

The Sequence Reconstruction Problem

Angela Angeleska, Sabrina Kleessen, and Zoran Nikoloski

Abstract Despite recent advances, assembly of genomes from the high-throughput data generated by the next-generation sequencing (NGS) technologies remains one of the most challenging tasks in modern biology. Here we address the sequence reconstruction problem, whereby, for a given collection of subsequences or factors, one has to determine the set of sequences compliant with the collection. First, we give a brief review of sequencing technologies, along with an exposition of the advantages and shortcomings of the existing algorithmic approaches to sequence assembly. In addition, we enumerate some properties of subsequences, which have been overlooked in the existing heuristic solutions despite their effect on the quality of the assembly. We then give an overview of the sequence reconstruction problem from a language-theoretic perspective, and present a comprehensive review of theoretical results that may prove relevant to the genome assembly problem. Finally, we outline a new optimization-based formulation which casts the sequence reconstruction problem as a quadratic integer programming problem.

1 Introduction

Next-generation sequencing (NGS) technologies of ever-increasing quality offer the possibility to use the plethora of sequence data that results from them to assemble whole genomes. Currently, tremendous amounts of time, money, and computational resources are being invested in genome sequencing. The existing

A. Angeleska (✉)
The University of Tampa, Tampa, FL, USA
e-mail: aangeleska@ut.edu

S. Kleessen · Z. Nikoloski
Systems Biology and Mathematical Modeling Group, Max Planck Institute for Molecular Plant
Physiology – Golm, Potsdam, Germany
e-mail: Kleessen@mpimp-golm.mpg.de; Nikoloski@mpimp-golm.mpg.de

NGS technologies cannot be used to sequence entire genomes at once; in fact, only short reads (i.e., substrings) are sequenced, and these must then be assembled to form the original genome. The resulting reads differ in length, depending on the technology used. To increase the coverage, the genome to be investigated is first copied several times. These copies are broken up randomly into fragments, which are then sequenced to produce reads. The aim of genome assembly is then to obtain the entire genome sequence with the help of the overlaps of all the reads while taking into account the particularities of the NGS technology used.

The contribution of this chapter is threefold: (1) to provide a brief review and history of existing methods for genome assembly, while critically comparing and contrasting them with existing results from language theory; (2) to offer a source for interdisciplinary understanding of the genome assembly problem, by presenting a comprehensive overview of the relevant mathematical results and pointing out problems where mathematical endeavor may improve the existing solutions; and (3) to formulate an optimization-based approach that provides a different framework for the problem at hand.

2 A Brief Review of Sequencing Technologies

In this section, we present a short background on DNA and whole-genome sequencing. In addition, we present a brief historical overview of the commonly used sequencing technologies, including their advantages and drawbacks. For a detailed review of next-generation sequencing technologies, we refer the reader to [16].

2.1 The Basics of Sequencing

DNA sequencing is a process used to determine the order of nucleotides (adenine, A; guanine, G; cytosine, C; and thymine, T) in a strand of a DNA molecule. Knowing exact DNA sequences is necessary for research in molecular biology, and such knowledge has resulted in numerous breakthrough applications in medicine, forensics, and other fields. The process of DNA sequencing of the full genome of an organism is referred to as complete or *whole-genome* sequencing. Whole-genome sequencing includes the sequencing of chromosomal, mitochondrial, and (in plants) chloroplast DNA.

Owing to the limited power of even the most modern technologies, a whole genome or a long DNA strand cannot be sequenced affordably as a whole piece in a reasonable time and with the desired quality. Therefore, most of the existing techniques shatter the molecule into millions of smaller pieces (anywhere between 20 and 2,000 bp), amplify each of them by the polymerase chain reaction (PCR), and then run them through sequencers. The resulting “small” DNA sequences are called *reads*, and are then assembled into longer segments and, eventually, genomes.

This process is known as *genome assembly* and is performed by the use of various algorithms known as *assembly algorithms* (with their respective implementations referred to as *assemblers*).

The sequencing can be *de novo*, where new sequences are assembled without a reference sequence, or *mapping-based*, where the method relies on a reference sequence. There are two types of assembly strategies, depending on the type of sequencing. In the *de novo* assembly approach, sequence reads are compared with each other, and then assembled into longer segments called *contigs* by using the overlaps of the sequences. The reference-based assembly approach involves mapping each read to a reference genome sequence [27].

To ensure that all nucleotides from the sequenced DNA are read, the sequencing process must have a certain *depth*. The average number of reads that contain a given nucleotide is called the *sequencing coverage or the depth*. It is denoted by C , and can be calculated from $C = NL/G$, where N is the total number of reads, L is the average read length, and G is the size of the whole genome. A higher-depth sequencing provides greater accuracy of the assembly [25].

2.2 The History of Sequencing Technologies

The ultimate goal of the genome sequencing and assembly process is to correctly determine the complete genome sequence of an organism and to characterize and annotate the protein-coding genes.

An understanding of organismal genomes is expected to revolutionize molecular medicine, pharmaceuticals, and environmental studies [11]. The first genome to be completely sequenced was bacterial, and this was done in 1995 [8]. The human genome was sequenced *de novo* in the Human Genome Project, which started in 1990, and was completed in 2003. The total cost of the project has been estimated at \$3 billion [38]. To illustrate the advancement of genome assembly techniques since then, we may mention that at the beginning of 2012, Life Technologies announced a sequencer designed to sequence an individual human genome in 1 day for a cost of \$1,000 [9]. Despite the fact that the latter example is not *de novo* sequencing, it shows a significant improvement in sequencing technologies, leading to a decrease in the time invested and the cost.

Historically, sequencing technologies can be divided into several categories. Before 1980, the sequencing procedure was performed manually [1]. Then, the Sanger sequencing method was adopted in laboratories around the world and was the prevalent method used for two decades. The capillary sequencer machine incorporated Sanger's sequencing method, for which Sanger won a Nobel Prize in 1980 [32]. This method was the main method used in the Human Genome Project.

The "Next-generation sequencing" (NGS) technologies are widely used today. Some of the sequencing methods commonly used are based on the Roche/454, Illumina, and SOLiD platforms [24]. More details of these platforms can be found in recent reviews [19, 31].

The Illumina reads are 50–150bp long with up to six billion reads per run, where as 454 can manage 400bp but with lower throughput. Therefore, these technologies are fast and cheap, and have relatively high coverage depth. Current second-generation sequencing technologies produce read lengths ranging from 35 to 400bp. The shortcomings of the NGS technologies are short reads and high data volume, which often lead to difficulties in assembly. Correct assembly and mapping are computationally challenging, as short segments create ambiguities in alignment and in genome assembly, which, in turn, can produce errors when the results are interpreted.

