed algorithms Two methods using dynamic programming (DP) on subproblems induced by decomposing the problem are SOPRA [23] and OPERA [31].

SOPRA [23] formulates a global optimization problem similar to SP for orienting the contigs. It use the articulation points in $\mathcal{G}$ to separate the problem instance into regions where the number of concordant orientations of read pairs are maximized. Note that this orientation step ignores the size of fragment lengths, *i.e.*, the step does not give contigs coordinates. The sparse structure of the contig graph permits an exact solution via a DP approach for simple regions, while a simulated annealing approach is employed to approximate the solution in more complex regions of the graph. The read pairs that do not have correct orientation in the optimal solution are removed. After the orientation step, read-pair distribution is used to determine the positions of contigs within a scaffold. If an inconsistency is found in the

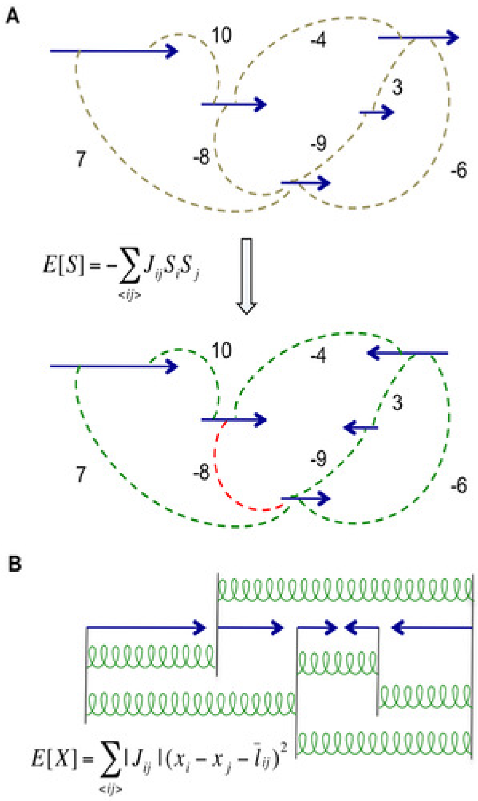

A
10
-4
3
7
-8
-9
-6
$E[S] = -\sum_{<ij>} J_{ij} S_i S_j$
10
-4
3
7
-8
-9
-6
B
$E[X] = \sum_{<ij>} |J_{ij}| (x_i - x_j - \bar{l}_{ij})^2$

Figure 2.4: Illustration of modeling constraints on the contig connectivity graph from [23] in SOPRA. (A) For two contigs i and j with orientations S_i and S_j connected through read pairs, the quantity J_{ij} represents relative orientation (sign of J_{ij}) and number of mate pairs (absolute value of J_{ij}). Minimizing E produces an orientation assignment that satisfies as many constraints as possible. Constraints not satisfied in the optimal configuration (shown in red) are ignored in the ordering and positioning step. (B) The relative position of contigs are modeled by the collection of mate pairs connecting contigs i and j, illustrated as a spring attached to the start points of contigs. The spring is relaxed when the distance between i and j in a solution is equal to the average suggested distance between the start points of i and j given by mate-pair constraints.

positioning step (*i.e.*, $x < i_{min}$ or $x > i_{max}$), the contigs attach to that edge is removed and the algorithm restarts at the orientation step. The iterative solution of optimizing orientation and positioning separately does not solve SP where the two needs to be optimized simultaneously. Thus, no guarantee can be given for the solution to be optimal. Figure 2.4 depicts part of the workflow in SOPRA but also illustrates the essence of the scaffolding problem, *i.e.*, orienting, ordering and positioning contigs given the graph topology and link information.

Like SOPRA, Opera [31] aims to build scaffolds with a minimum number of discordant links, solving the problem exactly for local regions induced by forming scaffolds between two contigs that are both larger than i_{max}. For a partial scaffold S' (induced by the subgraph $\mathcal{G}'$), the set of contigs that are not in S' but are neighbors to a contig in S' are concatenated to S' (one at the time). Each new neighbor concatenated to S' induces a new partial scaffold S'' (with a new set of neighbors), and the algorithm continues to produce partial scaffolds recursively in this way. The concatenation of a contig to a partial scaffold can induce concordant, discordant or both types of edges in the subgraph $\mathcal{G}'$. Opera continues to recursively build scaffolds in this way using the number of discordant edges p in $\mathcal{G}'$ (induced by the partial scaffold) as a design criterion. By treating p as fixed, they can obtain a polynomial time algorithm to find an optimal (with respect to a given p) solution to their slightly modified version of SP; maximizing bundled edges instead of individual links. The algorithm then tries all p starting from $p = 0$ and stops when a scaffold can be constructed. The algorithm is polynomial in the parameter w (for the width), where $w = i_{max}/c_{min}$ where c_{min} is the minimum contig length, that is w is the maximum number of contigs (assuming no overlaps) that i_{max} can span. This dictates how “wide” the contig graph can be for a given read pair library.

The approaches in SOPRA and Opera are algorithmically appealing. Although the approach presented in SOPRA maximizes the orientations of read pairs optimally, the arrangement ϕ_{max} is not guaranteed as fragment length information has not been regarded. Figure 2.5 shows the correct ordering (a) and the solution that maximizes the number of read pairs with correct orientation (b) in a region. The use of larger contigs to divide the problems in Opera relies on the fact that the larger contigs are correctly assembled with no or few erroneous links and the algorithm is sensitive to paired-end contamination [96] (discussed in Section 3.1). Furthermore, the recursive structure for building scaffolds in Opera is memory consuming if the treewidth (*i.e.*, set of neighbors) is large as each neighbor to a partial scaffold S' yields a new partial scaffold — creating a drastic increase in the number of partial scaffolds. Thus, a lower bound on contig size is introduced in the implementation of Opera. Finally, both SOPRA and Opera permits only unique placements of a contig, thus repeats will get no or unique placement. The relaxation of scaffolding with repeats is further examined in Opera [32].

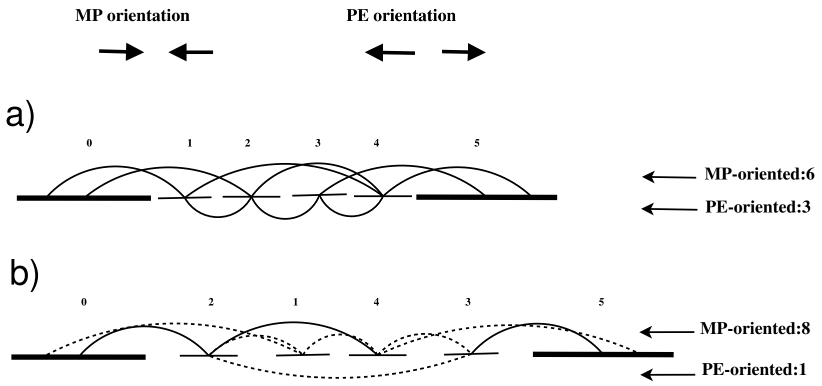


Figure 2.5: Illustration showing scaffolding error when maximizing number of consistent orientations due to incorrect assumptions about data. a) True placing of contigs in a region. b) Placing by only optimizing the number of correct mate-pair (MP) orientations (according to data assumptions). Links that are deviating from $[i_{min}, i_{max}]$ appear dotted.

Maximum matching

A matching M is a set of edges in a graph where no edges in M share a common vertex, and M is called maximal when no edge can be added to M without breaking the matching property. In a given graph, the matching(s) M with the largest possible number of edges is called a maximum matching. Let *intra-contig* edge refer to edges between the two nodes representing contig ends (5' and 3'). Notice that a matching consisting of all intra-contig edges, denoted M_{intra}, is a maximum matching. Let *inter-contig* edges denote edges between different contigs and M_{inter} denote a matching in $\mathcal{G}$ consisting of only inter-contig edges. M_{inter} naturally allows at most one edge from each contig end. Two maximum matchings M_{intra} (trivial to find) and M_{inter} forms simple paths in $\mathcal{G}$ — as there can only be one inter-contig edge linking a contig in each end. The simple paths naturally induces linear scaffolds with the exception that cycles are allowed in this formulation. To find a maximum matching covering all nodes without cycles is equivalent to finding a Hamiltonian path, thus NP-complete. Even relaxing the problem to finding a set of cycles and paths (with a given constraint on the numbers of cycles and paths) in $M_{inter} \cup M_{intra}$ covering every vertex in $\mathcal{G}$ where $\mathcal{G}$ contains no self-loops is NP-complete [17]. However, a solution permitting an arbitrary number of cycles can be found in polynomial time, *e.g*, with the blossom algorithm [26]. Therefore, a solution containing cycles is preferred for practical reasons. In case the final maximum matching contains cycles, they are given separate treatment.

Notice that the matching approach described above permits only a unique placement of a contig in a scaffold, thus repeats will get no or unique placement. More-over, at most one neighbor of any given contig can be chosen with this approach. As $\mathcal{G}$ contains information from the read pairs linking contigs, such as number of links and their placements on contigs, the information can be incorporated in SP formulated as a maximum matching problem.

Proposed algorithms [63] formulates SP as a minimum cost maximal matching problem. Intra-contig edges get a cost of 0. All inter-contig edges are associated with a strict positive cost corresponding to the maximum length difference in gap size by the links between the two contigs. The cost can be interpreted as describing the uncertainty of placements of read pairs in a given edge. A minimum cost maximum matching solution induces a set of scaffolds as described above. Every cycle is transformed into a path by removing the edge with the highest cost.

ScaffMatch [64] uses the number of links as the weight of the edges in $\mathcal{G}$ and solves a maximum weight matching problem providing two different algorithmic approaches. The first approach is using the blossom algorithm [26] to find a maximum weight matching among the inter-contig edges. The union of inter and intra contig edges form paths and if no cycle is present, this forms the solution. Otherwise, the edges with the smallest weights in each cycle is given -1 as weight in a new step of the blossom algorithm and this is repeated until the solution is free of cycles. The second approach is to iteratively choose the edge with the largest weight that does not introduce cycles or vertex with degree 2, until no more feasible edges are left. This reduces the runtime complexity of the method based on the Blossom algorithm from $O(n^3)$ to $O(n \log n)$.

As the maximum matching approach only permits one neighbor of every contig to be joined, this can be a limitation on assemblies where the fragment length is relatively large compared to contigs, making mate pairs link to several correct neighbors. [64] acknowledge this and introduce an "insertion"-step where contigs not chosen by the matching is inserted in gaps of existing scaffolds formed by the maximum matching solution.

Linear constraints and (M)ILPs

As read pairs follows a distribution $f(x)$, and has given relative orientation (*e.g.*, forward reverse as for Illumina mate pairs), they give information about relative placements and positions of the contigs. This information together with contig lengths induces distances between contigs and predominantly, a linear layout of contigs is assumed in the final solution. This interval structure motivates that the problem is partly geometric with similarities to the Interval scheduling problem, which is often formulated with (Integer) Linear Programs [107]. Therefore, one approach to SP is to find an ordering and orientation by formulating the problem as an Integer Linear Program induced by distance constraints based on $f(x)$ and contig lengths. The objective function is, similarly to SP, usually maximizing the number of concordant links (or as a minimization problem for the number of discordant links). The unknowns in this problem is the orientation of a contig in a scaffold (forward or reverse), the position of the contigs in the scaffold (positive integer) and a penalty (real valued) describing how much each edge in the contig graph agrees with the positions assigned to contigs in a given solution. The choice of domain for variables, *e.g.*, the penalty dictates if the problem is a mixed integer linear program (MILP) or ILP.

Proposed algorithms SILP2 [58] formulates an ILP that maximizes the log-likelihood of scaffold orientations by introducing boolean variables over contig- and edge orientations. The probabilistic framework is introduced by assigning probabilities to read links based on coverage similarity between linked contigs and alignment quality of reads. As they note, if all read pairs have the same probability of being aligned correctly and contigs have the same coverage, the ILP reduces to maximizing the number of concordant links. As variables are introduced for all contigs and edges, they employ a non-serial dynamic programming approach that takes advantage of the sparsity of $\mathcal{G}$. Assuming probabilities over alignments is novel to other approaches and better captures the uncertainty of links. However, modeling of a good probability distribution may be difficult. The probabilities described in [58] should be interpreted as weights as no probability distribution is provided or discussed. They give two separate parameters; p_{rep} describes a probability based on the amount of overlap with repetitive sequence and p_{cov} measures the normalized coverage difference between the two contigs where the mates are aligned. They state that the full "probability" of a read alignment could be either p_{rep}, p_{cov} or a factor of them.

