Graph algorithms for DNA sequencing – origins, current models and the future

Jacek Blazewicz[a,b,c], Marta Kasprzak[a,b,c], Michal Kierzynka[a,c,d,*], Wojciech Frohmberg[a,c], Aleksandra Swiercz[a,b,c], Pawel Wojciechowski[a,b,c], Piotr Zurkowski[a,c]

[a]*Institute of Computing Science, Poznań University of Technology, Poznań, Poland*
[b]*Institute of Bioorganic Chemistry, Polish Academy of Sciences, Poznań, Poland*
[c]*European Center for Bioinformatics and Genomics, Poznań, Poland*
[d]*Poznań Supercomputing and Networking Center, Poznań, Poland*

Abstract

With the ubiquitous presence of next-generation sequencing in modern biological, genetic, pharmaceutical and medical research, not everyone pays attention to the underlying computational methods. Even fewer researchers know what were the origins of the current models for DNA assembly. We present original graph models used in DNA sequencing by hybridization, discuss their properties and connections between them. We also explain how these graph models evolved to adapt to the characteristics of next-generation sequencing. Moreover, we present a practical comparison of state-of-the-art DNA *de novo* assembly tools representing these transformed models, i.e. overlap and decomposition-based graphs. Even though the competition is tough, some assemblers perform better and certainly large differences may be observed in hardware resources utilization. Finally, we outline the most important trends in the sequencing field, and try to predict their impact on the computational models in the future.

Keywords: DNA *de novo* assembly, graph algorithms, whole genome sequencing

1. Introduction

Development of modern computational biology would not be possible without a wide support of computer science. Over the years biology-related questions and prob-

*Corresponding author
Email address: `michal.kierzynka@cs.put.poznan.pl` (Michal Kierzynka)

lems were increasingly often answered or solved by computer algorithms. On the other hand, the same problems brought a fresh look at some areas of computer science, e.g. graph theory, stimulating new findings. As a result, computational biology may be considered as a prime example of combination of two fields of science that create a synergy effect. One specific example may be DNA sequencing, which is a fundamental problem in molecular biology. For decades both biochemical and computational methods associated with this problem have evolved, allowing a tremendous progress in various aspects of the science. This paper is an attempt to review the history of DNA sequencing models and how these influenced current approaches. A short discussion regarding future trends is outlined as well.

Before we dive into the world of sequencing algorithms, let us introduce some basic notions. First of all, DNA stands for *deoxyribonucleic acid* and it is composed of two twisted strands of *nucleotides* forming a double helix. Such chains encode the genetic information of living matter, including the human beings. There are four types of nucleotides in DNA that are distinguished on the basis of their nitrogenous bases: adenine (A), guanine (G), thymine (T) and cytosine (C). Individual bases from the opposite strands are linked by hydrogen bonds in such a way that adenine is always paired with thymine, and guanine is always paired with cytosine. As a result, given a fragment of DNA from one strand, one can always determine the sequence of the corresponding fragment in the opposite strand. Such property is referred to as *complementarity*. The length of a DNA molecule is usually expressed in base pairs (bp), or in the number of nucleotides for single-stranded chains.

DNA sequencing is the process of determining the sequence of nucleotides in a DNA. The story with DNA sequencing began in 1977 when Frederick Sanger with his colleagues proposed a method based on chain-terminating inhibitors [1]. The method was able to read DNA fragments, called *reads*, up to 700-900 bp in length. No computational algorithms were required, as the method was a pure wet laboratory approach. On the down side, such sequencing was pricey, prone to experimental errors and time consuming. In 1988 Edwin Southern introduced a new approach, called sequencing by hybridization (SBH) [2]. In SBH very short DNA fragments (usually 8–12 nucleotides) were read and in order to reconstruct the original molecules (typically a few hundreds of nucleotides) a computational part was needed. Among the pioneers of algorithmic approaches to SBH we can distinguish Y.P. Lysov with his colleagues [3] and P.A. Pevzner [4], who formulated the problem as finding a Hamiltonian path and an Eulerian path, respectively. Yet, a real breakthrough in sequencing came with the advent of so-called next-generation sequencing (NGS), starting from the beginning of the 21st century [5]. The main difference is that NGS methods can produce large numbers of short DNA reads, typically between 35 and 500 bp, at relatively low cost. As a result, this technique has opened the possibility for an affordable sequencing of whole genomes.

Although sequencing data are much larger now and contain different experimental errors, the graph theory layer of the algorithmic solution is more or less the same. The common goal of the *assembly* algorithms (algorithms for sequencing on a large scale) is to find a path in a labeled digraph (either overlap or decomposition-based graph), a path representing the resulting DNA sequence reconstructed from unordered subsequences given at the input. If the reconstruction is done only with the information coming from a sequencing experiment, we refer to this process as *de novo* assembly. Seemingly similar problem is when the reference genome is known. In this case, however, algorithms do not use graphs in the process of reconstruction and the problem is called *resequencing*. This paper focuses on the computational methods associated with *de novo* assembly, their origins and how they evolved in the context of NGS. Moreover, in order to investigate whether there are any substantial differences between the two graph classes, namely overlap and decomposition-based graphs, we carried out a comparative study based on the best currently available implementations.

The rest of the paper is organized in the following way. Section 2 describes in more detail the SBH method and the first algorithmic approaches based on graph theory. Moreover, it classifies graph types depending on their properties and presents models developed for erroneous data sets. Section 3 focuses on the models currently used for NGS data sets. It also describes a theoretical model for paired-end reads and concludes with a short discussion. Section 4 presents a comparative study of state-of-the-art implementations of *de novo* assemblers, representing the main graph families. Finally, conclusions and future trends are presented in Section 5.

2. Original models for DNA sequencing

2.1. Sequencing by hybridization

The DNA sequencing by hybridization consists of two phases: the *hybridization*, which is a biochemical experiment, and the algorithmic part processing the experimental output. The hybridization is based on the tendency of single complementary DNA strands to form a double-stranded complex, and allows discovering one-strand parts of the target DNA. The experiment is performed with the use of a microarray (a bio-chip) containing a library of all possible short DNA chains of a given length l (usually 8–12 nucleotides), thus of cardinality 4^l, and with many copies of the target DNA chain. During the biochemical reaction elements of the library join the longer DNA, if they are complementary to fragments of the DNA, and this can be observed through a fluorescent tagging. At the output one gets a subset of the library, which is expected to compose the target DNA [2].

From the computational point of view, the hybridization gives a set of strings of an equal length l over the alphabet $\{$A, C, G, T$\}$ (called a *spectrum*) and the goal is to use them to construct the resulting sequence of a known length n (a few hundred nucleotides) on the basis of the relation of overlaps between neighboring strings. In a theoretical case, when we assumed the spectrum is complete and with no error, it would consist of $n-l+1$ elements and in order to reconstruct the original sequence one should find an order of the elements such that neighboring ones would always overlap on $l-1$ positions.