Recently, new technologies such as “single-molecule real-time technology” (SMRT) and the nanopore sequencing method [19] have been proposed. The general characteristics of these methods are a shorter DNA preparation time before sequencing, capturing the nucleotide signal in real time, and longer reads. The error rates of single-molecule reads are high (less than 85 % nucleotide accuracy). To overcome this problem, a new hybrid method that combines these technologies with NGS and yields very good accuracy for long reads has been developed [14].

3 Three Categories of Assembly Algorithms

Although NGS technologies provide the possibility of fast and cheap sequencing, they result in a computationally challenging assembly problem. There are three basic approaches to designing assembly algorithms: the greedy-algorithm approach; overlap–layout–consensus methods, relying on overlap graphs; and de Bruijn methods, based on de Bruijn graphs.

3.1 Greedy Assembly

Two reads are considered to overlap if a prefix of one nucleotide sequence is the same as or very similar to a suffix of the other. The quality of an overlap is quantified by the size of the overlapping sequence and the percentage of matching base pairs in the overlapping region. All of the greedy assemblers use a variant of the greedy-algorithm approach. Some of greedy algorithms start by iteratively joining the reads with the best overlaps, forming multiple contigs. Others extend a given read to a contig by consecutively attaching the read that has the best overlap with the previous one. This is performed at both the 3' and the 5' end of the read until no further extensions are possible. An unassembled read is then chosen to be the start of a new contig. In this selection process, priority is given to reads of better quality. The greedy approach was first used on Sanger data sets by assemblers such as TIGR [35] and CAP3 [10]; a second approach was more recently applied to short-read data assemblers (e.g., SSAKE [37], VCAKE [13], and SHARCGS [5]).

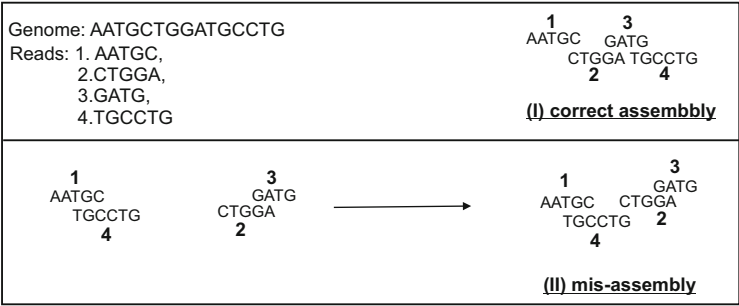


Fig. 1 Genome misassembly by greedy algorithm. (I) A small genome is to be assembled from four reads, labeled 1, 2, 3, and 4. The correct assembly is 1–2–3–4. (II) The greedy algorithm assembles 1 and 4 first, since they have the best overlap, of three nucleotides, and then 2 and 3 are assembled. As a result, the misassembled genome 1–4–2–3 is obtained

These assembly algorithms, like every other greedy algorithm, aim at determining the optimal global assembly by finding locally optimal assemblies at each step. This is a simple assembly strategy and very easy to implement, but it does not necessarily result in the optimal solution. In addition, a greedy assembly might also lead a misassembly, as illustrated in Fig. 1.

3.2 *Overlap–Layout–Consensus Assembly*

The *overlap–layout–consensus (OLC)* assembly approach is graph-based. A *graph* is a structure composed of a set of *vertices* connected by *edges*. If the edges are directed, the graph is called *directed*. If a vertex u is an endpoint of an edge e , we say that e is an incident edge on u . A *path* in a graph is an alternating sequence of vertices and edges $u_1, e_1, u_2, e_2, \dots, e_{k-1}, u_k$ such that each edge e_i in the sequence is incident on both u_i and u_{i+1} . A *Hamiltonian path* is a path that visits every vertex of the graph exactly once. Examples of a graph, a path, and a Hamiltonian path are given in Fig. 2 (I), (II), and (III), respectively.

The assembly of a genome by the OLC approach method can be seen as a mathematical problem of finding a Hamiltonian path in a directed graph. The graph structure used by OLC assemblers is called an overlap graph. The vertices of the overlap graph represent the reads, so that the graph has as many vertices as there are reads. The first (overlap) step in an OLC assembly is the identification of overlapping reads by pairwise comparison. The second (layout) step is the construction of the graph, such that two vertices are connected with an edge if the corresponding reads overlap. The direction of the edge is from the vertex that contains the overlap as a suffix towards the vertex that contains the overlap as a prefix. Each path in such a graph constructed in this way corresponds to an assembled contig, and each Hamiltonian path corresponds to a completely

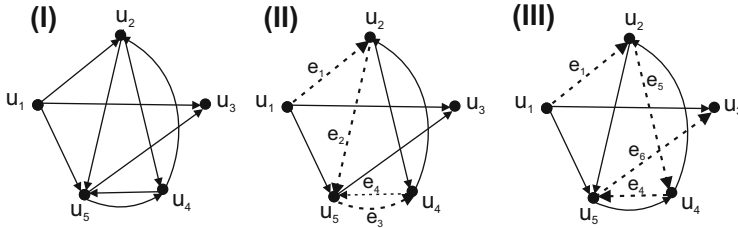


Fig. 2 (I) A directed graph with five vertices, labeled u_1, u_2, u_3, u_4, u_5 . (II) The alternating sequence of vertices and edges $u_1e_1u_2e_2u_5e_3u_4e_4u_5$ is a path. (III) The alternating sequence of vertices and edges $u_1e_1u_2e_5u_4e_4u_5e_6u_3$ is a Hamiltonian path

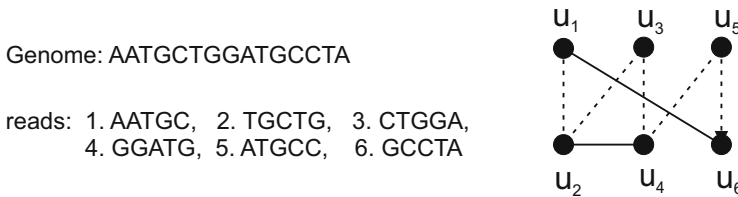


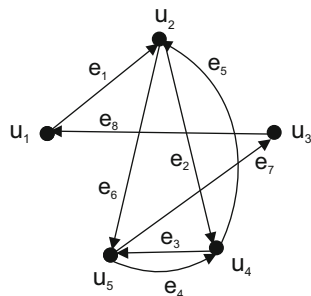
Fig. 3 Overlap graph. The graph corresponds to the overlap graph of the sequence used in Example 1

assembled genome. Finding a Hamiltonian path is then the third step in the process. A simplified example to illustrate the OLC assembly process is given in Fig. 3.

Example 1. This example refers to the overlap graph shown in Fig. 3. A small genome $\Gamma = \text{AATGCTGGATGCCTA}$ is to be assembled from reads 1, 2, 3, 4, 5, and 6. Each vertex of the graph corresponds to a read. Two vertices are connected by a directed edge if the corresponding reads have an overlap of at least two nucleotides. There is a Hamiltonian path 1–2–3–4–5–6 in the overlap graph. By listing the reads corresponding to each vertex on this path (including only a single copy of each overlap) in consecutive order as they appear in the path, one obtains the assembled genome.