MIP Scaffolder [99] and GRASS [35] formulate SP as a Mixed Integer Programming (MIP) problem. As for the DP approaches, articulation points are used to split the graph into biconnected components. They introduce real valued variables denoting penalties on how well distance constraints imposed by the edges are satisfied. The formulations are similar, but they use different techniques to find a solution.

MIP Scaffolder solves the MILPs on a subgraph $\mathcal{G}'$. $\mathcal{G}'$ is initialized containing only vertices representing the contigs and edges are constructed by adding the largest weighted edge (*i.e.*, the edge with the most links) one by one, until a threshold for the biconnected component size is reached. The individual solutions might introduce conflicting regions and they are resolved using heuristics such as removing edges that are stretched or contracted more than a given threshold.

GRASS uses an Expectation-Maximization algorithm. The maximization step obtains degrees of penalties on contig links given fixed contig orientations. The penalties are set according to what magnitude the constraints for a link is violated. If a penalty is higher than a given threshold, the penalty of the link is "de-activated", *i.e.*, its constraints are not considered. The expectation step is used to obtain the expected contig orientation of links given the activated penalties set in the maximization step. Links that are activated in the final solution are used for scaffolding.

Similarly to previously mentioned methods SLIQ [92] formulates a set of linear inequalities derived from the geometry of contigs. Scaffolds are formed by first finding relative orientations of contigs (by finding, *e.g.*, a spanning tree of $\mathcal{G}$). After orientations are fixed majority voting (total link weight on edges) by links satisfying the linear constraints is used to predict placements of contigs. As for previous described approaches, the ILP based methods presented here does not handle repeats.

Greedy

Not only is SP NP-complete for general graph instances, mammalian and plant genomes often give data instances containing millions of vertices and edges making super-linear polynomial approximations infeasible [41, 98]. However, if the majority of contigs is at most a couple of hundred base pairs (often the case in practice), a decision of which contigs are most important to scaffold, and which to leave out, is motivated. Also, as edges in a contig graph can be either spurious or true, a simple classifier for this has the potential to work well if the classifier is good enough. This motivates that a greedy algorithm, choosing which contigs to scaffold and what edges to trust, may be appealing in practice.

Proposed algorithms Greedy algorithms proposed to solve SP include SSPACE and Bambus [9, 81]. The published version of SSPACE choses the largest contig that has not been joined in a scaffold yet, and selects the edge with the most amount of links to extend the contig. If there are more than one edge and the ratio of number of links between the two edges with most links is smaller than a given threshold, SSPACE chooses the dominating edge to extend the scaffold, otherwise the extension is stopped. In later versions of SSPACE, it is noted that larger contigs in general has more links, therefore, they normalize the number of links with the contig lengths, creating a form of "link density". However, it is not described exactly how this procedure is performed. Similarly to SSPACE, Bambus2 builds scaffolds in the same greedy fashion with heuristics to remove edges with inconsistent fragment lengths suggested by contradicting links.

The simple greedy classifiers are practical and their implementations are competitive or even outperforms implementations of more advanced algorithmic approaches [41][4]. However, there are graph regions and characteristics common to contig graphs permitting simple solutions that a greedy edge classifier does not treat.

[4] Scaffolding consists of several steps and it is not yet shown what step(s) in a method that gives the performance difference — the core algorithmic framework on $\mathcal{G}$ or the pre- and post processing of data. So far, evaluations have only compared full package implementations (discussed in Section 3.3).

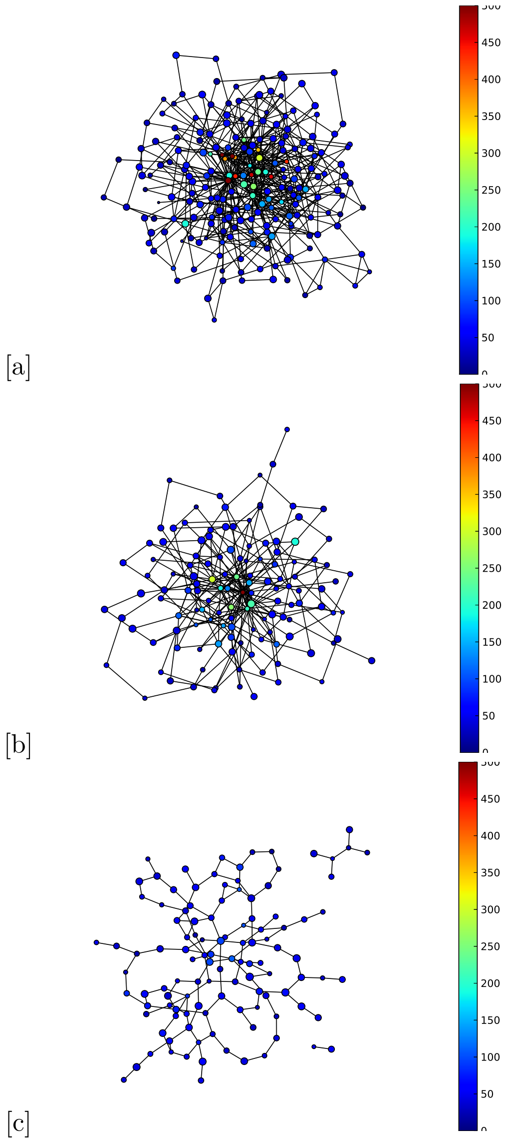


Figure 2.6: Contig graphs constructed by BESST for the ALLPATHS-LG contig assembly on *Rhodobacter*. The graph density is increased by repetitive and smaller contigs. The repetitive contigs can often be classified by looking at coverage (color coded in figure). All contigs are included in [a], [b] is the graph constructed by contigs $> i_{max}$ and [c] is obtained by fitering out repeats from [b] based on coverage. In the default setting of BESST it will start scaffolding with graph in [c], and include smaller contigs in a later step.

Mixed methods - Exploring structures and features common to contig graphs

Greedy methods are appealing for their speed, but one can take the greedy approaches a bit further by tailoring graph operations that can resolve smaller structures typical to the topology of contig graphs. Instead of choosing a universal decision function that choses what edges to trust in $\mathcal{G}$, a set of different heuristics can be chosen based on the local behavior of the graph.

Another observation is that a contig graph consists of true and false edges. If the majority of false edges are due to random independent events with relatively low frequency, *e.g.*, chimeric read pairs or misaligned reads due to substitution errors, the formulation in SP is motivated as the majority of links (hence edges) would be correct. But spurious edges might come from misassembled contigs, collapsed repeats or paired end contamination in a mate pair library (see Section 3.1), creating clusters of spurious links. In such cases SP is a too simplistic formulation of the scaffolding problem and instead, it is desirable to remove contigs or filter out edges that seem to be incorrect from the distribution of read pair links before scaffolding.

Proposed methods The built in scaffolder in ABySS [105] is based on a set of heuristics to resolve regions of various kinds based on the graph topology, such as closed simple bubbles, forks, repeat regions, tips, and weak, transitive, or ambiguous edges [45]. All of these "regions" involve a smaller local set of nodes and require only local operations. For example, a simple closed bubble consists of four nodes s, t, u, v in a DAG where s is a source and t is a sink and there exist exactly two paths s, u, t and s, v, t (simple) and there are no incoming edges from or outgoing edges to other parts of $\mathcal{G}$, except at the source and sink vertices (closed). In this case the relative placement of u and v are ambiguous (at least from the graph topology) and the authors remove u and v from the graph and an edge between s and t is added. Similar graph modifications are performed for the other events, usually involving around five nodes.

BESST [98] utilizes statistical information from read pairs and the size of contigs to first scaffold larger contigs (essentially working with a subgraph $\mathcal{G}'$ of $\mathcal{G}$ where smaller and repetitive contigs are filtered out), see Figure 2.6 for an example of forming $\mathcal{G}'$ (Figure 2.3). The main point is to look at the distribution of links across contigs and decide if they match the expected characteristics of the read pair library instead of maximizing the amount of links. This is motivated by the presence of artifacts in data such as chimeric contigs read pair library contamination discussed in Section 3.1 that can give rise to edges with several read pairs. In a second step, BESST find paths in a brute force Breadth First Search (BFS) with a heuristic stopping criterion of the BFS. The brute force path finding is based on the assumption of a relatively linear structure in a contig graph. As the number of paths can blow up for complex regions, the approach is in theory not efficiently designed. However, this is rarely seen in practice. Also, such a region might not be desirable to scaffold (obtain a linear order of the complex structure). The classification of larger and smaller contigs are made with respect to the contig length and fragment length of the read pair library. If there are no contigs classified as large enough for $f(x)$ to be border contigs, BESST fails. This classification threshold can be manually set, but it may come at a price of more aggressive scaffolding. This is a limitation with BESST. BESST was shown to perform especially well in relation to other scaffolders when scaffolding with wider read pair libraries [98], potentially due to the improved gap size estimation making contig placements more accurate [95], which has bigger effect with increased variation of fragment lengths.

[113] describe an algorithm implemented in the genome assembler SPAdes [4], that similarly to BESST uses a scoring function based on the distribution of reads to extend scaffolds. However, they work directly on the DBG. They evaluate paths to extend a scaffold with, where they motivate that the sparsity of the DBG permits this approach for general regions. If the graph has too many extension paths they set a heuristic stop threshold for the number of paths to evaluate.

SCARPA [24] is a scaffolder that attempts to remove contigs that seem misassembled. By using two nodes per contig, the problem of orienting contigs has a feasible solution if and only if $\mathcal{G}$ has no cycles containing an odd number of nodes. This can be realized by noting that an odd number of nodes means that the 3' and 5' ends are not the same — one contig contributes to two inter-contig edges. They find the minimal number of nodes (contigs) to remove to allow for a feasible solution. They adopt a similar approach (odd cycle removal) to find spurious edges. The algorithm is engineered to perform odd cycle removal of edges and contigs simultaneously and prefers removal of spurious edges to contigs. After the orientation step of contigs, they are ordered linearly by first removing any cycles with even number of nodes, then a LP formulation is used to get coordinates of contigs within a scaffold.

2.4 Scaffolding with other information

Read pairs provide a natural type of information for scaffolding as the approximate distance between reads is known. But they are not the only type of information available for scaffolding contigs. In addition to paired reads, there has been work on scaffolding with long reads, restriction maps, RNA-seq and reference sequence from related organisms.

Third generation sequencing long reads as offered by the PacBio RS methodology can solve complex genomic structure by, *e.g.*, span over repeats or phase alleles. By mapping the long reads to contigs, scaffolds can be created if the long read has parts that maps to different contigs, this is explored in, *e.g.*, [10]. However, the reads suffer from a relatively high error rate (approximately 15%) and the throughput is smaller than read pair sequencing. This makes long reads most useful for assembly and scaffolding of smaller genomes at present, where high enough coverage can be obtained to correct the errors in reads. Restriction maps detect known restriction sites within a sequence of DNA by cleaving it with a specific restriction enzyme. An optical restriction map is a restriction map that provides an ordered list of fragment sizes resulting from the restriction sites. Fragment sizes of any contig can be found by *in silico* digest of the restriction enzyme and these fragment sizes are compared to fragment sizes of the optical restriction mapping of the genome to order and orient contigs along the genome [75, 93, 109]. Genome-wide chromatin interaction data sets have also shown to be useful for scaffolding contigs [15]. To get a better coverage of the gene space in an assembly, RNA-seq data [73, 123, 94] can be used for scaffolding contigs around gene sequence. Reference sequences of related organisms has also been used for scaffolding [89, 11, 50], but have only been evaluated on prokaryotic or smaller eukaryotic genomes.

However, reference based assembly is not applicable to most *de novo* sequencing projects, restriction maps are often not available, RNA-seq data only have coverage over genes and contains no information about distance between reads which makes contig placement ambiguous, and the throughput and error rate of long reads currently makes it feasible only for smaller, less complex genomes. This makes read pair information the most commonly used (and often also the only applicable) source of information for scaffolding at present.

2.5 Evaluation of scaffolders

There exist several projects benchmarking genome assemblers, and their integrated scaffolders [25, 12, 101, 62] using simulated and real data with high quality reference sequences available. However, even with a reference sequence, they encountered several problems in identifying the best assembler. Defining the evaluation metrics is difficult.

- There is often a trade-off between the number of misassemblies and the contiguity of an assembly — but the trade-off level may be difficult to define.

- Determining what counts as misassembly is subjective and depends on the objectives of the downstream analysis. Defining possible misassemblies (*e.g.*, inversions, relocations, translocations, insertions, deletions [101]) is a computational problem in itself. Some of these misassemblies can be ordered in size — should they be weighted based on their significance in that case?