In this section, SBH is considered in its classical variant, where all input strings have equal length. There were also other variants, whose aim was to better handle experimental errors or ambiguity of final solution: interactive (multistage) DNA sequencing [6], isothermic DNA sequencing [7], or the sequencing with the use of universal [8] or degenerate nucleotides [9]. Among them the isothermic sequencing can be distinguished as the one which in its computational phase refers to a special graph, mentioned in Subsection 2.3.

2.2. Lysov and Pevzner models

First approaches to the problem of SBH concentrated on the theoretical error-free version. The first algorithm, which refers to a problem known from graph theory was proposed by Lysov and co-workers in [3]. They constructed a directed graph on the basis of the spectrum, where every element corresponds to a vertex, and two vertices are joined by an arc if their spectrum elements (labels) overlap on $l-1$ nucleotides: the suffix of length $l-1$ of the predecessor covers the prefix of the successor. The solution in such a graph is a Hamiltonian path, and the reconstructed DNA sequence is composed of the overlapping labels of the vertices read in the obtained order (see Example 1).

Next year brought equally elegant solution, but of significantly lower computational complexity. Pevzner proposed a graph model, in which an Eulerian path is looked for [4]. This time every element of the spectrum corresponds to an arc in a digraph. It starts in a vertex labeled by the prefix of length $l-1$ of the element and ends in a vertex labeled by its suffix of the same length. The sequence is reconstructed as before, by overlapping labels of vertices from the path.

Example 1. Let the one-strand DNA fragment to be sequenced be CAGAGTCAGTA, where $n = 11$. If the hybridization experiment involves the complete library of probes of length $l = 4$, of cardinality $4^4 = 256$, in the error-free case at the output one will get the set of all substrings of length 4 of the target DNA, i.e. $\{$AGAG, AGTA, AGTC, CAGA, CAGT, GAGT, GTCA, TCAG$\}$. The computational part orders the words in such a way, that neighboring words overlap on $l-1 = 3$ letters, for example AGAG after CAGA.

The graph from the algorithm of Lysov *et al.* is shown in Fig. 1 (A), and the graph from the Pevzner algorithm in Fig. 1 (B). There are two solutions of this problem, i.e. two Hamiltonian or Eulerian paths, respectively, in each graph. Resulting sequences, composed of labels of successive vertices on the paths are: CAGAGTCAGTA and CAGTCAGAGTA.

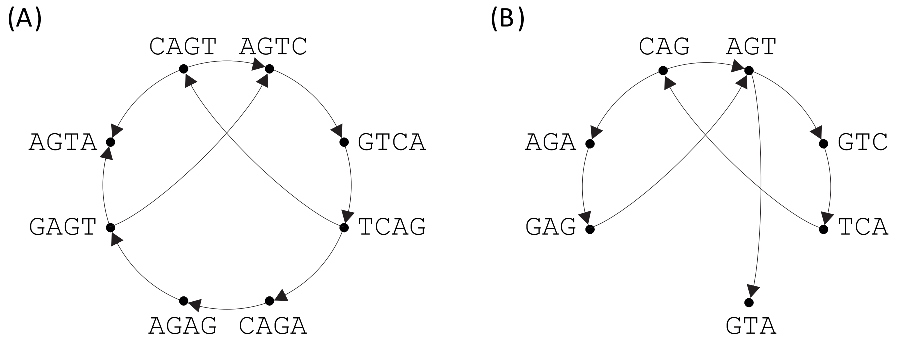


Figure 1: The reconstruction of an original sequence from an error-free spectrum. (A) The Lysov graph [3], where a Hamiltonian path is looked for. (B) The Pevzner graph [4], where an Eulerian path is looked for.

2.3. DNA graphs and others

Both Lysov and Pevzner algorithms are exact, but till 1999 no one explained, what is the relation between graphs proposed there. Why was it possible to replace searching for a Hamiltonian path, which is in general a strongly NP-hard combinatorial problem, by the Eulerian path problem, polynomially solvable?

The question was answered in [10]. The graphs from the method of Lysov *et al.*, so-called the *DNA graphs*, belong to the class of *labeled digraphs*. One of properties of the labeled digraphs is that they are *directed line graphs*, thus the Hamiltonian path problem is polynomially solvable for them. It is done by a transformation of a directed line graph to its *original graph* and by searching for an Eulerian path in the latter graph. A directed line graph G is connected with its original (directed) graph H by the following rules: vertices of G correspond to arcs of H and there is arc (x,y) in G if and only if the terminal endpoint of arc x is the initial endpoint of arc y in H. The existence of an Eulerian path in H is a necessary and sufficient condition of the existence of a Hamiltonian path in G (not valid for undirected graphs). The original graph for a DNA graph is a Pevzner graph constructed for the same spectrum.

De Bruijn graphs, essential also in the context of the subject of the next section, are labeled digraphs, which are constructed with all possible labels of a given length, over a given alphabet. For an alphabet of size a and labels of length k, a de Bruijn graph has a^k vertices, every one labeled by a different word over the alphabet. An arc joins two vertices if the suffix of length $k-1$ of the predecessor covers the prefix of the successor [11].

DNA graphs are vertex-induced subgraphs of de Bruijn graphs with $a = 4$. Pevzner graphs are neither DNA graphs, nor other vertex-induced subgraphs of de Bruijn graphs. They are subgraphs of DNA graphs, thus in consequence, subgraphs of de Bruijn graphs. They are not, in general, labeled/DNA graphs, because they have not 1–1 correspondence between the presence of arcs and the overlaps of vertex labels.

A series of papers on this subject explored further properties of labeled graphs, see for example [12, 13, 14, 15, 16]. On the other hand, there were also works aiming at discovering relations of several digraph classes — directed line graphs and related — having a polynomial-time solution of the Hamiltonian cycle/path problem [17, 18]. A reader may be especially interested in the systematization presented in Fig. 2.

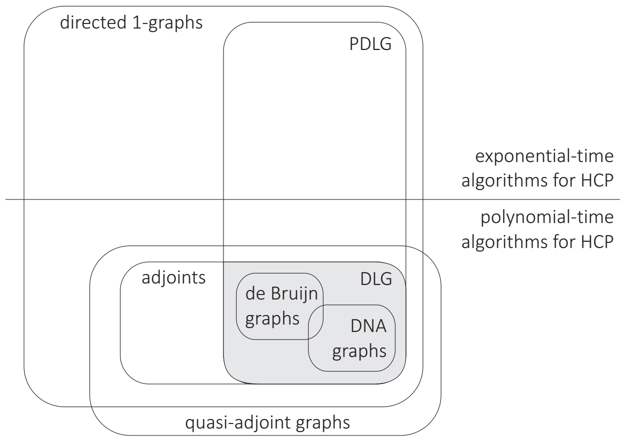


Figure 2: The relationship between a few digraph classes with reference to HCP solvability [18]. DLG stands for directed line graphs (the grey area), PDLG for partial directed line graphs, and HCP for the Hamiltonian cycle/path problem.