Two well-known assemblers that use the OLC method are Newbler [22] and Celera Assembler [26]. The OLC assembly method is exact, and therefore more accurate than the greedy approach. However, finding Hamiltonian paths in a graph is, in general, an NP-hard problem, for which there exists no efficient algorithm. It should also be pointed out that the resulting graphs include millions of vertices, so even efficient heuristics need to scale linearly with the order and size of the graph. Therefore, this approach might be convenient for smaller number of reads (i.e., larger read lengths). As the reads become shorter, as in the case of NGS data, the use of OLC assembly techniques becomes prohibitive. Some of the difficulties can be resolved by considering Eulerian instead of Hamiltonian paths and creating de Bruijn instead of overlap graphs, as discussed in the following section.

Fig. 4 Eulerian path. The sequence $u_1 e_1 u_2 e_2 u_4 e_3 u_5 e_4 u_4 e_5 u_2 e_6 u_5 e_7 u_3 e_8 u_1$ is an Eulerian path in the graph



3.3 De Bruijn Assembly

De Bruijn graphs were introduced in the 1940s by the Dutch mathematician Nicholas Govert de Bruijn long before their application in genome assembly. De Bruijn was interested in the following “string reconstruction problem”: Find the shortest superstring that contains as substrings all possible strings of a given length k over an arbitrary alphabet. He solved the problem by encoding it in directed graphs, later known as *de Bruijn* graphs.

For every sequence of length $k - 1$, there is a vertex in the *de Bruijn* graph. Two vertices u_1 and u_2 are connected by an edge directed from u_1 to u_2 if there is a k -mer that has the $(k - 1)$ -mer u_1 as its prefix and the $(k - 1)$ -mer u_2 as its suffix. The k -mer composed of u_1 and u_2 is the label of the edge connecting them. Therefore, traversing each edge in the graph yields a path whose edge labels give the smallest sequence that has all possible k -mers as subsequences.

A path that contains every edge of the graph exactly once is called an *Eulerian path*. Hence, finding an Eulerian path in the *de Bruijn* graph solves the sequence reconstruction problem. An example of an Eulerian path is depicted in Fig. 4.

De Bruijn graphs can readily be used to solve the genome assembly problem (see [12, 28]). If we consider an alphabet $\{A, C, G, T\}$ and, instead of using all possible k -mers as edges, we use only those generated from the reads, we can construct a *de Bruijn* graph as described above. An Eulerian path in such a graph corresponds to an assembled DNA sequence.

This approach has been used in many assemblers, such as Velvet [39], ABySS [34], and AllPaths [2]. For more details of how the *de Bruijn* graph was used in these “DBG” assemblers, see [25]. A simple example of DBG genome assembly is illustrated in Fig. 5 and described as follows.

Example 2. Consider the genome $\Gamma = \text{AATGCTGGATGCCTA}$. The set of reads is

$$\{\text{AATGC}, \text{TGCTG}, \text{CTGGA}, \text{GGATG}, \text{ATGCC}, \text{GCCTA}\}.$$

We construct a graph with a vertex set composed of every 3-mer obtained from the reads

Genome: AATGCTGGATGCCTA

Reads: 1. AATGC, 2. TGCTG, 3. CTGGA,
4. GGATG, 5. ATGCC, 6. GCCTA

Set of 3-mers: $u_1 = \text{AAT}$, $u_2 = \text{ATG}$, $u_3 = \text{TGC}$,
 $u_4 = \text{GCT}$, $u_5 = \text{CTG}$, $u_6 = \text{TGG}$,
 $u_7 = \text{GGA}$, $u_8 = \text{GAT}$, $u_9 = \text{GCC}$,
 $u_{10} = \text{CCT}$, $u_{11} = \text{CTA}$

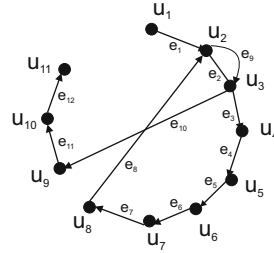


Fig. 5 De Bruijn graph. The graph on the *right* is the de Bruijn graph that is used to assemble the genome AATGCTGGATGCCTA

$$\{u_1 = \text{AAT}, u_2 = \text{ATG}, u_3 = \text{TGC}, u_4 = \text{GCT}, u_5 = \text{CTG}, u_6 = \text{TGG}, \\ u_7 = \text{GGA}, u_8 = \text{GAT}, u_9 = \text{GCC}, u_{10} = \text{CCT}, u_{11} = \text{CTA}\}.$$

A directed edge from vertex v to vertex w is included if v is a prefix and w is a suffix of a 4-mer that belongs to a read. The de Bruijn graph depicted in Fig. 5 is obtained. The path $u_1 e_1 u_2 e_2 u_3 e_3 u_4 e_4 u_5 e_5 u_6 e_6 u_7 e_7 u_8 e_8 u_9 e_9 u_{10} e_{10} u_{11} e_{11} u_1$ is an Eulerian path which corresponds to the assembled genome Γ .

Finding an Eulerian path in a graph is a computationally easy problem, rendering the de Bruijn approach more applicable than the Hamiltonian-path approach used in the OLC method. On the other hand, one of the drawbacks of the de Bruijn approach is the loss of information caused by decomposing a read into a path of k -mers [33].

3.4 Summary of Advantages and Disadvantages of the Popular Assembly Methods

The quality of DNA sequencing and assembly can be quantified by a few characteristics, including the speed, accuracy, and cost of the sequencing, the computational complexity of the assembly, and the accuracy of the assembly. All of these categories are interlaced with each other, and also all of them depend on the sequencing method, the biotechnology used, the computational approach, and the software. Unfortunately, there is no solve-it-all assembly or sequencing technique for the genome assembly problem. All of the methods described above might perform very well in terms of one characteristic, but not another.