- Regions around genes might be more valuable and therefore giving higher priority to assemble correctly in an evaluation.

Also, [101] clearly demonstrated how the same assembler can give significantly different quality on similar datasets (bacterial genomes) which hamper a generalized statement about the performance of a tool. This makes even referenced based assembly evaluation is difficult, but there are also attempts at reference free evaluation using the same type of data as in the original assembly (paired end or mate pair reads) [87, 114, 21, 40]. This problem is as difficult as genome assembly itself as input information is similar. QUAST [36] is another tool that evaluates assemblies using a reference genome and gene families (if provided). QUAST is easy and fast to use and offers a large amount of evaluation metrics that makes it a suitable tool for benchmarking assemblers and scaffolders on organisms with known genome sequence. However, the classification of misassemblies in QUAST is in some cases too simplistic as there is only one threshold determining if the misassembly is local (smaller) or not. This has its limitation in benchmarking assembly software as shown in [100]. Ideally, one would want more detailed information about the size of misassemblies (where possible), which motivates investigating edit distance between contigs/scaffolds and the reference. One way to approximate the quantity of misassembled sequence in a scaffold is to extract pairs of sequences separated by a given distance on the scaffolds and map them back to the reference to give a proportion of pairs whose orientation and approximate distance match [99, 34].

Using other type of data to evaluate an assembly is crucial, as the same type of data might bias the evaluation. CEGMA [80] uses eukaryotic core gene sets to map onto assemblies in order to evaluate how many genes are fully or partially represented in contigs. BUSCO [103] is a similar tool that uses single copy orthologous gene sets to look at the gene space quality of an assembly.

There have also been attempts to evaluate stand-alone scaffolders [41, 98]. As with evaluation of genome assemblers, it is difficult to nominate one scaffolder as the "winner", or obtain an objective ranking in general, as there are several possible metrics to evaluate by, often with a tradeoff in, *e.g.*, errors and contiguity. Also, the quality difference between datasets (both contig assemblies and read pair libraries) varies greatly, making scaffolders suited for datasets with different characteristics. However, one important conclusion from [41] was that the type of read aligner and its parameter settings have a big impact on the quality of the scaffolding. This suggests that scaffolders might benefit from also looking at alignment quality and adjust to aligner-specific artifacts such as in [58].

Chapter 3

Future work on Scaffolding

The number of algorithms proposed for scaffolding and the their variation in methodology suggests that scaffolding is a difficult problem. A possible explanation of the difficulty is the artifacts and unmodeled characteristics of the problem in practice. A simple example is that in SP, the mapping ϕ_{max} that maximizes the number of concordant links only permits one placement per contig, thus it does not address repetitive contigs. The following artifacts and characteristics are most likely present in biological data.

- *Sequencing technology artifacts*: Erroneous sequence introduced in reads from the sequencing technology such as single nucleotide substitutions and small indels (see Chapter 4) and increased variance in sequencing coverage. Specific to Illumina mate-pair protocol, spurious read pairs are created; chimeric reads (junction adapter in the middle of one mate in the read pair), chimeric read pairs (two distant fragments have been joined together in the circularization step) or PE-contamination (discussed in 3.1).

- *Repetitive contigs*: A contig that is present multiple times in the genome but assembled into one copy in the assembly. Therefore it needs to be placed in several locations (if possible) during scaffolding, see Figure 3.

- *Assembly errors*: Due to, *e.g.*, chimeric contigs caused by repetitive regions, see Figure 3.

- *Heterozygozity*: Multiple "copies" (not identical) of chromosomes in polyploid organisms. Similar or identical regions between copies are collapsed into one contig while highly heterozygous regions give rise to several contigs. The objectives can also vary: Either we are interested in the difference between copies in which case we want to separate (or phase) the copies, or we want to create a consensus assembly in which case we want to merge contigs from the same allele.

- *Misalignements*: Due to any of the above issues. Another cause is the repetitive sequence in contigs with respect to the read length or the aligned read pairs. Sequence that did not appear repetitive when assembling with reads of length y can be repetitive when scaffolding with read pairs with mates of length x, with $x < y$.

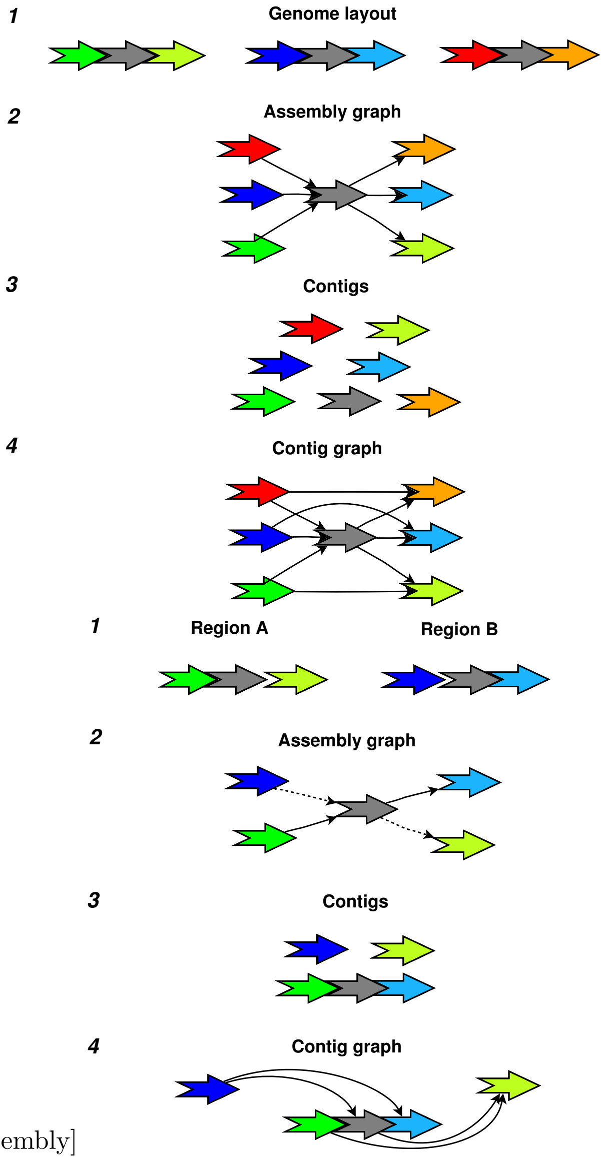
It is therefore questionable if solving the idealized problem formulations exactly, such as SP, gives a solution close to the true solution. The above stated issues are well known in the genome assembly community and comprehensive work has been made on the individual issues stated above, *e.g.*, cleaning of mate-pair libraries [78, 129], identifying misassembled contigs *de novo* [40, 21, 87], assembling repetitive regions [14, 84], and assembling highly heterozygous genomes [48]. These methods are either stand-alone or integrated in end-to-end[1] assemblers.

Although modularity in assembly pipelines is preferred due to the transparency in the algorithmic workflow (see Section 3.3), some problems are closely intertwined and solving them together gives a better solution than solving them in stand-alone modules (the orienting and ordering step in scaffolding is an example of this). Therefore the quality improvement in an integrated solution might be preferred over the transparency that modularity offers. As misassembled and repetitive contigs are part of input and output to the scaffolding problem in practice, and read pair data provide information well suited for breaking misassembled contigs and placing repeats, it motivates the integration of these problems.

However, methods addressing the above list of practical problems in scaffolding have not yet been widely integrated in scaffolders. To the author's knowledge, Scarpa [24] is currently the only stand-alone scaffolder that attempts to identify and remove misassembled contigs. Opera-LG [32] is the only stand-alone scaffolder that attempts to place repeats in multiple places in scaffolds. SILP2 is the only scaffolder to weight alignments based on their alignment features with a probabilistic framework — however they still consider only unique alignments. Also, among stand-alone scaffolders, only BESST and Opera-LG have shown to have accurate models for contig positioning. As such a fundamental problem is widely unnoticed, it highlights that work on scaffolding is still in its infancy and significant work needs to be done both on modeling the problem, and on the implementation side[2,3]. Furthermore, there is a paucity in literature on integrated scaffolding methods (except a few cases, (*e.g.*, SPAdes integrated scaffolder [84, 113] and ABySS [45]). The integrated scaffolding methods are often presented as a single section in an article describing the full end-to-end assembler, giving limited insight into the algorithms.

One of the practical problems with data listed above is the PE-contamination in Illumina mate-pair libraries. Although this library artifact is specific to Illumina sequencing technology, these libraries are the most widely used for scaffolding larger organisms and the most common data used to benchmark NGS scaffolding

[1] Taking raw read files such as fastq files and output contigs/scaffolds.
[2] http://bioinformatics.ninja/blog/2013/12/11/genome-scaffolders-suck/
[3] http://omicsomics.blogspot.co.uk/2014/01/envisioning-perfect-scaffolder.html



[Repetitive contig]

[Contig misassembly]

Figure 3.1: Examples of scaffolding instances. a) Repetitive contig: Repeats that cannot be solved in the assembly graph are collapsed into a single copy contig. This contig has multiple true placements in the scaffolds. b) Misassembled contig. Due to, *e.g.*, coverage drops, the assembly graph does not contain dotted edges. Therefore, a chimeric contig is formed. This is passed on downstream to the scaffolding step and the misassembled contig will give rise to edges that confuses scaffolding.

software[4]. So far BESST [96] is the only stand-alone scaffolder that identifies and models PE-contamination.

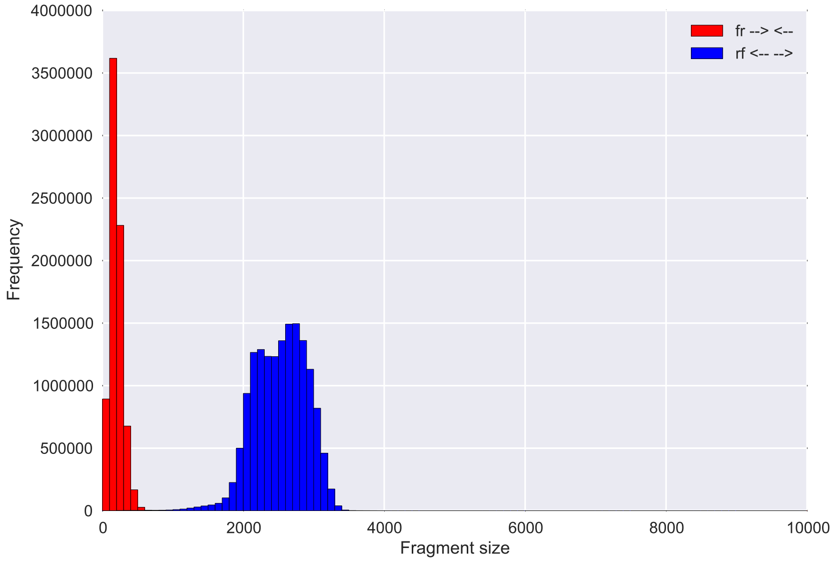


Figure 3.2: Histogram of fragment lengths of a read pair library from Illumina jumping library sequencing of Human Chromosome 14 (GAGE dataset [101]). Based on alignments with BWA-mem [55] onto the reference sequence, the paired end contamination is approximately 33% with a mean of 200 base pairs.

3.1 Paired end contaminated mate-pair libraries

The type of technology used to obtain read pairs and their main parameter, the fragment length, determine how far apart the reads are distributed on the genome and thereby at what distances contigs can be connected into scaffolds. A common sequencing technology for obtaining read pairs is Illumina sequence technology, called *jumping libraries* or *mate pair libraries* (MP). This sequencing technology has the highest throughput and lowest cost per base pair [59, 85], which makes it the most commonly used sequencing technology for larger genomes today. However, there are also numerous complications with the methodology. For example, the fragment length is not perfectly controlled and larger fragment length typically

[4] Mate-pair libraries has been used almost exclusively as benchmark data in articles introducing the methods presented in previous chapter as well as in the evaluation study from [41].

means a larger variance in the distribution of distances between the reads, making scaffolding harder [98]. Other examples of sequencing artifacts are multiple copies of identical reads pairs (duplicates) due to the amplification step in the sequencing protocol, and chimeric read pairs due to spurious joined fragments before the sequencing step. Another problem is the so-called paired-end (PE) contamination of MP libraries, which is a consequence of the MP library preparation. During the process, an unknown fraction of fragments that do not contain the circularization junction are sequenced. These misreads behave like PE reads, with opposite read direction to MP and effectively with a much smaller fragment length [43]. Hence, PE contamination reads may confuse a scaffolder that assumes an MP library is clean from contamination, suggesting a different relative order of contigs. Figure 3.2 shows an fragment length histogram from a MP library containing both the reverse forward read pairs (MP) and forward reverse (PE contamination) distribution. The contamination rate in MP libraries can be over 50% as observed by [34] and they noted “We found that the greatest challenge in scaffolding the data of this work originated from artifacts present in the read pair data.”. With these library artifacts of the Illumina mate-pair libraries, it is therefore questionable if the idealized formulation (SP) is a good problem formulation to optimize for.