The relationship includes the above-mentioned graphs, but also classes whose description goes beyond the matter of this paper. Directed line graphs (DLG) lie at the intersection of adjoints [19] and partial directed line graphs (PDLG, see [20]). Quasi-adjoint graphs (see [17]) are, in addition to the scheme, a superclass of graphs modeling the problem of isothermic DNA sequencing by hybridization (the graphs defined in [21], the problem introduced in [7]). The latter graphs can be either directed line graphs, adjoints not being directed line graphs, or outside adjoints. However, they always are directed 1-graphs, i.e. digraphs having no multiple arcs between any given pair of vertices.

2.4. Sequencing with erroneous data

The hybridization, as other biological experiments, usually ends with several errors in the output data. We distinguish two main kinds of error: false negatives and false positives. The false negative is a missing element in a spectrum — when a substring of length l of an original sequence does not appear in the spectrum. False positives are incorrect elements of a spectrum — words, that in fact are not parts of an original sequence. Presence of errors in the spectrum makes the combinatorial problem of DNA sequencing strongly NP-hard, even if we restrict the errors to only one kind: only false negatives or only false positives [22]. It is a usual situation in bioinformatics, as discussed in [23].

The paper of Pevzner, which brought the aforementioned algorithm, contained also a proposition of a second algorithm for the case of SBH with false negatives only [4]. Each false negative causes the lack of one arc in the graph. The method looks for these missing arcs, transforming the problem to the searching for a flow in a network built upon a directed bipartite graph $K_{m,m}$. Firstly, vertices in the Pevzner graph having different in- and outdegree are identified. They will constitute the two partite sets of the bipartite graph $K_{m,m}$: one of vertices with greater indegree and the other with greater outdegree. The vertices are replicated in $K_{m,m}$ the number of times equal to the difference between in- and outdegree. Next, arcs are added to make the bipartite graph complete, starting from vertices with greater indegree, with weights equal to the shift of vertex labels in their maximal possible overlap. The weight corresponds to the number of arcs in a path which should be introduced into the Pevzner graph in order to realize such a connection. Supplementing $K_{m,m}$ with source and sink together with associated arcs and assigning to all arcs capacity 1 finalizes the network construction, see Fig. 3. In this network the flow of value $m-1$ and of the minimum total weight is looked for, and the chosen connections between vertices of $K_{m,m}$ are to be added to the Pevzner graph in the form of paths joining the vertices, according to the Pevzner's manner. Example 2 illustrates the procedure.

Example 2. Figure 3 (A) presents the Pevzner graph constructed for the spectrum from Example 1 with one false negative, TCAG: $\{$AGAG, AGTA, AGTC, CAGA, CAGT, GAGT, GTCA$\}$. The network built upon directed $K_{2,2}$ is shown in Fig. 3 (B). The weight equal to 3 means that the labels GTA and CAG (in this order) do not overlap at all and three new arcs would be introduced for such connection: (GTA, TAC), (TAC, ACA) and (ACA, CAG). Fortunately, there is a much better flow of value 1, which involves the arc (TCA, CAG), i.e. our missing element TCAG. After completing the Pevzner graph (the left part of the figure) one gets the graph as in Fig. 1 (B) and can look for an Eulerian path in it.

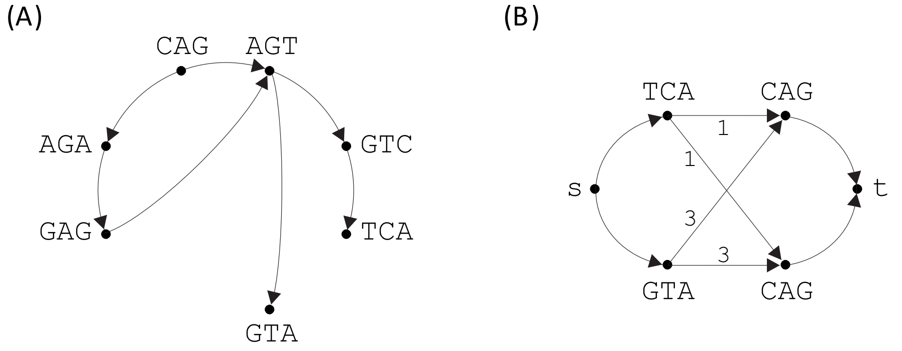


Figure 3: Graphs from the Pevzner algorithm for a spectrum with false negatives [4]. (A) The graph with missing arcs (here 1 missing arc). (B) The network constructed for identification of missing arcs.

The algorithm has a polynomial-time complexity, but actually it can be treated only as a heuristic. It does not find a solution in a few cases: when the minimum-cost flow does not have the value equal to the difference between $n-l+1$ and the spectrum cardinality, when a vertex with missing arcs has the same in- and outdegree, or when a completed Pevzner graph is not connected. If, in Example 2, the spectrum had one additional false negative CAGT, vertex CAG would be identified only as belonging (once) to the right part of $K_{m,m}$ and it would have no chance to be considered as the starting vertex of a missing arc. Then, the flow of minimum total weight would point to arc (TCA, CAG) and the reconstruction would end with a sequence shorter than required (AGTCAGAGTA).

The first algorithm accepting any kinds of error at the input was proposed by Blazewicz *et al.*, where the sequencing problem was modeled as a variant of the Selective Traveling Salesman Problem (STSP) in a complete directed graph [24]. In STSP vertices have assigned profits, arcs are weighted by costs, and the solution is a simple cycle on a subset of vertices with a maximum total profit, constrained by a given limit on the total cost. In the algorithm, instead of a cycle a simple path is searched for and all vertices have the same profit 1. Every element of a spectrum is represented by a vertex, arcs have costs equal to values of shifts in the maximum possible overlap between corresponding vertex labels, and the limit for the total cost is set to $n-l$. The path is equivalent to a sequence of a length not greater than n, composed of as many spectrum elements as possible. Vertices not included in a solution represent false positives, and shifts between labels greater than 1 are considered as false negatives. Apart from this, other works addressing the problem of erroneous data have been published, e.g. [25, 26, 27], but their detailed analysis is out of the scope of this article.

Example 3. This time let the spectrum produced for the original sequence CAGAGTCAGTA be $\{$ACCA, AGAG, AGTA, AGTC, CAGA, GAGT, GTCA$\}$, with two false negatives TCAG, CAGT, and one false positive ACCA. The matrix of costs of the complete graph from the method of Blazewicz *et al.* is shown in Fig. 4. There is no path of cost up to $n-l = 11-4 = 7$ including all 7 vertices, but there exist a few paths with 6 vertices, one of them corresponds to the original sequence.