The Sanger sequencing method produces reads ranging from 800 to 900 bp in length, which are much longer than the NGS read lengths (50–150 bp). Therefore, the assembly process is less complex, and an approach such as the OLC method might yield reasonable computational complexity. But the cost of generating Sanger data is much higher, and this technology is more time-consuming. Next-generation

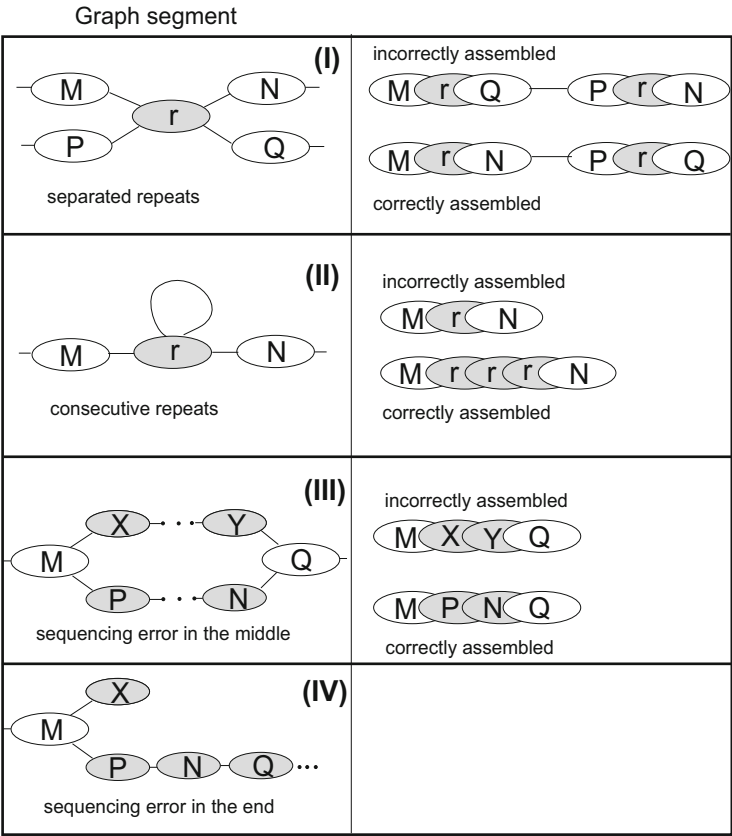


Fig. 6 Errors caused by repetition. On the *left-hand side*, a portion of an assembly graph that contains a repetitive segment is shown. On the *right-hand side*, the correct assembly is depicted, along with a possible incorrect assembly

sequencers can read base pairs at a thousandth of the cost of Sanger sequencers and faster, but the size of the reads makes the computation very laborious and error-prone. Therefore, there is no single assembler that will perform well on any type of data set. For example, although the greedy approach is computationally feasible and might work well on small genomes, its output may be far from the optimal. On the other hand, the robustness of the OLC method comes at the price of its being more computationally intensive, especially when the number of reads is large, which is the case for the NGS technologies. The de Bruijn approach can easily be implemented, but suffers from loss of information when the reads are chopped up into k -mers [30].

All existing assemblers and assembly approaches face the same problems, which are largely due to sequencing errors and repeats in the genome. Sequencing errors in the middle of the set of reads might produce “bubbles” (see Fig.6 (III)) in the graph structure used for assembly. This implies that there is no Hamiltonian path

that traverses all vertices in the region and no Eulerian path that includes the edges in both branches. On the other hand, if there is a sequencing error (or drop in coverage) at the end of the reads (see Fig. 6 (IV)), the graph has a “spur” – a short dead end that cannot belong to any path, but complicates the graph structure and causes ambiguities [25].

Repeated segments might cause two types of misassembly. The first type occurs when there are multiple consecutive copies of the same segment, as illustrated in Fig. 6 (II). Any assembly algorithm will have a problem in detecting the correct number of repeated copies, and, very often, fewer or more copies are included. The second type of misassembly (see Fig. 6 (I)) occurs when the repeated segments are apart from each other. There is then a possibility that the assembler will create a chimera by falsely joining two regions, and the resulting genome will be rearranged in comparison with the reference. Assembly errors that appear because of repeats are a very serious problem because sometimes the number of repeated copies is closely related to some phenotypic characteristic of an organism, and a shuffle of regions might be mistaken for a DNA rearrangement event [29].

Different strategies have been developed for different assemblers to resolve the problems caused by repeats. One of the most commonly used solutions is the use of mate pairs or paired ends. Mate pairs and paired ends are reads (of size 150–500 bp) generated in pairs from opposite ends of a longer DNA sequence. There is no significant difference between paired ends and mate pairs in terms of assembly, even though the laboratory techniques used to generate them are very different (see [23]). The pairs span larger regions, ranging from 200 to 20,000 bp. If a pair contains a repeat, it might provide enough information about the correct context of the repeated segments. For instance, in the situation depicted in Fig. 6 (I), the assembler would be able to identify the correct assembly if a mate pair which spans $M-N$ (or the first repeat r) and a mate pair that spans $P-Q$ (or the second repeat r) were available.

Another widely used approach for resolving repeats is a comparison of the depth of coverage for each contig (branch in the graph) that is in question. This method is based on the assumption that the set of reads is uniformly distributed throughout the genome (which is not generally true) and helps in estimating the number of repeated copies. This approach might be particularly helpful when one is “popping” bubbles or counting the number of repeats, as illustrated in Fig. 6. For instance, in Fig. 6 (III), if there were proportionally more reads (edges) matching the segment XY than matching the segment PN , the assembler would form a contig $MX YQ$. Also, in Fig. 6 (III), if the number of reads covering the repeated segment r was m -fold in comparison with the average depth of coverage, then m copies of r would be included in the contig [36].

Despite the attempts to resolve these problems, none of the current assemblers are applicable to all kinds of data sets while simultaneously exhibiting high accuracy and low computational complexity.

4 Theoretical Results

The results presented in this section are mainly combinatorial results on words over a given alphabet. Combinatorics on words is an area of discrete mathematics that initially dealt with problems in theoretical computer science, including formal languages, automata theory, coding theory, and the theory of computation. In mathematics, there are a number of problems that deal with reconstruction, for instance reconstruction of a function from its values at some points, of a group from its subgroups, of a graph from a subset of its subgraphs, and of an image from a sample point set, to name just a few. Combinatorics on words is an area that, among other problems, deals with the sequence reconstruction problem, and addresses the following aspects:

- Uniqueness of sequences reconstructed from the same set of subsequences;
- The existence of a reconstruction based on the size and the number of available subsequences; and
- Unambiguous reconstruction with respect to the structure and origin of the subsequences.

Some of these theoretical results can be applied directly to sequence assembly, and answer different questions related to DNA sequencing. Therefore, we have included a few of the results that we believe are the most relevant to assembly problems.

4.1 Definitions and Notation

Let Σ be a finite alphabet of symbols. The elements of Σ are called letters. The set of all sequences over Σ is denoted by Σ^* , and the elements of Σ^* are called *words* (or *sequences*). The *empty word* is denoted by λ . If w is a word over Σ , then we write $|w|$ to denote the length (measured as the number of letters) of w . We denote by Σ_n^* the set of all sequences over Σ of length n .