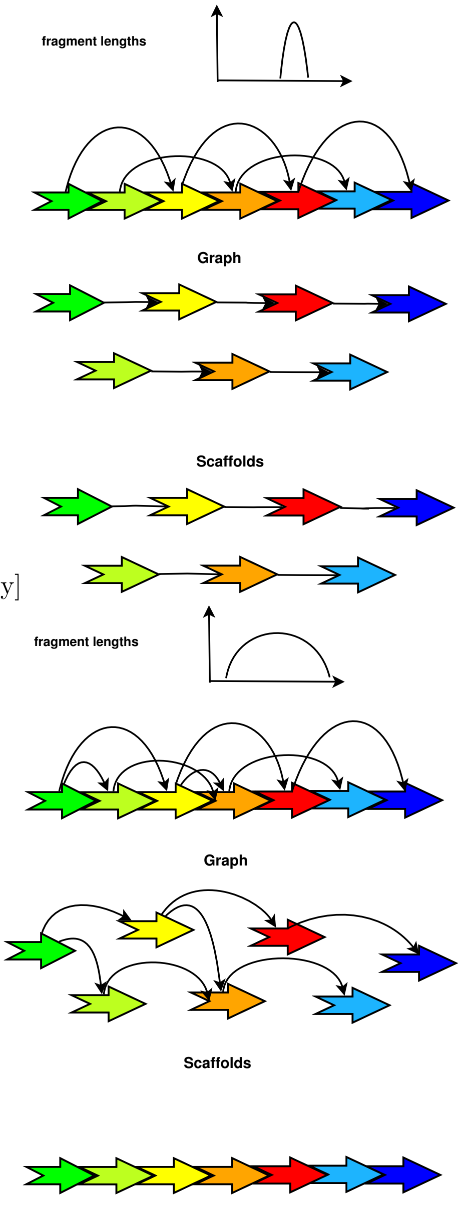
Although a decent MP library will contain more true MP reads than PE, and hence overshadow PE contaminants in terms of the total span coverage, it is not the case when there are many short contigs close to each other; making PE links dominate MP links, see Figure 2.5. PE-contamination has been discussed in work on integrated scaffolders in end-to-end assemblers such as ALLPATHS-LG [34] and MaSuRCA [129]. These methods relies on the fact that the orientation is observable, *e.g.*, by finding and removing the adapter sequence. But read pairs not containing the adapter will not have an observable orientation [78]. Furthermore, the methods for identifying and removing adapters are often inefficient, removing all the sequence to the 3’ end of the adapter [78], and quality of the output varies [46]. In theory, if there is a single simple path between the mates in the DBG, this gives unique information of the fragment length and orientation. However, due to gaps from drop in sequencing coverage or multiple paths between mates, orientation and fragment length of a read pair ultimately becomes stochastic. A method for scaffolding assuming stochastic orientation and fragment length of links is investigated in [96] and implemented in the update of BESST. We solve local scaffolding problems as ILPs with an introduced variable over the orientation. By using information from the interval structure of a contig graph (the lengths of contigs and fragment lengths), an efficient heuristic is provided to solve the ILP. Significant improvement was made in scaffolding over both integrated and stand-alone scaffolders on biological data from [101], illustrating the impact of PE-contamination in real data. By simulating different levels of PE-contamination, we investigated how other scaffolders are vulnerable to PE-contaminated libraries, resulting in increased number of misassemblies, more conservative scaffolding, and inflated assembly sizes. The drop in quality was most evident for fragmented assemblies.

3.2 Choosing fragment length distribution for scaffolding

On the user side, there is limited work on suggesting the read pair distribution characteristics to facilitate best possible scaffolding. In the study by made by [112] they argued that applying combinations of mate-pair libraries with insert sizes that match the distributions of repetitive elements improves contig scaffolding and can contribute to the finishing of draft genomes. They also concluded that larger or mid rage mate-pair libraries (5-25 kbp fragment length) were more efficient for genome structure analysis (larger scale connections of contigs) compared to applying a commonly used combination of paired-end reads and a 3 kbp mate-pair library. The study was performed by scaffolding an high quality contig assembly of the mammalian genome *Rattus norvegicus* with different combinations of read pair libraries using SSPACE. Conclusions was based on increase in N50 and decrease in number of scaffolds after scaffolding. They also evaluated scaffold quality on one chromosome for the combination of libraries that gave the largest N50 and smallest number of scaffolds (which was the combination of using all libraries[5]) by looking at concordance with contig order from optical mapping. Larger structural errors from the scaffolding were revealed in 236 100kbp bins out of 872 on the chromosome. As there were no evaluation of correctness between the different scaffolding runs the trade-off of connectivity and the number of misassemblies was not considered. It is therefore difficult to evaluate and quantify the gain in quality of the assembly from scaffolding as well as electing the best combination of mate-pair libraries. Also, SSPACE was the only scaffolder used, it might be that this particular scaffolder works well with the libraries suggested but other scaffolders benefit more from other library sizes. Finally, the outcome in result of this analysis depends on the initial quality of the contig assembly (N50 of 37kbp), which is large compared to all the libraries used. Therefore, larger libraries might be suitable for this particular assembly, not for more fragmented assemblies.

A more theoretical study was performed by [119] where they similarly to [112] concluded that mate-pair libraries tailored the the repeat distribution of the genome gave the best results, and therefore advocated performing assemblies with paired-end reads before constructing the mate-pair libraries. However, contrary to [112], they found that “in high-coverage prokaryotic assemblies, libraries of short mate-pairs (about 4-6 times the read-length) more effectively disambiguate repeat regions than the libraries that are commonly constructed in current genome projects”. The evaluation was performed on a large range of simulated prokaryotic datasets assembled with SOAPdenovo. The analysis focused both more theoretically on the complexity of finding paths between mate-pairs in a DBG (to be able to form a contig/scaffold), and on the final result (N50) of SOAPdenovo.

[5] This is not surprising as SSPACE (similarly to BESST and SCARPA) will create scaffolds with using library at the time (in increasing order). Taking the scaffolds created in previous step as contigs for the new step. This is dangerous for quality as scaffold errors are passed on to the next step and has the potential to build onto chimeric scaffolds.



[Narrow mate pair library]

[Wide mate pair library]

Figure 3.3: Illustration of one benefit of using a wider library distribution to a narrow one. If there is no or little variance in fragment length (a) adjacent contigs might not be linked. If the library has more variable fragment length, connectivity in contig graph has more links enabling scaffolding the whole region into one scaffold.

Although these studies advocates choosing mate-pair sizes tailored to the repeat distribution of the genome, more work could be done. For instance, contig sizes in relation to fragment length is crucial for scaffolding quality. As shown in [96], fragmented assemblies with (relatively) large fragment length libraries can cause inflated assembly sized containing lots of gaps due to the inability to place several neighboring contigs — a practical problem with many of todays scaffolders. Not only the size of fragment length matters; the variation of fragment lengths will affect scaffolding quality. With more variance in fragment lengths, contigs are more difficult to place accurately as well as evaluating misassembled contigs with intra-contig read pairs. However, there is also a benefit of more variance in fragment length if only one library is used. A library with higher variance will in general link together more contigs as it consists of more short range and long range read pairs. This can facilitate the connectivity in scaffold regions linking together more contigs, see Figure 3.3. However, scaffolding methods that can use several (narrow) libraries simultaneously with varying insert sizes are to prefer as it gives both the connectivity and accurate placements.

3.3 Evaluation

No study has yet been made on how significant the discrepancy are between formulations such as SP, and the practical problem of scaffolding. In other words, how much the optimal result of SP differ from true scaffolds. The lack of such evaluations could be due to difficulty in implementing such an exact algorithm, the time complexity of solving SP optimally, and/or the fact that merely evaluating scaffolds is difficult (see Section 2.5). Another potentially more important limitation in evaluations is the inability to evaluate improvements in the "core" methodologies of scaffolders.

Both independent evaluations, *e.g.*, [41], and evaluations in new-method publications evaluates the full "package" implementations rather than evaluating the core algorithms. Both genome assembly and scaffolding (viewed as a sub-step in a genome assembly pipeline) consists of several steps

- Filtering, error correction and processing of data.
- Core algorithm(s), *e.g.*, DBG assembly and scaffolding.
- Heuristics to remove infeasible solutions produced by the core algorithm.
- Finalizing and finishing output.

It is therefore difficult to interpret what parts contribute to the improvements. Although it's appreciated by the community to provide an end-to-end solution that is easily used in practice, it is a drawback for the development of new algorithms. Firstly, because it inhibits of evaluations of the individual steps in the algorithm. Secondly, for availability (*e.g.*, code and documentation) of the separate steps as modular implementations, discussed in, *e.g.*, [83, 67].

3.4 Summary

Although scaffolding with read pairs from NGS methods might only be around for a couple of more years until sequencing techniques providing longer reads with less errors take over, scaffolding is still an active research area. Some examples of limitations with current scaffolding algorithms that is important to address are

- Repeat resolving integrated with scaffolding — Being able to scaffold (place) the repeats that are spanned over, as discussed in [32].
- Soft alignments - Weighting the links by probability of alignment, as discussed in [58]. Or even better, probabilistic assignment over reads, allowing multiple mappings of a read.
- Heterozygozity — Identify and scaffod graph regions containing split allele contigs from polyploid organisms. The objective could be either phasing out scaffolds representing several alleles or to create a consensus scaffold from the split allele contigs.
- Multiple libraries — Scaffold with information from several libraries simultaneously, *e.g.*, as in [34]. Currently most stand-alone scaffolding methods are designed for one library, and use multiple libraries only in an independent step-wise manner.
- Meta scaffolding — using several scaffolders to build consensus scaffolds.

Chapter 4

Structural variation

4.1 Introduction

A *Structural Variation* is a region where two or more genomes within an organism are differing from each other. In SV detection, the problem is to find regions where sequences are differing. The start and stop positions of these regions are called *break points*. As with the scaffolding problem, there are several types of data to use for structural variation detection. However we limit the discussion to NGS reads. More specifically, the scenario we discuss is when NGS reads have been sequenced from a *donor* genome and we have a *reference* genome to compare against. Notably, the reference is usually a single sequence although, *e.g.*, humans are *diploid*, that is having two "copies" of chromosomes. Here "copies" are quoted to highlight the fact that they are not identical - which is partly due to structural variation. In the two "copies" of a human chromosome, small differences in genomic content exists that might be either a single nucleotide polymorphism (SNP's) or larger differences where sequences of the genome might have been deleted, inserted or cut/copy-pasted elsewhere. This variation is natural and necessary for evolution. A large pool of different versions of genes help to protect a species from extinction from, *e.g.*, virus infections as it is more likely that some organisms will carry a copy of a gene that makes the organisms immune to that virus. Therefore, one of the reasons to why Structural Variants (SVs) are interesting to find and study is that they can help develop new vaccines and medicines by, *e.g.*, isolating the gene(s) with variants responsible for the immunity of a virus.

Although genetic variation serve as "protection" of a species, it comes at a price of causing dysfunctional genes. Although researchers currently has a limited knowledge on both methods of detection and the effect of SVs, studies have been showing causality between SVs and numerous diseases and complex traits such as epilepsy, Parkinson's, Alzheimer's disease, schizophrenia, and autism [108]. Another area where SV's are studied is cancer research. Inherited or accumulated over time (somatic) mutations in the genome might make an individual cell start to copy uncontrollably. The new copies most likely have the mutations passed on from the predecessor cell and new SVs accumulate.

Originally, structural variants were defined as as insertions, deletions and inversions greater than 1kb in size [30], but recent sequencing techniques have widened this spectrum to include smaller events [3]. Here, we don't put a size constraint on the classification of a variant. Typically SNPs are not included in the definition of structural variation. The size range of SVs varies from single base pairs to over 3Mbp [22] where most SVs are smaller than 10kbp [69, 61].

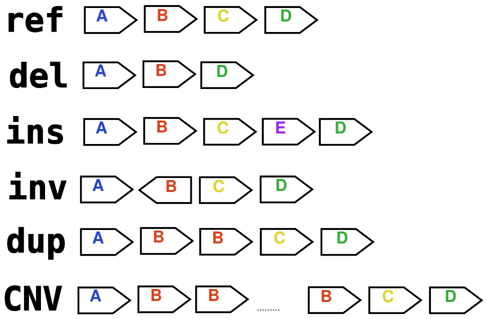


Figure 4.1: Examples of common types of SVs. From top: order of segments in the reference sequence, deletion, insertion, inversion, duplication, copy number variant.