3. Current models for NGS data

3.1. Next-generation sequencing

The practical limitation of SBH, which is the lack of scalability to longer DNA targets, has driven the development of next-generation sequencing methods. However, with the advent of NGS characteristics of the data sets have changed significantly

	ACCA	AGAG	AGTA	AGTC	CAGA	GAGT	GTCA
ACCA	-	3	3	3	2	4	4
AGAG	4	-	2	2	4	1	3
AGTA	3	3	-	3	4	4	4
AGTC	4	4	4	-	3	4	1
CAGA	3	1	3	3	-	2	4
GAGT	4	4	1	1	4	-	2
GTCA	3	3	3	3	2	4	-

Figure 4: Matrix of costs assigned to arcs of the graph from [24] for the spectrum from Example 3

and the assembly algorithms had to adjust accordingly. In contrast to SBH, the input sequences are now much longer (typically 75-500 nucleotides), may have different lengths and contain misreadings, i.e. single or multiple nucleotide insertions, deletions or substitutions. Moreover, the reads come from either of the two strands of the sequenced DNA, but no information is given from which one. The reads may also come in a form of so-called *paired-end* or *mate-pair* reads. These are pairs of reads for which an approximate separation distance is known. Furthermore, depth of the genome *coverage*, i.e. the average number of reads that align to, or „cover" each nucleotide of a given genome, is usually variable, even within the same experiment. In addition, repetitive genome regions (i.e. transposable elements, short tandem repeats and large segmental duplications) became a major problem as the target length of a genome to assemble had increased tremendously. This was achieved at the expense of huge amount of data generated in NGS experiments. All these properties make the DNA assembly problem well known for its high computational complexity. Even in the case of error-free data coming from a single DNA strand the problem is strongly NP-hard (cf. a similar problem of the Shortest Common Superstring [28]). Finally, unlike in the SBH error-free model of Lysov et al., graphs created here do not have in general the useful property of being directed line graphs. This is because of the error types present in NGS data, and more precisely, because of the inexact matching allowed between reads. As a result, the models presented in the previous section have evolved and are presented below.

3.2. Overlap graphs

Overlap graphs may be seen as a modified version of Lysov's model, only adapted to the standards of NGS data. Each read is represented as a vertex of directed graph *G* and the overlapping sequences are connected by arcs that indicate the overlapping direction, like in the original model. One of the major differences is how the overlaps between individual reads are defined. Since the reads are much longer, as compared to SBH, the overlapping regions typically cover only a small fraction. Moreover, the overlapping sections do not have to match perfectly, because of the misreadings. As a result, there are some arcs that are more confident than others. Hence, they are usually weighted, with the weight corresponding e.g. to *alignment* score and its length (alignment refers to an arrangement of two sequences in which similar regions are matched together; for a detailed definition see [29]). An example of the overlap graph is illustrated in Figure 5.

Additionally, since reads come from both strands of the DNA double helix, each vertex usually has also a dual vertex corresponding to the reverse complement version of a given read. This would not be necessary, assuming an algorithm could determine the original strand for each read. However, this is hard to achieve in practice. Therefore, the traversing algorithms usually apply the dual vertex approach, and mark both vertices as visited any time either of them is actually reached.

Likewise in the original model, finding a result in such defined graph *G* could be, at least in theory, done by finding a Hamiltonian path. This time, since the graph is weighted, the path ideally should satisfy the condition of the highest possible profit or the lowest possible cost, which is a variation of the Traveling Salesman Problem. There are, however, some practical problems associated with such attitude. First, due to errors in data and variations in the coverage depth it cannot be guaranteed that a Hamiltonian path exists at all. Graph *G* may be simply disconnected, e.g. due to sequencing gaps along the genome. In this case, a Hamiltonian path could be sought for each connected component of graph *G*, resulting in a set of disjoint *contigs*, i.e. continuous parts of a genome. Yet, the second problem is the size of the graph structure, which definitely prohibits any exponential algorithms. Therefore, only heuristic algorithms that try to cover the graph structure with relatively long paths are considered in practice. An

```
target DNA sequence:  AGATGTATTATTATTCTACGAGTGG
            read a:   AGATGTATTA
            read b:         ATTATTATTC
            read c:           TATTGTTCTA
            read d:              TATTCTACGA
            read e:                  CTACGAGTGG
```

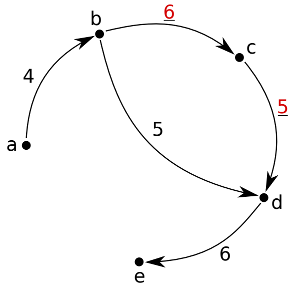


Figure 5: An example of the overlap graph constructed for five reads. Each arc connects the reads overlapping on a number of nucleotides (not lower than a predefined bound, here 4) and is labeled with a number corresponding to alignment score (here 1 point per 1 matched nucleotide). The scores that are affected by a mismatch (penalty -1 for a single mismatch) are underlined. For clarity, the reverse complement reads are not shown.

interesting exception is the work presented in [30], where authors propose to apply an exact algorithm solving the Minimum Path Covering Problem, which in general is NP-hard too, but easy for acyclic digraphs [31]. The main problem was to transform the overlap graph into an acyclic one, which was done by a heuristic method. However, the overlap graphs are usually far from being acyclic, which constitutes a real obstacle in this approach.

3.3. Decomposition-based graphs

The second type of graph model that is nowadays widely exploited in the context of NGS evolved from the concept presented by Pevzner, cf. Section 2.2. Even though it is commonly referred to as the de Bruijn graph model, such naming convention is not quite correct as we explained in Section 2.3.

In this model, the input reads are decomposed into a series of shorter fragments, all of the same length. These subsequences are often called k-mers, where k is a parameter for the length of each subsequence. Each k-mer is represented as an arc outgoing from a vertex labeled by its prefix of length $k-1$ and incoming to a vertex labeled by its suffix of length $k-1$. Thus, only exact matchings are allowed, like in the Pevzner's approach. Additionally, the reverse complement versions of the reads are considered here as well, for the same reason like in the overlap graphs. The decomposition technique used in this model is needed, because the original reads would be too long to overlap in an ideal way with any other read. Furthermore, a great portion of errors present in the NGS data set (e.g. single nucleotide misreadings) may be corrected with statistical models based on k-mer frequencies or by procedures for repairing the graph structure. An example of the decomposition-based graph is presented in Figure 6.