We often refer to the n th letter in w as the n th position. A sequence v over Σ is called a (scattered) *subsequence* of the word $w = w_1w_2 \cdots w_n$ if $v = v_{i_1}v_{i_2} \cdots v_{i_s}$, for some $1 \leq i_1 < i_2 < \cdots < i_s \leq n$. In other words, we say that w is a *supersequence* of v . The set of all subsequences of length t of a sequence w is called the t -spectrum of w , and we denote it by $S_t(w)$. A sequence v over Σ is called a *factor* of the word w if there are words x and y over Σ such that $w = xvy$. Note that x and/or y may be empty. These factors are special types of subsequences. Given a word w such that $w = xy$, we define $w - x = y$. A factor v is said to be a *prefix* of the word $w = xvy$ if $x = \lambda$ and, similarly, v is a *suffix* of w if $y = \lambda$. The *overlap* of two words w and w' , denoted by $o(w, w')$, is defined as the maximal factor v which is a suffix of w and a prefix of w' . It may also be that $v = \lambda$.

For two words w and w' , we define an operation “ \circ ”, called *composition*, by which $w \circ w' = w(w' - o(w, w'))$. In other words, we concatenate the two words, though a single copy of their largest overlap.

Example 3. Let $\Sigma = \{a, b, c\}$ and $w = aaaacccbabacc$, $w' = baccbba$. The word $v = bacc$ is the maximal factor – a suffix of w and prefix of w' . Therefore, $v = o(w, w')$. The composition of w and w' is $w \circ w' = aaaacccbabaccbba$.

4.2 Sequence Reconstruction from Subsequences

The Russian mathematician V. Levenshtein investigated the problem of reconstruction of a sequence from its subsequences and supersequences. In [18], Levenshtein determined the number of subsequences needed to reconstruct an unknown sequence. The result is stated below in Theorem 1. Let X be an unknown sequence of length n . For a number $t < n$ we wish to determine how many different subsequences of length t are needed to uniquely reconstruct X . This answers the following question: What is the minimum number of reads of length t that one needs to reconstruct a genome? We note here that considering subsequences instead of factors might seem far from reality in terms of genome assembly, but in fact the mate pairs are special types of subsequences and, furthermore, each read can be viewed as a subsequence when we take sequencing errors into account.

All of the sequences in Σ_n^* are considered first, and they are compared pairwise to count the subsequences of length t that they have in common, i.e., we find the cardinality of the sets $S_t(w) \cap S_t(v)$ for every $w, v \in \Sigma_n^*$. Let $N(n, t)$ denote the maximum size of the set of subsequences of length t shared between two sequences of length n , i.e.,

$$N(n, t) = \max\{|S_t(w) \cap S_t(v)|, w, v \in \Sigma_n^*\}.$$

Theorem 1. *The minimum number of subsequences of length t that are needed to reconstruct an unknown sequence X of size $|X| = n$ equals $N(n, t) + 1$.*

Levenshtein also provided an efficient algorithm in [18] for performing the reconstruction. The number $S_t(w)$ depends on the sequence w (see [17]) and, therefore, there is no exact formula for calculating $N(n, t)$. This number can be calculated recursively as follows:

$$N(n, t) = S(n, t) - S(n - 1, t) + S(n - 2, t - 1),$$

where $S(n, t) = \max\{|S_t(w)|, w \in \Sigma_n^*\}$.

The problem with applying these results to genome assembly is the fact that there are sequences for which the existence of $N(n, t) + 1$ different subsequences is not guaranteed. For more details, we refer the reader to Example 4 below. In some

cases, more than half of the sequences of a given length do not have enough different subsequences from which they can be reconstructed.

Example 4. Let $\Sigma = \{A, G, T, C\}$ and let $X \in \Sigma^*$ be such that $|X| = 5$. Let $t = 3$. According to Theorem 1, one needs $N(5, 3) + 1$ different subsequences of length 3 to reconstruct X . By examining the subsequences of length 3, one can easily conclude that $|S_3(w) \cap S_3(v)| = 7$, where $w = AGTCG$ and $v = ATGCG$. Therefore, $N(5, 3) + 1 \geq 8$, and one needs eight or more different subsequences of length 3 to uniquely construct X . Note that no sequence composed of fewer than three different symbols from Σ has eight different subsequences. This implies that those sequences composed of a single symbol (in total, four such sequences) and those composed of two different symbols (in total, $2^5 * C(4, 2) = 192$ sequences) do not have eight different subsequences. Therefore, at least 196 sequences of length 5, out of 1,024 in total, cannot be uniquely reconstructed. In other words, there are high odds, greater than $\frac{1}{6}$, that a given sequence X cannot be reconstructed.

Besides the structure of X , the reconstruction depends on the size of the available subsequences (reads). By changing the number t , one might be able to find enough subsequences to reconstruct X , as discussed in the next subsection.

4.3 Reconstruction Based on the Size of the Subsequences

There are two types of questions that can be asked about reconstruction with respect to the size of the given subsequences:

- What is the smallest k such that one can reconstruct any word of length n from the multiset of its subsequences of length k ?
- What is the smallest k such that one can reconstruct a word from the set of its different subsequences of length k ?

Note that a *multiset* is a set of elements such that multiple occurrences of an element are allowed and their number is known. Therefore, the first problem is about reconstruction of a sequence given all of the repeats and the number of copies of each repeat, which is a far from realistic data requirement. The second problem is more relevant to genome assembly, since it does not require input information about the repeated reads.

Manvel et al. [21] showed that the reconstruction of a sequence of length n is unique if all subsequences of size k are given, where $k \geq n/2$. In addition, it was proven that a unique reconstruction is not possible for $k < \log_2 n$. This implies that two words of length n that have identical sets of subsequences up to $n/2$ might not be identical. The lower bound on k has been improved several times. In [15], the bound $k \geq 5 + \frac{16}{7}\sqrt{n}$ was given, and Dudik et al. [6] showed that

$$k \geq 3^{(\sqrt{2/3} - o(1)) \log_3^{1/2} n}.$$

These results thus provide estimates of k as an answer to the first problem above.

In any case, the condition $k \geq n/2$, which implies that to uniquely reconstruct one particular small genome namely that of a virus found in *Escherichia coli* with $n = 5,386$, the read sizes must be greater than 2,193 bp. This is when the repeats and their multiplicities are known and also all subsequences are available, conditions that are far from realistic.

The result most relevant to genome assembly is one presented in [7]. The authors of that paper examined finite words over an alphabet $\Gamma = \{a, \bar{a}, b, \bar{b}\}$ of pairs of letters, where each word $w_1 w_2 \dots w_t$ is identified with its reverse complement $\overline{w_t w_{t-1} \dots w_1}$. This approach takes account of the reads and their reverse complements, as is actually done in the genome assembly process. It was shown that the smallest k for which every word of length n over Γ is uniquely determined by the set of its subwords of length up to k is given by $k \approx 2n/3$ [7]. To put this result into perspective in relation to genome assembly, let us consider one of the smallest genomes, of size $n = 5,386$. Then $k \approx 2 * 5,386/3 = 3,590.6$, which implies that one needs all possible reads of length 3,590 to uniquely assemble a genome of size 5,386. This number becomes significantly larger for larger genomes. For instance, the size of the human genome is estimated at 3.2 billion bp, and thus one needs all of the $k \approx 2.13$ billion bp long subsequences to uniquely reconstruct the genome.