4.2 Structural variation types

Consider an alignment of the donor sequence to the reference sequence. If two adjacent nucleotides b_1 and b_2 (downstream ordered) in the donor sequence are not adjacent in the reference sequence, we define b_1 to be a breakpoint to a SV. If there exists sequence between b_1 and b_2 in the donor not present in the reference, we denote this an *insertion*. If the there exists sequence between b_1 and b_2 in the reference not present in the donor, we denote this a *deletion*. These are two fundamental types of SVs and there are many other types of SVs that can be seen as a combination of deleted and inserted sequence (*e.g.*, cut and pasted or copied and pasted), see Figure 4.1.

4.3 Current methods

Methods for detecting SVs are usually based on one, or a combination, of the following information sources: (1) read pairs (RP), (2) split read/softclipped (SR/SO), (3) read depth (RD), (4) *de novo* assembly (AS) [66, 128, 57], see Figure 4.2. Two common approaches when using information from (1-3) is to either model the data or learn SV types by a machine learning approach [68, 19].

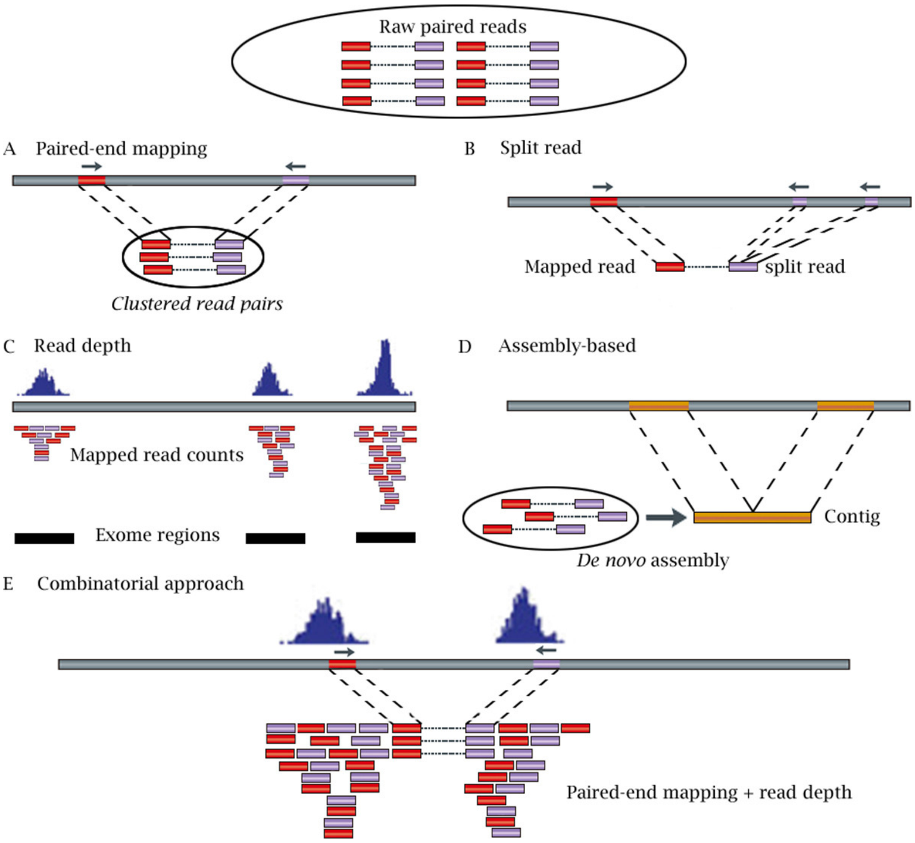


Figure 4.2: Different information sources used for SV detection. (A) Inconsistent alignments (distance or orientation) of read pairs. (B) Reads mapping partly, or with several parts mapping to different locations. (C) Coverage deviations in a region, (D) split or partly aligned contigs from a de novo assembly. (E) any combination of method (A-D). Illustration from [128]

Read pairs

Recall that x denotes the fragment length and μ, σ denote the mean and standard deviation of the fragment length distribution $f(x)$. The distance between mates can be used to detect variants; if reads align too close or far away to each other on the reference genome, it might suggest that a variant is present in the sequenced genome. We refer to read pairs mapping with an anomalous fragment length as being *discordant*. If they map with an approximate expected distance, we refer to them as *concordant*. Notice that the relative orientations of the aligned read pairs can also be used to find inversions but we will limit the discussion to detection of insertions and deletions using only the fragment length. The exact cutoff for abnormal fragment length is dependent on the application. The advantage with RP approaches is that they can span over, and find SVs in repetitive regions such as duplications, CNVs and insertions or deletions in repetitive regions, where split-read, softclipped or coverage based methods have difficulties mapping reads. A disadvantage is that only an approximate breakpoint location can be given using read pairs information. The parameters μ, σ of a library also changes the landscape of what SV sizes can be detected. Larger μ can span larger variants and smaller σ gives more power to detect smaller variants. However in sequencing libraries, μ and σ is positively correlated. A quantitative study from [6] showed that libraries with larger fragment lengths are better for approximative detection of SV's, whereas smaller ones are better for accurate localization.

Proposed methods Numerous structural variation algorithms have been proposed using RP to detect variants. The usage of information from fragment length distances among tools varies. In many cases, a simple cutoff $\mu \pm k\sigma,\ k \in R$ is used to classify reads as supporting a variant [18, 7, 38], which means that only significantly deviating fragment lengths are used for detection. The use of only such read pairs may simplify the computational power needed but it sacrifices the the sensitivity and power as studied in [65]. There are also tools with a statistical model/approach that utilizes also the concordant reads. [65] looks at the mean fragment length of both concordant and discordant reads over positions. This allows them to classify insertions and deletions based on statistical significant deviation of μ_p (the position specific mean fragment length) from μ. This method finds more and smaller variants compared to methods that uses only discordant reads, as they use more information. However, due to the occurrence of heterozygous variants it is unfeasible to include all read pairs spanning a position in the statistical test over a position — only reads that has enough overlap and similar insert size are grouped together. This is done to increase the likelihood that the read pairs come from the same copy. [33] models the number of discordant and concordant read pairs over a region as a following a binomial distribution. This model allows them to statistically classify insertions and deletions based on significant accumulation of discordant read pairs to what is expected under the null distribution. However, any binary classification causes loss of information, thus statistical power, as they do not tell how much above or below the cutoff a value is. This is studied in [29], where they conclude that under a normal distribution, 100 continuous observations are statistically equivalent to 158 binary observations for the best possible "cut point", which is the mean. The loss of information becomes worse the further away the cut point is from the mean. As the fragment length distribution often is approximately normal, the result can be roughly applied to observing fragment lengths (with cut points at $\mu \pm k\sigma$). Another approach has been with non parametric testing of the distribution over a region, *e.g.*, the Kolmogorov-Smirnov test [54].

[97] showed that a common null-hypothesis used for testing variants based on RP is wrong and formulated an updated null-hypothesis based on the Lander-Waterman model: Let $\bar{x}$ denote the average fragment length observed over a position p on the reference sequence and, as before, μ be the mean of the library fragment length distribution f. A commonly used null-hypothesis is then $H_0 : \bar{x} = \mu$ [18, 65, 33], and statistical testing of variants are performed using H_0. That is, a test showing if $\bar{x}$ deviates significantly from μ over p. [97] claims that this null-hypothesis is wrong because μ is not the expected "local" mean over a given position p. If a denotes the minimum number of base pairs that needs to be aligned on each side of a position, and μ_p, σ_p the position specific mean and standard deviation, they showed that

Result 1. *We have* $\mu_p \approx \mu + \frac{\sigma^2}{\mu - \left(2q+1\right)}$ *given* $f \sim N(\mu, \sigma)$ *and i.i.d. observations of fragment lengths.*

Result 2. *We have* $\sigma_p \approx \sqrt{\sigma^2 - \frac{\sigma^4}{\left(\mu-(2q+1)\right)^2}}$ *given* $f \sim N(\mu, \sigma)$ *and i.i.d. observations of fragment lengths.*

where these formulas are based on simplified assumptions but are accurate in practice. The accuracy decreases the bigger the truncation of a fitted normal distribution to $f(x)$ has (enough probability density is on the positive x-axis). *E.g.*, if $\sigma > \mu$, then Result 2 would be undefined in the real domain. Furthermore, these formulas are accurate if all read pairs over a position is included in $\bar{x}$. In practice, read pairs included in $\bar{x}$ are method specific. For example, Breakdancer and Ulysses only includes x if ($x < \mu - k\sigma$ or $x > \mu + k\sigma$) and Clever select read pairs with enough overlap and similar fragment lengths. However, the updated null-hypothesis gives better results for statistical approaches such as in Clever, and the difference increases with increasing σ given fixed μ [97].

Older studies such as [6, 90] used simulated read pairs (from Lander-Waterman model) with the aim to identify parameters such as the coverage and fragment length distribution and their role in how they change the number of SVs detected using read pairs. But the library sequencing techniques are known to poorly follow this distribution [118, 82].

Split reads/softclipped reads

Split and softclipped read methods (SR/SO) consider reads that have not been aligned consistently to the reference. SR approaches consider reads where at least two parts of a read are aligned to different locations, see Figure 4.2. SO methods use reads that have only one part aligned, with the the rest of bases unaligned (called *softclipped* bases). As a reasonable read aligner will break the alignment at (or in close proximity to) the breakpoint, these approaches provide more accurate breakpoint predictions compared to methods using RP. Current methods using SR/SO require the parts of reads to be aligned uniquely [102, 116, 126, 1, 47, 37]. The requirement of uniqueness needs longer read length than RP methods as parts of reads are trivially shorter sequences than the reads, therefore less likely to be unique. As mentioned in Section 4.3, requiring unique alignments is also the disadvantage for detecting repetitive SVs, or SVs located in repetitive regions, compared to RP approaches as the repetitive regions can distort the alignments. Hence, they are most powerful on unique regions of the reference genome [5]. This is a significant problem as there are studies such as [124] showing that SV breakpoints are not distributed uniformly but are more often found within or close to repeat elements and regions of low complexity. However, with the increasing read length due to advancements in sequence technologies, these methods have potential to become more powerful as a read and its split parts are more likely more likely be long enough for unique alignment. Also, insertions that are longer than the fragment length, thus undetectable by RP approaches, can be discovered with SR/SO methods.

Read depth

Read depth (see Figure 4.2) is mainly used to infer CNVs [128]. The general assumption is that coverage on the reference genome increase/decrease in regions with different number of copies of a sequence in relation to the average coverage. After reads from the donor have been aligned to the genome, the general approach among RD based methods is to first infer and normalize the sequencing coverage across the genome. In a second step, the copy number is inferred to determine gain or loss compared to the reference. A common approach is to infer the copy number assuming that coverage follows a Poisson distribution [2, 122, 49], with various modifications to adjust for the overdispersion seen in real data. There are also non parametric models based om Hidden Markov Models [125] or iterative binning of coverage depth in regions using Baysian Information Criterion [121].

The increased variation in sequencing coverage from intrinsic properties of the genome such as GC bias, and local sequencing error rates from error prone sequence motifs [27], makes modeling of read depth difficult. In addition, reads that are aligned to multiple locations greatly affects all methods. Using only unique alignments, or randomly assign a location both have flaws as discussed in [128]. However, RD based methods can give information about exact copy numbers as opposed to RP and SR approaches.

Assembly

As [65] noted, a large fraction of the false positive predictions of SVs are due to misalignments. The performance of the alignment based methods are therefore dependent on the accuracy of the alignments. It is also interesting to know how sensitive the algorithms are to misaligned reads, furthermore, how much results varies with different aligners or specific alignment parameters. It is not unlikely that SV calling methods are sensitive to alignments as alignments showed to affect results significantly for scaffolding [41].

Reconstructing the donor genome (or at least a approximate graph representation of it due to read errors, uncovered regions and repeats) from the reads by assembly overcomes the mapping step — thus the spurious signals from mapping errors. One can immediately examine the assembly graph for structures that are formed by SVs. In case of insertions in the donor genome, the inserted sequence may be represented as a path in the assembly graph and can therefore be provided as output, which is not possible for alignment based methods unless an additional assembly step are incorporated. Assembly based SV detection without a read alignment step given in [44]. They perform a *de novo* assembly using a DBG constructed from multiple samples to detect local graph structures such as a bubbles (branching and convergence of two paths of length k) suggesting a SNPs or "path divergence"-sites (branching and convergence of two paths with unequal lengths) suggesting insertions or deletions. [77] uses a similar setup, with a graph constructed from several samples, but a different method relying on a Poisson mixture model for copy number (CNV) estimation of paths (contigs).