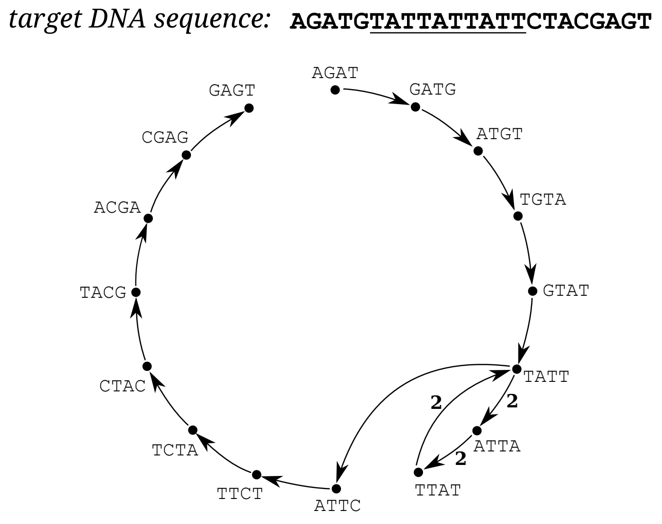


Figure 6: An example decomposition-based graph for a given sequence and $k = 5$. The underlined fragment of the sequence is a repetitive fragment that causes a cycle in the graph. The affected arcs are marked with a value representing their multiplicities (here 2).

In the decomposition-based graph approach, an arc may be thought of as a fragment of a read of length k, and a vertex – as a common subsequence of all the incident arcs. With such model, the DNA *de novo* assembly problem may be naively defined as finding an Eulerian path. However, one needs to remember that the graph may have no such a path due to the already described properties of NGS data. Moreover, since the k-mers are not unique within a given data set (because of the coverage depth and repetitive genomes) the problem is more often defined as finding a set of paths containing all the edges (or, more realistically, a vast majority) at least once. Interestingly, Pevzner et al. [32] noticed that any graph walk that contains all the reads as subwalks represents a valid assembly. Consequently, they formulated the DNA *de novo* assembly problem as finding the shortest superwalk. Medvedev et al. proved that such formulated problem is NP-hard [33]. Additionally, in order to address the problem of multiple occurrences of the same k-mers more precisely, the arcs in the described model should be labeled with the number representing the desired number of walks through each arc. This leads to the multiplicity of the arcs, and hence the original graph becomes a multigraph. Such defined problem is referred to in literature as *de Bruijn Superwalk with Multiplicities* and was proved to be NP-hard by Kapun and Tsarev in [34].

3.4. Models for paired-end reads

Most of the NGS data are produced now with the use of the paired-end sequencing protocol, as it was proven to be much more efficient for *de novo* sequencing than the standard single-read protocol. In the paired-end sequencing, instead of a set of single reads, one gets a set of paired reads placed close to each other in the target DNA, but in opposite strands. The distance between two reads from a pair (called the inner distance or the gap distance, usually of length of several hundred or thousand nucleotides) is a parameter of the protocol, but it can be set only approximately. The total length of a paired-end read, including the inner distance, is often referred to as *insert size*.

Till 2011 such kind of information was included into assembly methods as a secondary hint allowing to solve ambiguous walks or to order disjoint contigs, see e.g. [35, 36]. With reference to a graph model, the pairs of reads were represented as some bridges connecting corresponding vertices or greater parts of a graph. They did not influence the graph construction fundamentally.

Medvedev and co-workers proposed a new concept of so-called *paired "de Bruijn" graphs* which are graphs based on the decomposition of reads into k-mers, incorporating information about paired fragments [37]. Let us recall that these graphs, similarly as usual decomposition-based graphs, are actually not de Bruijn graphs, because they have incomplete sets of both vertices and arcs. The authors assume that the input pairs of reads come from the same DNA strand, i.e. are partially translated to their complementary counterparts. As it is not to be done in practice, they proposed duplicating the input data and building a double graph. The graphs of Medvedev *et al.* are constructed as follows. Pairs of reads are decomposed into pairs of k-mers: "left" k-mer in a given pair comes from the decomposition of the "left" read, and the "right" k-mer lies at the same position as the "left" one but within the corresponding "right" read. Arcs in the graph represent pairs of k-mers and vertices – pairs of their prefixes and suffixes, see Fig. 7 (A). The solution is an Eulerian path in the graph.

In the paper two types of such a graph were distinguished: when the inner distance d between reads is assumed to be exact (the same for all pairs of reads in the instance) or approximate, from a range $d \pm \Delta$. The *exact graph* is defined as in the above paragraph; the *approximate graph* is constructed from the exact one by gluing vertices whose "left" labels are the same and "right" labels overlap with the allowed shift $\pm\Delta$, see Fig. 7 (B). If $\Delta \geq \frac{1}{2}l$, instead of overlapping labels one must compute the shortest paths in the graphs between the vertices to be glued.

Such graphs are less tangled, in general, than the decomposition-based graphs for single reads, because of the requirement of overlapping both "left" and "right" k-mers from the pairs. Even if we allow for a big deviation Δ, which we expect in reality, the approximate graph will contain a number of Eulerian paths not greater than the usual decomposition-based graph build for the same original sequence and error-free data. The following example visualizes the models.

Example 4. For the original sequence TATTTATTACGTACG and for the inner distance between reads equal to 1, the pairs of reads (after translation into the same DNA strand) are: TATTT+TTACG, ATTTA+TACGT, TTTAT+ACGTA, TTATT+CGTAC, TATTA+GTACG. After their decomposition into pairs of 4-mers we get TATT+TTAC, ATTT+TACG, TTTA+ACGT, TTAT+CGTA, TATT+GTAC, and ATTA+TACG. The exact graph from [37] is shown in Fig. 7 (A). It has one Eulerian path corresponding to the original sequence. In order to obtain the approximate graph for $\Delta = 1$ one glues the vertices TTA+ACG and TTA+CGT, as they have the same "left" labels and "right" labels overlapping with the shift 1, see Fig. 7 (B). The approximate graph has two Eulerian paths corresponding to sequences TATTTATTACGTACG and TATTATTTACGTACG. For the comparison, the decomposition-based graph constructed without the information about pairing is shown in Fig. 7 (C). This time it enables reconstruction of the same DNA sequences as the approximate graph.