The following result shows that no sequence can be uniquely determined by a k -spectrum composed of factors only. Namely, the maximum length n such that every word of this length can be uniquely determined by its factors of size k , for some $k \leq n$, is k . This implies that there are words of length greater than k which cannot be uniquely reconstructed from their set of factors of length k (see [20]).

Example 5. Let $w = ababab \dots ab$ and $v = bababa \dots ba$ be two sequences of size n . We consider the sets of factors of length k , $F_k(w)$ for w and $F_k(v)$ for v , where $k = n - 1$. We have $F_k(w) = F_k(v) = \{abab \dots a, baba \dots b\}$, and therefore w cannot be uniquely reconstructed from $F_k(w)$.

4.4 Reconstruction Based on the Structure and Origin of the Factors

In this section, we consider some results due to Carpi and De Luca [3] and Carpi et al. [4]. Unlike Levenshtein, whose results do not guarantee reconstruction of every possible sequence, these authors analyzed the reconstruction of sequences from sets of factors which permit reconstruction of the entire word. This analysis is based on the notion of boxes. Before we state the main result, we need some preliminary definitions.

Definition 1. The *initial box* of a word w is the shortest unrepeated prefix of w . The *terminal box* of a word w is the shortest unrepeated suffix of w .

Definition 2. *Superbox* is a factor of w that can be written as asb , where $a, b \in \Sigma$, s is repeated factor and as, sb are un-repeated factors of w .

The main result can be summarized in the following theorem.

Theorem 2. *Any finite word w is uniquely determined by the initial box, the terminal box, and the set of superboxes.*

Example 6. Consider the small genome $ATCCTATCAT$. The initial box is $ATCC$, the terminal box is CAT , and the set of superboxes is $CCT, CTA, TATCA$.

An efficient algorithm for finding the initial box, terminal box, and superboxes is given in [4]. In relation to the genome assembly problem, the theorem implies that one can uniquely assemble a genome if special reads are known (they all might be of different lengths). Those special reads are the maximal unrepeated subsequences of the genome, i.e., the unrepeated sequences such that each of their proper subsequences is repeated in the genome.

Long interspersed nuclear elements (LINEs) are repeated in the human genome and can be as long as a million base pairs. Based on Theorem 2, one would require a read of length approximately one million base pairs to uniquely assemble the human genome.

5 An Optimization-Based Approach

In this section, we present some preliminary results obtained from our optimization-based approach to solving the sequence reconstruction problem, by providing a set of mathematical formulations as programming problems which deal with different aspects of the assembly problem.

Let \mathfrak{S} be a sequence over Σ such that $|\mathfrak{S}| = n$. Let W be a collection of (all) factors of \mathfrak{S} with length k . Our goal is to reconstruct \mathfrak{S} by appropriately ordering the elements of W . To this end, we determine the optimal solution with the minimum number of mismatches by encoding the problem into a quadratic integer programming problem.

Let $W = \{w_1, w_2, \dots, w_N\}$ be a set of factors of \mathfrak{S} and let $|w_i| = l$, where $l \geq 2$, for every $i \in \{1, 2, \dots, N\}$. If there is a subset $W' = \{w'_1, \dots, w'_{N'}\} \subset W$ and a permutation $P = (p_1, p_2, \dots, p_{N'})$ such that $\mathfrak{S} = w'_{p_1} \circ w'_{p_2} \circ \dots \circ w'_{p_{N'}}$, then we say that \mathfrak{S} is covered by W , and the pair (W', P) is called a perfect coverage of \mathfrak{S} . There may be no perfect coverage or there may exist multiple perfect coverage pairs for a given \mathfrak{S} and W . Reconstructing \mathfrak{S} from a set of factors $W = \{w_1, w_2, \dots, w_N\}$ means finding a perfect coverage (W', P) , if it exists. Let (W', P) be a pair made up of $W' \subset W$ and a permutation $P = (p_1, p_2, \dots, p_{N'})$ of the elements of W' , such that $|\mathfrak{S}| = |w'_{p_1} \circ w'_{p_2} \circ \dots \circ w'_{p_{N'}}|$. We call (W', P) a coverage of \mathfrak{S} . If the letters at the i th positions of \mathfrak{S} and $w'_{p_1} \circ w'_{p_2} \circ \dots \circ w'_{p_{N'}}$ differ, then we say that these two words have a *mismatch* at the i th position. In the case where a perfect coverage does

not exist for \mathfrak{S} , it is interesting to consider a coverage with the minimum number of mismatches.

The sequence reconstruction problem described above can be translated directly into the problem of genome assembly from a large number of reads (factors). Namely, one can look at \mathfrak{S} as a genome that has to be reconstructed from a set of reads $W = \{w_1, w_2, \dots, w_N\}$. Owing to mutations and errors in sequencing, the sequence that is reconstructed from W may contain mismatches with respect to the actual genome. Therefore, finding a coverage of \mathfrak{S} with the minimum number of mismatches would optimize the problem of genome assembly.

We define $x_{ijk} \in \{0, 1\}$ for $i \in \{1, 2, \dots, N\}$, $j \in \{1, 2, \dots, l\}$, $k \in \{1, 2, \dots, n\}$ by

$$x_{ik} = \begin{cases} 1 & \text{if the } j\text{th position of } w_i \text{ is assigned to the } k\text{th position in } \mathfrak{S}, \\ 0 & \text{otherwise.} \end{cases}$$

Since $|w_i| = l$ for every i , the first constraint is given by

$$\forall i, \sum_{j=1}^l \sum_{k=1}^n x_{ijk} = l. \quad (1)$$

To satisfy the condition that each position in \mathfrak{S} is covered by at least one word w_i (i.e., there are no gaps in the sequence that we reconstruct), the following constraint is added:

$$\forall k, \sum_{j=1}^l \sum_{i=1}^N x_{ijk} \geq 1. \quad (2)$$

In addition, we have to ensure the overlaps among the words are along the length l of each word:

$$\forall i \forall k, l \leq k \leq n, \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} \leq l. \quad (3)$$

We need three more constraints to obtain the intended result. The first of these guarantees that every symbol of w_i is matched to exactly one position in \mathfrak{S} . The second prevents the inclusion of multiple copies of a word, and the third does not allow stacking of words. These constraints are as follows:

$$\forall i \forall j, \sum_{k=1}^n x_{ijk} = 1, \quad (4)$$

$$\forall i \forall k, \sum_{j=1}^l x_{ijk} \leq 1, \quad (5)$$

$$\forall j \forall k, \sum_{i=1}^N x_{ijk} \leq 1. \quad (6)$$

We introduce two new binary variables z_{ik} and y_{ik} , such that $z_{ik}, y_{ik} \in \{0, 1\}$ such that

$$z_{ik} = 1, y_{ik} = 1 \text{ if } \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} = l, \quad (7)$$

$$z_{ik} = 0, y_{ik} = 1 \text{ if } l < \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} < 2l, \quad (8)$$

$$z_{ik} = 1, y_{ik} = 0 \text{ if } 0 \leq \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} < l. \quad (9)$$