The *de novo* approach offers the advantage of being unbiased by alignments but the use of DBG structure without additional read pair information makes it challenging to detect and "untangle" variants occurring in repetitive regions in the genome. [91] provides an assembly method specialized to detect insertions by comparing the DBG against the reference sequence and performing assembly on the DBG where novel inserted sequence is detected. One of the big challenges with the *de novo* assembly approach is the difficulty in producing high quality assemblies.

Combined methods

As each of the information sources RP/SR/SO/RD/AS has their advantages and disadvantages, a combined approach seems reasonable. There exists a plethora of methods using combined information sources. A common combination is to use softclipped, split or read pair information to identify approximate breakpoint positions which will be used as marker regions where local assemblies will be performed by taking the softclipped reads together with their mates [116, 117, 86]. After contigs in each region are formed, they are aligned back to the reference with the hope that they will align over the breakpoint of interest. One part will map upstream to the breakpoint and the other part will dictate what type of SV that has occurred after the break point. The advantage of these methods is that they can call more complex variants such as translocations and relocations where a piece of sequence has been cut and pasted on another chromosome — these are variants that will be called as one deletion and one insertion separately, if detected at all, with other approaches. There also exists work on combining several variant calling methods to find consensus break points where local assemblies will be performed [120].

Chapter 5

Present investigations

All included papers in this thesis concentrates on applications using read pairs as input, where the fragment length distribution plays a key role. Paper I-III involves method development for the scaffolding problem, with paper I and III focusing on specific sub-problems around scaffolding. Paper IV builds on the model in paper I and extend it to apply in other scenarios such as for structural variant detection.

I: The first paper shows an improved model to estimate gap sizes, hence contig placement, in the scaffolding problem using read pair data and the fragment length distribution. The assumption of normally distributed data enables derivation of an analytical expression making computations instant.

II: The second paper introduces a new scaffolder that can scaffold large genomes fast. The main idea is that looking at other information from links between contigs, such as their distribution, might guide the scaffolding better than the commonly employed technique of maximizing the number of links. The presented scaffolder heuristically splits the scaffolding problem into two steps. The first step scaffolds larger contigs using scoring derived from statistical testing and characteristics of link distribution. In a second step, smaller contigs are scaffolded to fill gaps, or link together, the scaffolds formed by larger contigs. This step explores the sparse graph structure of the graph and uses a breadth first search to choose the best path.

III: In the third paper we account for, and model, paired-end-contamination in mate-pair libraries when scaffolding. We show that the contamination provides useful short range link information of contigs and state an Integer Linear Program (ILP) to find correct order and orientation of contigs. With this method we see significant improvement over the method presented in paper II, as well as other integrated and standalone scaffolders. Our experiments show the impact that PE-contamination has on scaffolders that does not model this. We observe an increased number of misassemblies, more conservative scaffolding, and inflated assembly sizes with increased levels of PE-contamination.

IV: The fourth paper generalizes the model in paper I for other applications. A key application is detection structural variants using fragment length information. We use our model to show that a commonly assumed null-hypothesis distribution used to detect structural variants is subject to bias, and we give a null-hypothesis that better fits data given common assumptions. The discrepancy between the two null-hypotheses is bigger for insert size libraries with larger variance. We compare results between the two null-hypotheses and show that structural variation callers based on statistical models can benefit from applying our corrected null-hypothesis.

Chapter 6

Conclusions and Future perspectives

I have presented an overview of current research on scaffolding and structural variation detection. The two biological problems of "order contigs as they appear on the genome" and "find the locations where two genomes differ" are general and not necessarily bound to the type of data presented in this thesis (reads from sequencing, for examples of other data, see Chapter 2.4). NGS data is however currently one of the most promising information sources to tackle these problems, with the sequencing techniques continuing to improve in, *e.g.*, read length, throughput, error rate, and more uniform coverage.

6.1 Sequencing technology

As sequencing improves we will most likely continue to see improvements from new methods adapted to the latest sequencing data. Third generation sequencing techniques[1] offering significantly longer reads continue to increase their throughput and are already being frequently used for assembly of bacterial genomes [52, 56, 51]. However, the relatively high cost and error rate compared to NGS methods limits these sequencing techniques from assembling larger genomes. Therefore, a recent common trend in genome assembly has been to develop hybrid assembly methods combining high error level long reads with low error rate short reads from NGS data. Numerous algorithms developed, for a summary see [51].

Although there has been work on SV detection with third generation sequencing, *e.g.*, [16], short read sequencing from NGS methods to this day continues to be the leading sequence technology for detection of structural variation which makes most

[1] The exists several definitions of what "Third generation" sequencing is, hence its difference to Next generation varies. Here we simply observe that some of the techniques that have been labeled as "Third generation" output reads substantially longer than common "Next generation"-techniques.

methods being developed for this type of data. An explanation to this could be that a lot of the attention in SV detection community is directed toward variants in the human genome, and its role in disease [110, 108]. This often requires several patients analyzed and sequenced so far making NGS data often the only affordable solution.

6.2 Integrated methods

Using several information sources, *e.g.*, different sequencing techniques, is a common approach for data inference algorithms. Another methodology is to use an integrated method that combines several algorithms to give a consensus solution. This is the case also in bioinformatics. With a plethora of algorithms using different information to call variants, several integrated methods have been proposed giving consensus variant prediction from output of several algorithms [72, 79, 120, 70]. Another reason to use an integrative approach in particular for SV-detection is, as [57] points out, that algorithms are currently limited in both type and size of variants they can predict. An integrative method that can call variants of several types and sizes would be preferable to manually running several algorithms and derive consensus variants — especially if the interesting variant types and sizes is not known beforehand.

Several integrated solutions to infer (assemble) consensus genome assemblies (*i.e.*, meta-assemblers) have also been proposed [115, 106, 76, 71] but assembling genome assemblies is likely to be, at least in theory, as difficult as sequence assembly itself. Nevertheless, integrated methods seems to be a promising way forward also here, unless sequence technology improves significantly.

6.3 Gold standard benchmark data

The genome assembly community now has several sequencing data sets for high quality finished genomes of various sizes that together with software such as QUAST can be used to benchmark algorithms on. Several challenges on “gold standard” data have also been proposed [101, 62, 25, 12] which helps the development of methods. This is still not common in SV detection, where a lack of gold standard data hampers evaluation and benchmarking of existing tools. This makes it difficult to give recommendations for the use of specific callers in specific circumstances, as [57] concludes. Creating “gold standard” benchmark data is therefore still a hot topic in a SV detection and, more recently, public challenges have been proposed [28].

Bibliography

[1] Alexej Abyzov and Mark Gerstein. AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision. *Bioinformatics*, 27(5):595–603, 2011.

[2] Alexej Abyzov, Alexander E. Urban, Michael Snyder, and Mark Gerstein. CNVnator: An approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Research*, 21(6):974–984, 2011.

[3] Can Alkan, Bradley P. Coe, and Evan E. Eichler. Genome structural variation discovery and genotyping. *Nat Rev Genet*, 12(5):363–376, 2011.

[4] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, Alexey V Pyshkin, Alexander V Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A Alekseyev, and Pavel A Pevzner. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 19(5):455–477, 2012.

[5] Christoph Bartenhagen and Martin Dugas. Robust and exact structural variation detection with paired-end and soft-clipped alignments: SoftSV compared with eight algorithms. *Briefings in Bioinformatics*, 2015.

[6] Ali Bashir, Stanislav Volik, Colin Collins, Vineet Bafna, and Benjamin J. Raphael. Evaluation of paired-end sequencing strategies for detection of genome rearrangements in cancer. *PLoS Comput Biol*, 4(4):e1000051, 2008.

[7] Derek M. Bickhart, Jana L. Hutchison, Lingyang Xu, Robert D. Schnabel, Jeremy F. Taylor, James M. Reecy, Steven Schroeder, Curt P. Van Tassell, Tad S. Sonstegard, and George E. Liu. RAPTR-SV: a hybrid method for the detection of structural variants. *Bioinformatics*, 31(13), 2015.

[8] Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In Timo Lepistö and Arto Salomaa, editors, *Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Berlin Heidelberg, 1988.

[9] M. Boetzer, C.V. Henkel, H.J. Jansen, D. Butler, and W. Pirovano. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics*, 27(4):578–579, 2011.

[10] Marten Boetzer and Walter Pirovano. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. *BMC Bioinformatics*, 15(1):211, 2014.

[11] Emanuele Bosi, Beatrice Donati, Marco Galardini, Sara Brunetti, Marie-France Sagot, Pietro Lió, Pierluigi Crescenzi, Renato Fani, and Marco Fondi. MeDuSa: a multi-draft based scaffolder. *Bioinformatics*, 31(15), 2015.

[12] Keith Bradnam, Joseph Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanc Birol, Sebastien Boisvert, Jarrod Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi Chou, Jacques Corbeil, Cristian Del Fabbro, T Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno Fonseca, Ganeshkumar Ganapathy, Richard Gibbs, Sante Gnerre, Elenie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph Hiatt, and Isaac Ho. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1):10, 2013.

[13] Guy Bresler, Ma'ayan Bresler, and David Tse. Optimal assembly for high throughput shotgun sequencing. *BMC Bioinformatics*, 14(Suppl 5):S18, 2013.

[14] Ma'ayan Bresler, Sara Sheehan, Andrew H. Chan, and Yun S. Song. Telescoper: de novo assembly of highly repetitive regions. *Bioinformatics*, 28(18):i311–i317, 2012.

[15] 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. *Nat Biotech*, 31(12):1119–1125, 2013.

[16] Mark J. P. Chaisson, John Huddleston, Megan Y. Dennis, Peter H. Sudmant, Maika Malig, Fereydoun Hormozdiari, Francesca Antonacci, Urvashi Surti, Richard Sandstrom, Matthew Boitano, Jane M. Landolin, John A. Stamatoyannopoulos, Michael W. Hunkapiller, Jonas Korlach, and Evan E. Eichler. Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, 517(7536):608–611, 01 2015.

[17] A Chateau and R Giroudeau. A complexity and approximation framework for the maximization scaffolding problem. *Theoretical Computer Science*, 2015.

[18] K. Chen, J. W. Wallis, M. D. McLellan, D. E. Larson, J. M. Kalicki, C. S. Pohl, S.D. McGrath, M.C. Wendl, Q. Zhang, D. P. Locke, X. Shi, R. S. Fulton, T.J. Ley, R.K. Wilson, L. Ding, and E. R. Mardis. BreakDancer: an

algorithm for high-resolution mapping of genomic structural variation. *Nature Methods*, 6(9):677–681, 2009.

[19] Matteo Chiara, Graziano Pesole, and David S. Horner. SVM2: an improved paired-end-based tool for the detection of small genomic structural variations using high-throughput single-genome resequencing data. *Nucleic Acids Research*, 40(18):e145, 2012.

[20] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. In *WABI*, volume 7534 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2012.

[21] Scott C. Clark, Rob Egan, Peter I. Frazier, and Zhong Wang. ALE: a generic assembly likelihood evaluation framework for assessing the accuracy of genome and metagenome assemblies. *Bioinformatics*, 29(4):435–443, 2013.

[22] Donald F Conrad and Matthew E Hurles. The population genetics of structural variation. *Nat Genet*, 39(7 Suppl):30–36, 2007.

[23] A. Dayarian, T.P. Michael, and A.M. Sengupta. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11:345, 2010.

[24] Nilgun Donmez and Michael Brudno. SCARPA: scaffolding reads with practical algorithms. *Bioinformatics*, 29(4):428–434, 2013.