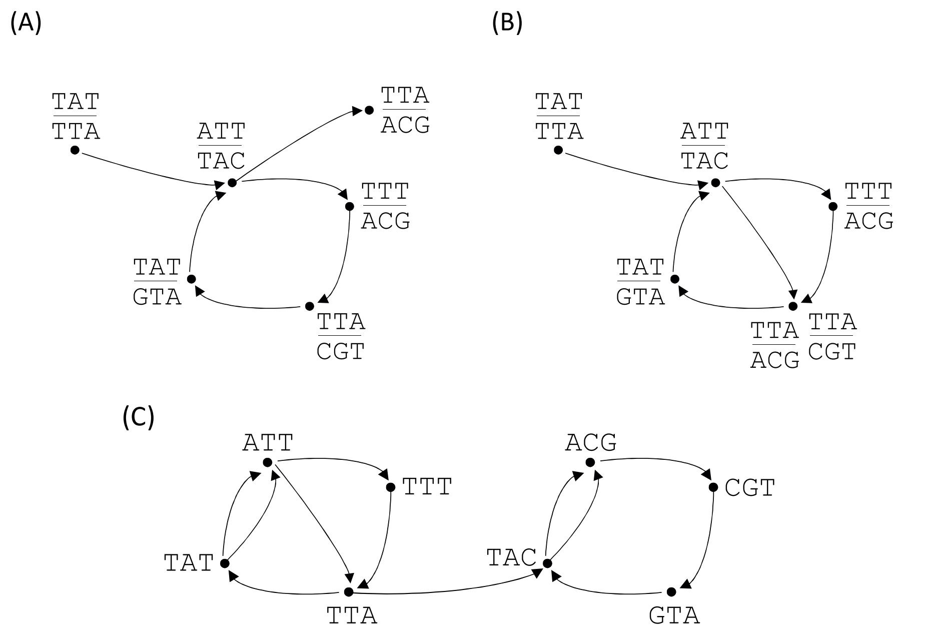


Figure 7: Graphs constructed for the data from Example 4. (A) The exact graph incorporating the information about pairing reads [37]. (B) The approximate graph with $\Delta = 1$ [37]. (C) The decomposition-based graph constructed for single reads

3.5. Discussion

Let us now shortly deliberate on some of the properties of the two most common graph types used in modern DNA *de novo* assembly. A careful reader observed that finding a solution in both models for the real NGS data is NP-hard. Nevertheless, in general both representations have very different properties that are worth mentioning here.

The number of nodes in the overlap graph increases linearly with the number of reads and therefore with the depth of coverage. The depth of coverage also has a direct impact on the number or arcs, which may increase proportionally to the squared coverage depth, and therefore can be very high. This can be especially noticeable for highly repetitive genomes, as multiple repetitions of the same fragment are in general seen as one fragment with increased coverage. All this usually results in high memory requirements. To give an example, a non-repetitive genome of length 50 Mbp sequenced with 100 bp long reads and $10\times$ coverage may result in an overlap graph with around 10 million nodes (including reverse complement) and 100 million arcs, whereas with coverage increased to $100\times$ – around 100 million nodes and 10 billion ($10 \cdot 10^9$) arcs. This could translate to memory requirements of around 800 MB and 75 GB for $10\times$ and $100\times$ coverage, respectively. However, it should be stressed that these numbers can vary depending on a given software implementation and applied optimizations, and as such should not be taken for granted.

On the other hand, the number of nodes in the decomposition-based graph does not depend so much on the number of reads or coverage depth since k-mer repeats are collapsed into a single node. As a result, it is mainly genome size that influences the size of the decomposition-based graph. This brings substantial memory savings, which are crucial for NGS large data sets. Again, to give an example, for a non-repetitive genome composed of 50 Mbp, the decomposition-based graph would consist of around 100 million arcs and roughly the same number of nodes. This could translate to memory requirements of around 1150 MB. One should keep in mind that these numbers already take into account the reverse complement sequences. A somewhat broader comparison of memory footprint of both graph models is presented in Figure 8.

Another disadvantage of the overlap graphs is their construction complexity. In order to detect the overlapping sequences, most of the methods perform sequence alignment. Whether or not this is done for all possible pairs of sequences, the alignment process is usually very time consuming. To some extent, this has been addressed by development of efficient alignment tools, e.g. [38, 39, 40, 41, 42], speeding up the whole process by a few tens to a few hundred times. To give an example, in order to determine $10 \cdot 10^9$ arcs in the aforementioned overlap graph, one would need to spend 2 hours, assuming 100 bp reads, very fast aligning algorithm running at sustained 140 GCUPS (Giga Cell Updates Per Second), and that only 10 candidates were considered for each resulting arc. To put this in context, the same computations performed in a serial manner on a CPU would take over 110 hours [38]. Even though parallel and

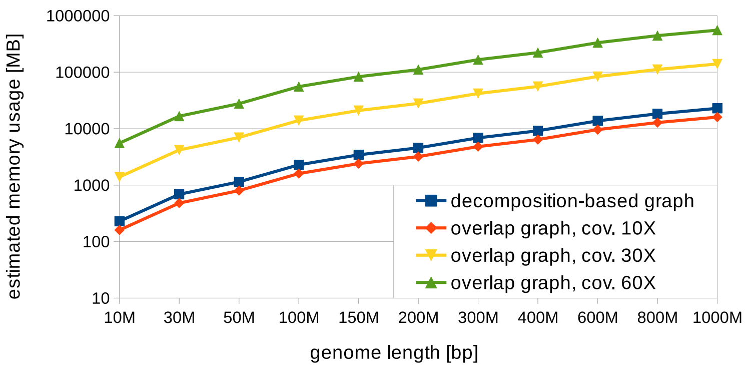


Figure 8: Estimated amount of memory needed for different graph models depending on genome size and sequencing coverage depth. A single node was calculated as 4 bytes representing an ID of a given read, and a single arc as 8 bytes: 4 bytes for the ID of an incident node and 4 bytes either for weight or multiplicity value for overlap and decomposition-based graphs, respectively. Reads are assumed to be 100 bp long. Note that the presented numbers may vary depending on implementation and optimizations used.

accelerated techniques have made a huge contribution in this area, the construction of overlap graphs still requires a considerable computational effort, which is not needed in the case of the decomposition-based graphs. Note that methods for alignment-free sequence comparison have also been investigated in literature, e.g. those based mainly on word frequencies [43]. In this context one can also apply methods based on problems related to string similarity, such as e.g. some variant of the minimum common string partition problem [44]. However, as heuristics, they cannot guarantee such a high precision as the alignment process.

The overlap graphs are typically used in algorithms based on the overlap-layout-consensus strategy. As the name suggests, once the overlaps between individual reads are detected and their layout is established by a traversing method, consensus contig sequences still need to be found. This is usually done by performing a sort of multiple sequence alignment guided by the traversing order and deciding about correct nucleotides in the case of ambiguity. In contrast, methods based on the decomposition-based graphs do not require the consensus step as the final contig sequences are inferred directly from the traversing order.

In return, the overlap graphs tend to deal well with sequencing errors and small heterozygous differences. This is achieved by allowing a few mismatches in the alignment process. The sequencing error correction and heterozygosity interpretation can be done in the consensus step. In contrast, the sequencing errors tend to introduce a lot of bias to the decomposition-based graphs as only identically matching *k*-mer overlaps are considered in this model. As a result, additional methods for reads error correction are indispensable here. Moreover, the heterozygous differences lead to multiple separate paths, which increase the complexity of the decomposition-based graph structure.