The constraints in Eqs. (10)–(14) below ensure that the new variables are properly defined (as described in Eqs. (7)–(9)):

$$\forall i \forall k, l \leq k \leq n \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} - l \leq (1 - z_{ik})l, \quad (10)$$

$$\forall i, \forall k, l \leq k \leq n \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} - l \geq -z_{ik}l, \quad (11)$$

$$\forall i \forall k, l \leq k \leq n \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} - l \geq (1 - y_{ik})(-l), \quad (12)$$

$$\forall i \forall k, l \leq k \leq n \sum_{j=1}^l \sum_{m=k}^{k+l-1} x_{ijm} - l \leq y_{ik}l, \quad (13)$$

$$\forall i \forall k, 1 \leq z_{ik} + y_{ik} \leq 2. \quad (14)$$

To ensure that a factor w_i fits at only one position, the following constraint is added:

$$\forall i \sum_{k=1}^n (z_{ik} + y_{ik} - 1) = 1. \quad (15)$$

Moreover, owing to the consecutive property, we set

$$\forall i \forall k, 1 \leq k \leq (l-1), \forall j, (k+1) \leq j \leq l, x_{ijk} = 0, \quad (16)$$

$$\forall i \forall k, (n-l+2) \leq k \leq n, \forall j, 1 \leq j \leq ((l-1)-(n-k)), x_{ijk} = 0. \quad (17)$$

The optimal solution for the number of mismatches can then be found by using the following objective function, which is to be minimized:

$$\min \sum_{k=1}^n \sum_{i,j,i',j'} (w_i^j \neq w_{i'}^{j'}) x_{ijk} x_{i'j'k} \leq y_{ik} l \text{ for all } i, i' \neq i'. \quad (18)$$

Note that this objective function is quadratic and that the number of variables that we introduce depends on the number of reads. Furthermore, the constraints imposed are linear in the integer variables, which render this formulation a quadratic integer programming problem.

In addition to the general case of the quadratic programming problem described above, one can consider three different variants of the problem, where (1) we use a subset of the factors (instead of all possible factors), (2) we use a subset of paired factors, and (3) we use a subset of the factors including their inverses. Each of these variants is relevant to the genome assembly problem and can be matched to some of the NGS technologies that are being employed. However, because of the nature of this chapter, we shall not give details of the formulation of the optimization problem for these cases.

Finally, we should point out that solving an integer programming problem is, in general, an NP-hard problem. The computational challenges are reinforced by the large number of variables that would arise in any realistic application. Although there are efficient techniques for relaxing the integer constraints, it remains to be investigated how well this optimization-based formulation and its relaxations can solve the problem of genome assembly, even in the case of small, contrived examples.

6 Conclusion

Efficient and accurate solution of the genome assembly problem offers the possibility of improving not only our understanding of the diversity of nature but also the state of the art in medical research. Owing to its numerous applications, this problem has gained considerable attention in the bioinformatics community.

The principal aim of this chapter was to present a comprehensive overview of overlooked mathematical results which may prove relevant to improving the existing heuristic solutions to the genome assembly problem. This necessitated the inclusion of a high-level description of the existing NGS technologies, so that

researchers in applied mathematics might formulate the appropriate theoretical settings. By illustrating the implications of language-theoretic results in realistic scenarios, we may have given the impression that the current technologies do not guarantee the uniqueness of the resulting genome assembly (regardless of the algorithmic approach used). However, the existing theoretical results pertain to special alphabets, use some special subsequences, or discard information about the existence of particular subsequences (which could be empirically verified). Therefore, we believe that a critical comparative view of these results could propel the development of language-theoretical tools that are closer and thus more relevant to realistic genome assembly scenarios.

To this end, we have also presented a preliminary optimization-based formulation of the genome assembly problem as a quadratic integer programming problem, whose performance, with appropriate relaxation, will be addressed in future work. The merit of the integer programming formulation is that it provides a new and potentially useful formulation of the genome assembly problem. We would like to point out that the optimization-based formulation requires information about the estimated size of the genome (here, N). Interestingly, none of the graph-based assembly methods have this as a requirement, rendering the comparison of the resulting genomes (which are likely to be of different lengths and coverage) a nontrivial task. Currently, however, we fail to see how the optimization-based formulation may be used to address the comparison of assemblies from different assemblers.

References

1. J. Adams, DNA sequencing technologies. *Nat. Educ.* **1**(1) (2008)
2. J. Butler, I. MacCallum, M. Kleber, I.A. Shlyakhter, M.K. Belmonte, E.S. Lander, C. Nusbaum, D.B. Jaffe, ALLPATHS, de novo assembly of whole-genome shotgun microreads. *Genome Res.* **18**, 810–820 (2008)
3. A. Carpi, A. De Luca, Words and special factors. *Theor. Comput. Sci.* **259**(1–2), 145–182 (2001)
4. A. Carpi, A. De Luca, S. Varricchio, Words, univalent factors, and boxes. *Acta Inform.* **38**, 409–436 (2002)
5. J.C. Dohm, C. Lottaz, T. Borodina, H. Himmelbauer, SHARCGS, a fast and highly accurate short read assembly algorithm for de novo genomic sequencing. *Genome Res.* **17**, 1697–1706 (2007)
6. M. Dudik, L.J. Schulman, Reconstruction from subsequences. *J. Comb. Theory A* **103**, 337–348 (2003)
7. P.L. Erdos, P. Ligeti, P. Sziklai, D.C. Torney, Subwords in reverse-complement order. *Ann. Comb.* **10**, 415–430 (2006)
8. R.D. Fleischmann, M.D. Adams, O. White, R.A. Clayton, E.F. Kirkness, A.R. Kerlavage, C.J. Bult, J.F. Tomb, B.A. Dougherty, J.M. Merrick, K. McKenney, G. Sutton, W. FitzHugh, C. Fields, J.D. Gocyne, J. Scott, R. Shirley, L. Liu, A. Glodek, J.M. Kelley, J.F. Weidman, C.A. Phillips, T. Spriggs, E. Hedblom, M.D. Cotton, T.R. Utterback, M.C. Hanna, D.T. Nguyen, D.M. Saudek, R.C. Brandon, L.D. Fine, J.L. Fritchman, J.L. Fuhrmann, N.S.M. Geoghagen, C.L. Gnehm, L.A. McDonald, K.V. Small, C.M. Fraser,