[25] D. Earl, K. Bradnam, J. St John, A. Darling, D. Lin, J. Fass, H.O. Yu, V. Buffalo, D.R. Zerbino, M. Diekhans, N. Nguyen, P.N. Ariyaratne, W.K. Sung, Z. Ning, M. Haimel, J.T. Simpson, N.A. Fonseca, I. Birol, T.R. Docking, I.Y. Ho, D.S. Rokhsar, R. Chikhi, D. Lavenier, G. Chapuis, D. Naquin, N. Maillet, M.C. Schatz, D.R. Kelley, A.M. Phillippy, S. Koren, S.P. Yang, W. Wu, W.C. Chou, A. Srivastava, T.I. Shaw, J.G. Ruby, P. Skewes-Cox, M. Betegon, M.T. Dimon, V. Solovyev, I. Seledtsov, P. Kosarev, D. Vorobyev, R. Ramirez-Gonzalez, R. Leggett, D. MacLean, F. Xia, R. Luo, Z. Li, Y. Xie, B. Liu, S. Gnerre, I. MacCallum, D. Przybylski, F.J. Ribeiro, S. Yin, T. Sharpe, G. Hall, P.J. Kersey, R. Durbin, S.D. Jackman, J.A. Chapman, X. Huang, J.L. DeRisi, M. Caccamo, Y. Li, D.B. Jaffe, R.E. Green, D. Haussler, I. Korf, and B. Paten. Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome Res*, 21(12):2224–2241, 2011.

[26] Jack Edmonds. Paths, trees, and flowers. In Ira Gessel and Gian-Carlo Rota, editors, *Classic Papers in Combinatorics*, Modern Birkhäuser Classics, pages 361–379. Birkhäuser Boston, 1987.

[27] Robert Ekblom, Linnea Smeds, and Hans Ellegren. Patterns of sequencing coverage bias revealed by ultra-deep sequencing of vertebrate mitochondria. *BMC Genomics*, 15(1):467, 2014.

[28] Adam D Ewing, Kathleen E Houlahan, Yin Hu, Kyle Ellrott, Cristian Caloian, Takafumi N Yamaguchi, J Christopher Bare, Christine P'ng, Daryl Waggott, Veronica Y Sabelnykova, ICGC-TCGA DREAM Somatic Mutation Calling Challenge participants, Michael R Kellen, Thea C Norman, David Haussler, Stephen H Friend, Gustavo Stolovitzky, Adam A Margolin, Joshua M Stuart, and Paul C Boutros. Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection. *Nat Meth*, 12(7):623–630, 07 2015.

[29] Valerii Fedorov, Frank Mannino, and Rongmei Zhang. Consequences of dichotomization. *Pharmaceutical Statistics*, 8(1):50–61, 2009.

[30] Lars Feuk, Andrew R. Carson, and Stephen W. Scherer. Structural variation in the human genome. *Nat Rev Genet*, 7(2):85–97, 2006.

[31] S. Gao, W.-K. Sung, and N. Nagarajan. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *J Comput Biol*, 18(11):1681–1691, 2011.

[32] Song Gao, Denis Bertrand, and Niranjan Nagarajan. OPERA-LG: Efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees. *bioRxiv*, 2015.

[33] Alexandre Gillet-Markowska, Hugues Richard, Gilles Fischer, and Ingrid Lafontaine. Ulysses: accurate detection of low-frequency structural variations in large insert-size sequencing libraries. *Bioinformatics*, 31(6):801–808, 2015.

[34] Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J. Ribeiro, Joshua N. Burton, Bruce J. Walker, Ted Sharpe, Giles Hall, Terrance P. Shea, Sean Sykes, Aaron M. Berlin, Daniel Aird, Maura Costello, Riza Daza, Louise Williams, Robert Nicol, Andreas Gnirke, Chad Nusbaum, Eric S. Lander, and David B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[35] A.A. Gritsenko, J.F. Nijkamp, M.J. Reinders, and D. de Ridder. GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, 28(11):1429–1437, 2012.

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

[37] Steven N Hart, Vivekananda Sarangi, Raymond Moore, Saurabh Baheti, Jaysheel D Bhavsar, Fergus J Couch, and Jean-Pierre A Kocher. SoftSearch: Integration of multiple sequence features to identify breakpoints of structural variations. *PLoS ONE*, 8(12):e83356, 2013.

[38] Matthew Hayes, Yoon Soo Pyon, and Jing Li. A model-based clustering method for genomic structural variant prediction and genotyping using paired-end sequencing data. *PLoS ONE*, 2012.

[39] Pinar Heggernes, Dieter Kratsch, and Daniel Meister. Bandwidth of bipartite permutation graphs in polynomial time. In Eduardo Sany Laber, Claudson Bornstein, Loana Tito Nogueira, and Luerbio Faria, editors, *LATIN 2008: Theoretical Informatics*, volume 4957 of *Lecture Notes in Computer Science*, pages 216–227. Springer Berlin Heidelberg, 2008.

[40] Martin Hunt, Taisei Kikuchi, Mandy Sanders, Chris Newbold, Matthew Berriman, and Thomas Otto. REAPR: a universal tool for genome assembly evaluation. *Genome Biology*, 14(5):R47, 2013.

[41] Martin Hunt, Chris Newbold, Matthew Berriman, and Thomas Otto. A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*, 15(3):R42, 2014.

[42] Daniel H Huson, Knut Reinert, and Eugene W Myers. The greedy path-merging algorithm for contig scaffolding. *J ACM*, 49(5):603–615, 2002.

[43] Illumina. Data processing of Nextera mate pair reads on Illumina sequencing platforms. Technical note, 2012.

[44] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat Genet*, 44(2):226–232, 2012.

[45] S.D. Jackman, Raymond A.G, and I. Birol. Scaffolding a genome sequence assembly using ABySS. https://github.com/sjackman/abyss-paper/blob/master/scaffold.md, 2012.

[46] H. Jiang, R. Lei, S.W. Ding, and S. Zhu. Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. *BMC Bioinformatics*, 15(182), 2014.

[47] Yue Jiang, Yadong Wang, and Michael Brudno. PRISM: Pair read informed split read mapping for base-pair level detection of insertion, deletion and structural variants. *Bioinformatics*, 2012.

[48] Rei Kajitani, Kouta Toshimoto, Hideki Noguchi, Atsushi Toyoda, Yoshitoshi Ogura, Miki Okuno, Mitsuru Yabana, Masahira Harada, Eiji Nagayasu, Haruhiko Maruyama, Yuji Kohara, Asao Fujiyama, Tetsuya Hayashi, and Takehiko Itoh. Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads. *Genome Research*, 2014.

[49] Günter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arné Clevert, Andreas Mitterecker, Ulrich Bodenhofer, and Sepp Hochreiter. cn.MOPS: mixture of Poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate. *Nucleic Acids Research*, 2012.

[50] Mikhail Kolmogorov, Brian Raney, Benedict Paten, and Son Pham. Ragout—a reference-assisted assembly tool for bacterial genomes. *Bioinformatics*, 30(12):i302–i309, 2014.

[51] Sergey Koren and Adam M Phillippy. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Current Opinion in Microbiology*, 23:110 – 120, 2015. Host–microbe interactions: bacteria * Genomics.

[52] Miriam Land, Loren Hauser, Se-Ran Jun, Intawat Nookaew, Michael R Leuze, Tae-Hyuk Ahn, Tatiana Karpinets, Ole Lund, Guruprased Kora, Trudy Wassenaar, Suresh Poudel, and David W Ussery. Insights from 20 years of bacterial genome sequencing. *Functional & Integrative Genomics*, 15(2):141–161, 2015.

[53] Roy Lederman. Building approximate overlap graphs for DNA Building approximate overlap graphs for DNA assembly using random-permutations-based search. http://www.cs.yale.edu/publications/techreports/tr1470.pdf, 2012.

[54] Seunghak Lee, Fereydoun Hormozdiari, Can Alkan, and Michael Brudno. MoDIL: detecting small indels from clone-end sequencing with mixtures of distributions. *Nat Meth*, 6(7):473–474, 2009.

[55] H. Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv:1303.3997*, 2013.

[56] Yu-Chieh Liao, Shu-Hung Lin, and Hsin-Hung Lin. Completing bacterial genome assemblies: strategy and performance comparisons. *Scientific Reports*, 5:8747 EP –, 03 2015.

[57] Ke Lin, Sandra Smit, Guusje Bonnema, Gabino Sanchez-Perez, and Dick de Ridder. Making the difference: integrating structural variation detection tools. *Briefings in Bioinformatics*, 2014.

[58] James Lindsay, Hamed Salooti, Ion Mandoiu, and Alex Zelikovsky. ILP-based maximum likelihood genome scaffolding. *BMC Bioinformatics*, 15(Suppl 9):S9, 2014.

[59] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology*, 2012:11, 2012.

[60] I MacCallum, D Przybylski, S Gnerre, J Burton, I Shlyakhter, A Gnirke, J Malek, K McKernan, S Ranade, TP Shea, L Williams, S Young, C Nusbaum, and DB Jaffe. ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads. *Genome Biol*, 10(10):103, 2009.

[61] Jeffrey R. MacDonald, Robert Ziman, Ryan K. C. Yuen, Lars Feuk, and Stephen W. Scherer. The Database of Genomic Variants: a curated collection of structural variation in the human genome. *Nucleic Acids Research*, 42(D1):D986–D992, 2014.

[62] Tanja Magoc, Stephan Pabinger, Stefan Canzar, Xinyue Liu, Qi Su, Daniela Puiu, Luke J. Tallon, and Steven L. Salzberg. GAGE-B: An evaluation of genome assemblers for bacterial organisms. *Bioinformatics*, 2013.

[63] Veli. Mäkinen, Djamal Belazzougui, Fabio Cunia, and Alexandru I. Tomescu. *Genome-Scale Algorithm Design.* Camebridge Journals, 2015.

[64] Igor Mandric and Alex Zelikovsky. ScaffMatch: Scaffolding algorithm based on maximum weight matching. *Bioinformatics*, 31(16), 2015.

[65] T. Marschall, I. Costa, Canzar S., Bauer M., G.W. Klau, A. Schliep, and A Schönhuth. CLEVER: Clique-Enumerating Variant Finder. *Bioinformatics*, 28(22):2875–2880, 2012.

[66] Paul Medvedev, Monica Stanciu, and Michael Brudno. Computational methods for discovering structural variation with next-generation sequencing. *Nat Meth*, 6(11s):S13–S20, 2009.

[67] Páll Melsted. Dear assemblers, we need to talk ... together. https://pmelsted.wordpress.com/2014/07/17/dear-assemblers-we-need-to-talk-together/, 2014.

[68] Jacob J Michaelson and Jonathan Sebat. forestSV: structural variant discovery through statistical learning. *Nat Meth*, 9(8):819–821, 2012.

[69] Ryan E. Mills, W. Stephen Pittard, Julienne M. Mullaney, Umar Farooq, Todd H. Creasy, Anup A. Mahurkar, David M. Kemeza, Daniel S. Strassler, Chris P. Ponting, Caleb Webber, and Scott E. Devine. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome Research*, 21(6):830–839, 2011.

[70] Takahiro Mimori, Naoki Nariai, Kaname Kojima, Mamoru Takahashi, Akira Ono, Yukuto Sato, Yumi Yamaguchi-Kabata, and Masao Nagasaki. isvp: an integrated structural variant calling pipeline from high-throughput sequencing data. *BMC Systems Biology*, 7(Suppl 6):S8, 2013.

[71] Hamid Mirebrahim, Timothy J. Close, and Stefano Lonardi. De novo meta-assembly of ultra-deep sequencing data. *Bioinformatics*, 31(12):i9–i16, 2015.

[72] Marghoob Mohiyuddin, John C. Mu, Jian Li, Narges Bani Asadi, Mark B. Gerstein, Alexej Abyzov, Wing H. Wong, and Hugo Y.K. Lam. Metasv: an accurate and integrative structural-variant caller for next generation sequencing. *Bioinformatics*, 31(16):2741–2744, 2015.

[73] A Mortazavi, E Schwarz, B Williams, L Schaeffer, I Antoshechkin, B Wold, and P Sternberg. Scaffolding a Caenorhabditis nematode genome with RNA-seq. *Genome Res*, 20(12):1740–1747, 2010.

[74] Eugene W. Myers, Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo, Michael J. Flanigan, Saul A. Kravitz, Clark M. Mobarry, Knut H. J. Reinert, Karin A. Remington, Eric L. Anson, Randall A. Bolanos, Hui-Hsien Chou, Catherine M. Jordan, Aaron L. Halpern, Stefano Lonardi, Ellen M. Beasley, Rhonda C. Brandon, Lin Chen, Patrick J. Dunn, Zhongwu Lai, Yong Liang, Deborah R. Nusskern, Ming Zhan, Qing Zhang, Xiangqun Zheng, Gerald M. Rubin, Mark D. Adams, and J. Craig Venter. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, 2000.

[75] N Nagarajan, TD Read, and M Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24:1229–1235, 2008.

[76] Jurgen Nijkamp, Wynand Winterbach, Marcel van den Broek, Jean-Marc Daran, Marcel Reinders, and Dick de Ridder. Integrating genome assemblies with maia. *Bioinformatics*, 26(18):i433–i439, 2010.