One of the major disadvantages of the decomposition-based graphs is that the information about the original reads is lost. Although the input reads are sometimes remembered and somehow referred to during a traversing procedure, such a process is heuristic and it does not restore the whole initial information, but significantly increases the complexity of an algorithm. This implies several limitations of this model. First, the relatively short *k*-mers seriously reduce the potential of using long reads in order to resolve genome repetition problems. In contrast, the information about reads is preserved in the overlap graphs and therefore they work much better with longer reads. This property may be extremely desirable while working on highly repetitive genomes. Second, in the scaffolding phase (where contigs are joined to create even longer assemblies) in the decomposition-based model it is necessary to map the reads to contigs in order to utilize the paired-end information. This is not necessary in the case of the overlap graphs, as such contigs already contain information about reads.

As a result, it is hard to conclude which of the two models suits best the requirements of NGS. In order to find out, we have prepared a practical comparison which is presented in the next section.

4. A practical comparison of modern *de novo* assemblers

In this section we juxtapose outcomes of the most popular implementations of *de novo* assemblers representing both methodologies, namely using the overlap graphs and the decomposition-based graphs. The main goal of this practical comparison is to see whether there are any substantial differences in the quality of the assembly results between these two theoretical models.

4.1. The assemblers of choice

The three software tools representing the decomposition-based graph model are: Velvet 1.2.10 [35], SOAPdenovo 2.04 [45] and Platanus 1.2.1 [46]. These packages seem to be extremely popular among the scientific community as of 2015. Velvet, despite being quite old (it dates back to 2007), is regularly updated and therefore it is recognized as a reference implementation among the community. SOAPdenovo, in turn, has gained attention due to its speed, memory efficiency and high quality assemblies. It was used, among others, to assemble the genome of the giant panda [47]. Finally, the newest algorithm – Platanus was published only in 2014 but interestingly was designed to assemble highly heterozygous genomes. All these tools were optimized to work well with the high-throughput NGS data. Additionally, one assembler that is not tested but deserves a special attention is ALLPATHS-LG [48]. This high quality assembler is also based on the decomposition-based graph model and is recognized across the world for its accuracy. However, it is not able to assemble paired-end data sets without additional long-fragment read library, and therefore is not considered in our comparison.

On the other hand, the implementations based on the idea of overlap graphs are less popular nowadays, mainly due to memory consumption that became especially high with the NGS data. Nevertheless, this model is still valued in many laboratories for putting the quality of the assembly first, when computational resources are not a limiting factor. A prime example of this family is Celera Whole-Genome Shotgun (WGS) Assembler, originally developed in 1999 and used to assemble the first whole genome shotgun sequence of a multi-cellular organism [49]. Over the years it was updated to support 454, Illumina and even PacBio/Oxford reads. In our test it is used in version 8.1. The second software from this class is String Graph Assembler (SGA) 0.10.13 [50]. In this case, the overlaps are calculated with the use of FM-index [51] which is a compressed full-text substring index based on the Burrows-Wheeler transform and can be used to efficiently find the number and locations of a pattern within the compressed text. Therefore the method itself is very efficient, in terms of both memory and time. Additionally, the authors transform the graph into a so-called string graph by removing transitive edges. In spite of that, the main idea of the overlap graphs is preserved and this application also represents the opposite front

Table 1: Basic information about the two data sets that were used to compare the algorithms for DNA assembly. * refers to reads before preprocessing, **refers to reads after preprocessing

Data set	1	2
species	*Homo sapiens* (human)	*Caenorhabditis elegans* (nematode)
considered chromosomes	14	all (7)
sequence length	107,043,718 bp	100,267,633 bp
reference genome in pieces	one continuous sequence	yes: 7
sequencing technology	Illumina HiSeq 2000	Illumina GA IIx
paired-end	yes	yes
avg. read length *	101 bp	107 bp
avg. read length **	99.7 bp	108.9 bp
reads № **	2 x 12,015,343	2 x 30,436,661
avg. depth of coverage **	26	66
insert size	159 bp	232 bp
σ of insert size	18 bp	56 bp

in our comparative study.

4.2. Testing methodology

All the algorithms were tested on two data sets: human chromosome 14 from the Genome Assembly Gold-Standard Evaluations [52] and *Caenorhabditis elegans* strain N2 (accession number DRA000967[1]). For these data sets the reference assemblies are well known and we use them to assess the quality of contigs and scaffolds produced by individual tools. This is a standard way of evaluating *de novo* assemblers. The main properties of both data sets are listed in Table 1. Although the lengths of the sequences to reconstruct are almost the same, these data sets are very different with respect to their repetitiveness and coverage depth, which makes them a good benchmark.

[1]*DRASearch* available at https://trace.ddbj.nig.ac.jp/DRASearch

Before the assembly process, the reads from both data sets were preprocessed in a standard way. Adapters that are specific to Illumina sequencers were removed from the reads. Reads containing unknown nucleotides (*N*) were completely removed. Moreover, *C. elegans* reads were trimmed and filtered to a minimum average quality value of 30 over the length of 60 consecutive nucleotides. These values for chromosome 14 reads were 20 and 30, for quality and length, respectively. Additionally, low quality bases were cut from the beginning and from the end of all reads. The average depth of coverage for chromosome 14 was calculated excluding the leading and trailing unknown nucleotides in the reference sequence. Hence, the value is a bit higher than would result just from the chromosome length and the number and length of preprocessed reads. This was not the case for *C. elegans*, whose reference genome is complete.

Each assembly software was run several times to adjust parameters. The resulting contigs and scaffolds were mapped to the reference genome using the gold-standard Bowtie [53]. This mapping tool was chosen since it offers an exact scoring scheme and semi-global alignment, which is exactly what was needed. The mapping information from the standard SAM files was used to calculate and present different statistics for each assembly method, separately for contigs and scaffolds. First, we present the information about the length and the quality of 10 longest contigs/scaffolds produced by each algorithm. The quality is calculated as percentage of identity between a contig/scaffold and its best mapping. Then, we present the percentage of genome length covered by contigs/scaffolds, the cumulative length of contigs/scaffolds wrt. the genome length, and the standard NG50 metric. The latter represents the length of a contig/scaffold that crosses the 50% of genome length, given that all the contigs/scaffolds are placed one after another and sorted from the longest to the shortest one. Moreover, we present the percentage of cumulative contigs/scaffolds length correctly mapped to the genome. This measurement strongly depends on the quality of the longest contigs/scaffolds. It is worth noting that extremely short contigs/scaffolds (shorter than 250 bp) are not considered in this section at all.

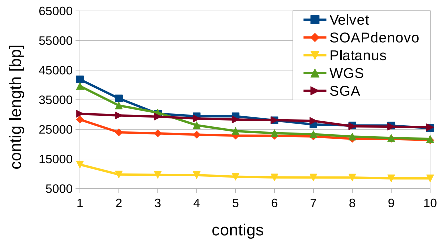
Finally, the amount of time and memory needed by each method is compared and discussed. The time represents the overall run time of a given method, i.e. the elapsed real (wall clock) time used by the process. The amount of memory was measured using the *VmRSS* (resident set size) information available in Linux in the $/proc/PID/status$ file associated with a given process.