- H.O. Smith, J.C. Venter, Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science* **269**(5223), 496–512 (1995)
9. <http://www.lifetechnologies.com/content/lifetech/us/en/home/about-us/news-gallery/press-releases/2012/life-technologies-introduces-the-bechtop-io-protocol.html.html>. Accessed Mar 2013
 10. X. Huang, A. Madan, CAP3: a DNA sequence assembly program. *Genome Res.* **9**, 868–877 (1999)
 11. Human Genome Project Information, Genomic science program. <http://www.genomics.energy.gov>. Accessed Oct 2012
 12. R.M. Idury, M.S. Waterman, A new algorithm for DNA sequence assembly. *J. Comput. Biol.* **2**(2), 291–306 (1995)
 13. W.R. Jeck, J.A. Reinhardt, D.A. Baltrus, M.T. Hickenbotham, V. Magrini, E.R. Mardis, J.L. Dangel, C.D. Jones, Extending assembly of short DNA sequences to handle error. *Bioinformatics* **23**, 2942–2944 (2007)
 14. S. Koren, M.C. Schatz, B.P. Walenz, J. Martin, J.T. Howard, G. Ganapathy, Z. Wang, D.A. Rasko, W.R. McCombie, E.D. Jarvis, A.M. Phillippy, Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nat. Biotechnol.* **30**, 693–700 (2012)
 15. I. Krasikov, Y. Roditty, On a reconstruction problem of sequences. *J. Comb. Theory* **A77**, 344–348 (1997)
 16. H. Lee, H. Tang, Next-generation sequencing technologies and fragment assembly algorithms. *Methods Mol. Biol.* **855**(2), 155–174 (2012)
 17. V. Levenshtein, Reconstruction of objects from a minimum number of distorted patterns. *Dokl. Math.* **55**, 417–420 (1997)
 18. V. Levenshtein, Efficient reconstruction of sequences from their subsequences or supersequences. *J. Comb. Theory A* **93**, 310–332 (2001)
 19. L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, M. Law, Comparison of next-generation sequencing systems. *J. Biomed. Biotechnol.* **2012**, 1–11 (2012)
 20. J. Manuch, Characterization of a word by its subwords, in *Developments in Language Theory – Foundations, Applications, and Perspectives, Proc. DLT 2000*, ed. by G. Rozenberg, W. Thomas, pp. 210–219
 21. B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith, P. Stockmeyer, Reconstruction of sequences. *Discret. Math.* **94**, 209–219 (1991)
 22. M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, L.A. Bemben, J. Berka, M.S. Braverman, Y. Chen, Z. Chen, S.B. Dewell, A. de Winter, J. Drake, L. Du, J.M. Fierro, R. Forte, X.V. Gomes, B.C. Godwin, W. He, S. Helgesen, C.H. Ho, S.K. Hutchison, G. Irzyk, S.C. Jando, M.L.I. Alenquer, T.P. Jarvie, K.B. Jirage, J. Kim, J.R. Knight, J.R. Lanza, J.H. Leamon, W.L. Lee, S.M. Lefkowitz, M. Lei, J. Li, K.L. Lohman, H. Lu, V.B. Makhijani, K.E. McDade, M.P. McKenna, E.W. Myers, E. Nickerson, J.R. Nobile, R. Plant, B.P. Puc, M. Reifler, M.T. Ronan, G.T. Roth, G.J. Sarkis, J.F. Simons, J.W. Simpson, M. Srinivasan, K.R. Tartaro, A. Tomasz, K.A. Vogt, G.A. Volkmer, S.H. Wang, Y. Wang, M.P. Weiner, D.A. Willoughby, P. Yu, R.F. Begley, J.M. Rothberg, Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–380 (2005)
 23. P. Medvedev, M. Stanciu, M. Brudno, Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods* **6**, S13–S20 (2009)
 24. M. Metzker, Sequencing technologies – the next generation. *Nat. Genet.* **11**, 31–46 (2010)
 25. J.R. Miller, S. Koren, G. Sutton, Assembly algorithms for next-generation sequencing data. *Genomics* **95**(6), 315–327 (2010)
 26. E.W. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, M.J. Flanigan, S.A. Kravitz, C.M. Mobarry, K.H. Reinert, K.A. Remington, E.L. Anson, R.A. Bolanos, H. Chou, C.M. Jordan, A.L. Halpern, S. Lonardi, E.M. Beasley, R.C. Brandon, L. Chen, P.J. Dunn, Z. Lai, Y. Liang, D.R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G.M. Rubin, M.D. Adams, J.C. Venter, A whole genome assembly of *Drosophila*. *Science* **287**, 2196–2204 (2000)
 27. P.C. Ng, E.F. Kirkness, Whole genome sequencing. *Methods Mol. Biol.* **628**, 215–226 (2010)
 28. A.P. Pevzner, T. Haixu, S.M. Waterman, An Eulerian path approach to DNA fragment assembly. *PNAS* **98**(17), 9748–9753 (2001)

29. A.M. Phillippy, M.C. Schatz, M. Pop, Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol.* (2008). doi:10.1186/gb-2008-9-3-r55
30. M. Pop, Genome assembly reborn: recent computational challenges. *Brief Bioinform.* **10**(4), 354–366 (2009)
31. M. Quail, M.E. Smith, P. Coupland, T.D. Otto, S.R. Harris, T.R. Connor, A. Bertoni, H.P. Swerdlow, Y. Gu, A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics* **13**(1), 341 (2012). doi:10.1186/1471-2164-13-341
32. F. Sanger, A.R. Coulson, A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J. Mol. Biol.* **94**, 441–448 (1975)
33. M.C. Schatz, A.L. Delcher, S.L. Salzberg, Assembly of large genomes using second-generation sequencing. *Genome Res.* **20**(9), 1165–1173 (2010)
34. J.T. Simpson, K. Wong, S.D. Jackman, J.E. Schein, S.J. Jones, I. Byrol, ABySS, a parallel assembler for short read sequence data. *Genome Res.* **19**, 1117–1123 (2009)
35. G.G. Sutton, O. White, M.D. Adams, A.R. Kerlavage, TIGR assembler: a new tool for assembling large shotgun sequencing projects. *Genome Sci. Technol.* **1**, 9–19 (1995)
36. T.J. Treangen, S.L. Salzberg, Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat. Rev. Genet.* **13**(2), 36–46 (2012)
37. R.L. Warren, G.G. Sutton, S.J. Jones, R.A. Holt, Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* **23**, 500–501 (2007)
38. K.A. Wetterstrand, DNA sequencing costs: data from the NHGRI large-scale genome sequencing program. <http://www.genome.gov/sequencingcosts>. Accessed Oct 2012
39. D.R. Zerbino, E. Birney, Velvet, algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18**, 821–829 (2008)

Discrete and Topological Models in Molecular Biology

Jonoska, N.; Saito, M. (Eds.)

2014, XIII, 524 p. 211 illus., 154 illus. in color.,

Hardcover

ISBN: 978-3-642-40192-3