[77] Jurgen F. Nijkamp, Marcel A. van den Broek, Jan-Maarten A. Geertman, Marcel J. T. Reinders, Jean-Marc G. Daran, and Dick de Ridder. De novo detection of copy number variation by co-assembly. *Bioinformatics*, 28(24):3195–3202, 2012.

[78] Jared O'Connell, Ole Schulz-Trieglaff, Emma Carlson, Matthew M. Hims, Niall A. Gormley, and Anthony J. Cox. NxTrim: optimized trimming of Illumina mate pair reads. *Bioinformatics*, 31(12), 2015.

[79] Hemang Parikh, Hariharan Iyer, Desu Chen, Mark Pratt, Gabor Bartha, Noah Spies, Wolfgang Losert, Justin M. Zook, and Marc L. Salit. svclassify: a method to establish benchmark structural variant calls. *bioRxiv doi: 10.1101/019372*, 2015.

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

[81] M. Pop, D.S. Kosack, and S.L. Salzberg. Hierarchical scaffolding with Bambus. *Genome Res*, 14(1):149–159, 2004.

[82] Maria S. Poptsova, Irina A. Il'icheva, Dmitry Yu. Nechipurenko, Larisa A. Panchenko, Mingian V. Khodikov, Nina Y. Oparina, Robert V. Polozov, Yury D. Nechipurenko, and Sergei L. Grokhovsky. Non-random DNA fragmentation in next-generation sequencing. *Sci. Rep.*, 4, 2014.

[83] Pjotr Prins, Joep de Ligt, Artem Tarasov, Ritsert C Jansen, Edwin Cuppen, and Philip E Bourne. Toward effective software solutions for big biology. *Nat Biotech*, 33(7):686–687, 2015.

[84] Andrey D. Prjibelski, Irina Vasilinetc, Anton Bankevich, Alexey Gurevich, Tatiana Krivosheeva, Sergey Nurk, Son Pham, Anton Korobeynikov, Alla Lapidus, and Pavel A. Pevzner. ExSPAnder: a universal repeat resolver for DNA fragment assembly. *Bioinformatics*, 30(12):i293–i301, 2014.

[85] Michael A Quail, Miriam Smith, Paul Coupland, Thomas D Otto, Simon R Harris, Thomas R Connor, Anna Bertoni, Harold P Swerdlow, and Yong Gu. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics*, 13:341–341, 2012.

[86] Aaron R Quinlan, Royden A Clark, Svetlana Sokolova, Mitchell L Leibowitz, Yujun Zhang, Matthew E Hurles, Joshua C Mell, and Ira M Hall. Genome-wide mapping and assembly of structural variant breakpoints in the mouse genome. *Genome Research*, 20(5):623–635, 2010.

[87] Atif Rahman and Lior Pachter. CGAL: computing genome assembly likelihoods. *Genome Biology*, 14(1):R8, 2013.

[88] FJ Ribeiro, D Przybylski, S Yin, T Sharpe, S Gnerre, A Abouelleil, AM Berlin, A Montmayeur, TP Shea, BJ Walker, SK Young, C Russ, C Nusbaum, I MacCallum, and DB Jaffe. Finished bacterial genomes from shotgun sequence data. *Genome Res*, 22(11):2270–2277, 2012.

[89] DC Richter, SC Schuster, and DH Huson. OSLay: optimal syntenic layout of unfinished assemblies. *Bioinformatics (Oxford, England)*, 23(13):1573–1579, 2007.

[90] Anna Ritz, Ali Bashir, and Benjamin J. Raphael. Structural variation analysis with strobe reads. *Bioinformatics*, 26(10):1291–1298, 2010.

[91] Guillaume Rizk, Anaïs Gouin, Rayan Chikhi, and Claire Lemaitre. MindTheGap : integrated detection and assembly of short and long insertions. *Bioinformatics*, 2014.

[92] R. S. Roy, Chen K. C., A. M. Sengupta, and Schliep A. SLIQ: Simple linear inequalities for efficient contig scaffolding. *Journal of Computational Biology*, 19(10):1162–1175, 2012.

[93] Subrata Saha and Sanguthevar Rajasekaran. Efficient and scalable scaffolding using optical restriction maps. *BMC Genomics*, 15(Suppl 5):S5, 2014.

[94] K. Sahlin. BESST_RNA: Scaffolding of genomic assemblies with rna seq data. https://github.com/ksahlin/BESST_RNA, 2015.

[95] K. Sahlin, N. Street, J. Lundeberg, and L. Arvestad. Improved gap size estimation for scaffolding algorithms. *Bioinformatics*, 28(17):2215–2222, 2012.

[96] Kristoffer Sahlin, Rayan Chikhi, and Lars Arvestad. Genome scaffolding with PE-contaminated mate-pair libraries. *bioRxiv doi: 10.1101/025650*, 2015.

[97] Kristoffer Sahlin, Mattias Frånberg, and Lars Arvestad. Correcting bias from stochastic insert size in read pair data — applications to structural variation detection and genome assembly. *bioRxiv doi: 10.1101/023929*, 2015.

[98] Kristoffer Sahlin, Francesco Vezzi, Björn Nystedt, Joakim Lundeberg, and Lars Arvestad. BESST - Efficient scaffolding of large fragmented assemblies. *BMC Bioinformatics*, 15(1):281, 2014.

[99] L. Salmela, V. Mäkinen, N. Välimäki, J. Ylinen, and E. Ukkonen. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, 27(23):3259–3265, 2011.

[100] Leena Salmela, Kristoffer Sahlin, Veli Mäkinen, and AlexandruI. Tomescu. Gap filling as exact path length problem. In Teresa M. Przytycka, editor, *Research in Computational Molecular Biology*, volume 9029 of *Lecture Notes in Computer Science*, pages 281–292. Springer International Publishing, 2015.

[101] S.L. Salzberg, A.M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T.J. Treangen, M.C. Schatz, A.L. Delcher, M. Roberts, G. Marcais, M. Pop, and J.A. Yorke. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Res*, 22(3):557–567, 2012.

[102] Jan Schröder, Arthur Hsu, Samantha E Boyle, Geoff Macintyre, Marek Cmero, Richard W Tothill, Ricky W Johnstone, Mark Shackleton, and Anthony T Papenfuss. Socrates: identification of genomic rearrangements in tumour genomes by re-aligning soft clipped reads. *Bioinformatics*, 30(8):1064–1072, 2014.

[103] 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*, 2015.

[104] Jared T. Simpson and Richard Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, 22(3):549–556, 2012.

[105] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J.M. Jones, and İnanç Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[106] Hayssam Soueidan, Florence Maurier, Alexis Groppi, Pascal Sirand-Pugnet, Florence Tardy, Christine Citti, Virginie Dupuy, and Macha Nikolski. Finishing bacterial genome assemblies with mix. *BMC Bioinformatics*, 14(Suppl 15):S16, 2013.

[107] Frits C. R. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2(5):215–227, 1999.

[108] Paweł Stankiewicz and James R. Lupski. Structural Variation in the Human Genome and its Role in Disease. *Annual Review of Medicine*, 61(1):437–455, 2010.

[109] Haibao Tang, Xingtan Zhang, Chenyong Miao, Jisen Zhang, Ray Ming, James Schnable, Patrick Schnable, Eric Lyons, and Jianguo Lu. ALLMAPS: robust scaffold ordering based on multiple maps. *Genome Biology*, 16(1):3, 2015.

[110] The 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 11 2012.

[111] E Ukkonen. Approximate string matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992.

[112] Sebastiaan van Heesch, Wigard Kloosterman, Nico Lansu, Frans-Paul Ruzius, Elizabeth Levandowsky, Clarence Lee, Shiguo Zhou, Steve Goldstein, David Schwartz, Timothy Harkins, Victor Guryev, and Edwin Cuppen. Improving mammalian genome scaffolding using large insert mate-pair next-generation sequencing. *BMC Genomics*, 14(1):257, 2013.

[113] Irina Vasilinetc, Andrey D. Prjibelski, Alexey Gurevich, Anton Korobeynikov, and Pavel A. Pevzner. Assembling Short Reads from Jumping Libraries with Large Insert Sizes. *Bioinformatics*, 2015.

[114] F. Vezzi, G. Narzisi, and B. Mishra. Feature-by-feature–evaluating de novo sequence assembly. *PLoS ONE*, 7(2):e31002, 2012.

[115] Riccardo Vicedomini, Francesco Vezzi, Simone Scalabrin, Lars Arvestad, and Alberto Policriti. GAM-NGS: genomic assemblies merger for next generation sequencing. *BMC Bioinformatics*, 14(Suppl 7):S6, 2013.

[116] Jianmin Wang, Charles G Mullighan, John Easton, Stefan Roberts, Sue L Heatley, Jing Ma, Michael C Rusch, Ken Chen, Christopher C Harris, Li Ding, Linda Holmfeldt, Debbie Payne-Turner, Xian Fan, Lei Wei, David Zhao, John C Obenauer, Clayton Naeve, Elaine R Mardis, Richard K Wilson, James R Downing, and Jinghui Zhang. CREST maps somatic structural variation in cancer genomes with base-pair resolution. *Nat Meth*, 8(8):652–654, 2011.

[117] Neil I Weisenfeld, Shuangye Yin, Ted Sharpe, Bayo Lau, Ryan Hegarty, Laurie Holmes, Brian Sogoloff, Diana Tabbaa, Louise Williams, Carsten Russ, Chad Nusbaum, Eric S Lander, Iain MacCallum, and David B Jaffe. Comprehensive variation discovery in single human genomes. *Nat Genet*, 46(12):1350–1355, 2014.

[118] Michael Wendl and W Brad Barbazuk. Extension of Lander-Waterman theory for sequencing filtered DNA libraries. *BMC Bioinformatics*, 6(1):245, 2005.

[119] Joshua Wetzel, Carl Kingsford, and Mihai Pop. Assessing the benefits of using mate-pairs to resolve repeats in de novo short-read prokaryotic assemblies. *BMC Bioinformatics*, 12(1):95, 2011.

[120] Kim Wong, Thomas Keane, James Stalker, and David Adams. Enhanced structural variant and breakpoint detection using SVMerge by integration of multiple detection methods and local assembly. *Genome Biology*, 11(12):R128, 2010.

[121] Ruibin Xi, Angela G. Hadjipanayis, Lovelace J. Luquette, Tae-Min Kim, Eunjung Lee, Jianhua Zhang, Mark D. Johnson, Donna M. Muzny, David A. Wheeler, Richard A. Gibbs, Raju Kucherlapati, and Peter J. Park. Copy number variation detection in whole-genome sequencing data using the Bayesian information criterion. *Proceedings of the National Academy of Sciences*, 108(46):E1128–E1136, 2011.

[122] Chao Xie and Martti Tammi. CNV-seq, a new method to detect copy number variation using high-throughput sequencing. *BMC Bioinformatics*, 10(1):80, 2009.

[123] Wei Xue, Jiong-Tang Li, Ya-Ping Zhu, Guang-Yuan Hou, Xiang-Fei Kong, You-Yi Kuang, and Xiao-Wen Sun. L_RNA_scaffolder: scaffolding genomes with transcripts. *BMC Genomics*, 14(1):604, 2013.

[124] Lixing Yang, Lovelace J. Luquette, Nils Gehlenborg, Ruibin Xi, Psalm S. Haseley, Chih-Heng Hsieh, Chengsheng Zhang, Xiaojia Ren, Alexei Protopopov, Lynda Chin, Raju Kucherlapati, Charles Lee, and Peter J. Park. Diverse mechanisms of somatic structural variations in human cancer genomes. *Cell*, 153(4):919–929.

[125] C Yau, O Papaspiliopoulos, G O Roberts, and C Holmes. Bayesian Nonparametric Hidden Markov Models with application to the analysis of copy-number-variation in mammalian genomes. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 73(1):37–57, 2011.

[126] Kai Ye, Marcel H. Schulz, Quan Long, Rolf Apweiler, and Zemin Ning. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21):2865–2871, 2009.

[127] DR Zerbino and E Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18(5):821–829, 2008.

[128] Min Zhao, Qingguo Wang, Quan Wang, Peilin Jia, and Zhongming Zhao. Computational tools for copy number variation (CNV) detection using next-generation sequencing data: features and perspectives. *BMC Bioinformatics*, 14(Suppl 11):S1, 2013.

[129] Aleksey V. Zimin, Guillaume Marçais, Daniela Puiu, Michael Roberts, Steven L. Salzberg, and James A. Yorke. The MaSuRCA genome assembler. *Bioinformatics*, 29(21):2669–2677, 2013.