The computational tests were done in Poznań Supercomputing and Networking Center on a cluster named *moss*. All the methods were run on the same hardware: a single node with double Intel Xeon CPU (E5-2670, 2.60GHz), 16 physical cores in total, and 512 GB of DDR3-1333 memory. Where possible, methods were run on all cores.

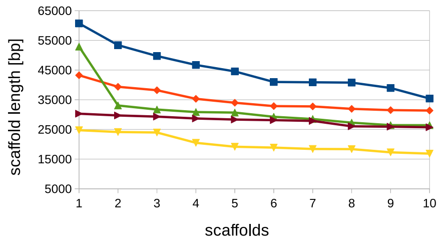
4.3. Comparative study results

First of all, we compare the length of the longest contigs and scaffolds produced by each method for both data sets, see Figure 9. It is plain to see that the longest contigs are produced by Velvet, but also WGS and SGA have very long contigs. On the other hand, by a large margin the shortest contigs were constructed by Platanus for the human chromosome. This could be because of the low coverage in this particular data set. Other assemblers seem to deal well with this type of input data. In the case of scaffolds, almost all methods succeeded to considerably extend their outputs. One exception is SGA that produced almost the same sequences for scaffolds and for contigs. We suspect that the scaffolding part of the SGA method in the tested version is not mature enough. It is also worth noting that whereas contigs/scaffolds constructed by some methods, e.g. SOAPdenovo, are more or less of the same length, their length tends to drop rather quickly for some other methods, e.g. Velvet.

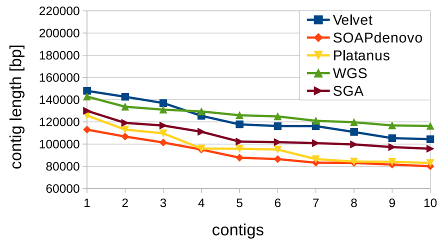
Tables 2 and 3 present the quality of ten longest contigs and scaffolds produced by each method for *H. sapiens* and *C. elegans* data sets. Interestingly, almost every method produced contigs of very high quality, usually above 99% of identity. The only contig that could not be mapped was constructed by SOAPdenovo. In contrast, on the scaffolding side not all the methods performed so well. In the case of the human chromosome, eight out of ten scaffolds constructed by Velvet could not be mapped, which is a very poor result. This method also failed to construct one scaffold in the second data set. On the other hand, one assembler that deserves a special attention is SGA. Although it did not extend the length of its contigs during scaffolding phase, it



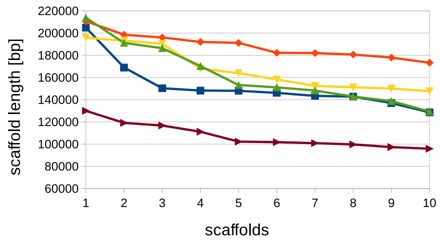
(a) *H. sapiens* contigs



(b) *H. sapiens* scaffolds



(c) *C. elegans* contigs



(d) *C. elegans* scaffolds

Figure 9: Length of 10 longest contigs/scaffolds for each method.

Table 2: Percentage of identity calculated for ten longest contigs and scaffolds produced by each method for the *H. sapiens* data set. n/m indicates that a given sequence could not be mapped to the reference genome, most probably because of its poor quality.

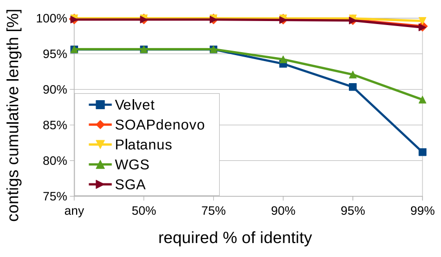
№	Contigs					Scaffolds				
	Velvet	SOAPdenovo	Platanus	WGS	SGA	Velvet	SOAPdenovo	Platanus	WGS	SGA
1	99.60	99.98	99.99	99.86	100.00	n/m	99.62	97.10	99.82	100.00
2	99.50	99.94	99.92	99.93	99.88	85.40	99.80	98.42	99.93	99.88
3	99.66	99.93	99.87	99.98	99.87	n/m	99.98	98.09	n/m	100.00
4	99.90	99.99	99.97	99.96	99.99	n/m	99.99	98.09	99.91	99.99
5	99.96	99.97	99.80	99.91	100.00	n/m	99.94	98.74	99.98	100.00
6	99.83	n/m	100.00	99.98	99.86	n/m	100.00	97.93	98.51	99.86
7	99.22	99.97	99.75	99.93	99.87	n/m	99.94	97.62	99.79	99.87
8	99.84	99.99	99.82	99.98	99.99	n/m	99.73	98.53	99.89	99.99
9	99.41	99.82	99.89	99.96	99.83	n/m	99.74	96.86	99.80	99.83
10	99.89	99.98	99.97	99.89	99.80	97.63	99.42	98.08	99.96	99.80

Table 3: Percentage of identity calculated for ten longest contigs and scaffolds produced by each method for the *C. elegans* data set.

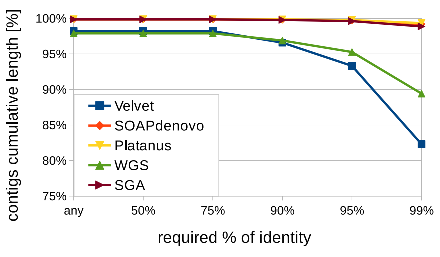
№	Contigs					Scaffolds				
	Velvet	SOAPdenovo	Platanus	WGS	SGA	Velvet	SOAPdenovo	Platanus	WGS	SGA
1	99.75	100.00	100.00	99.99	100.00	90.34	99.86	99.94	99.87	100.00
2	99.96	100.00	100.00	100.00	100.00	99.69	99.78	99.95	99.98	100.00
3	99.97	100.00	100.00	100.00	100.00	n/m	99.89	99.98	99.88	100.00
4	99.97	99.99	100.00	99.99	100.00	99.66	99.90	99.88	91.95	100.00
5	99.97	98.45	100.00	99.74	100.00	99.50	100.00	99.97	99.99	100.00
6	99.74	100.00	99.99	99.97	100.00	99.67	99.61	99.96	99.94	100.00
7	99.97	100.00	100.00	100.00	100.00	99.77	99.94	99.99	99.99	100.00
8	99.95	100.00	100.00	99.98	100.00	99.87	100.00	99.89	99.99	100.00
9	99.97	100.00	99.30	100.00	100.00	99.86	99.83	99.86	99.93	100.00
10	99.92	99.99	99.99	99.99	100.00	99.60	99.97	99.96	99.99	100.00

has the highest rate of assemblies mapped to the reference genome in an ideal way. As it turned out, the *C. elegans* data set was in general easier to reconstruct. We attribute this to a better quality of its reads and a higher coverage depth.

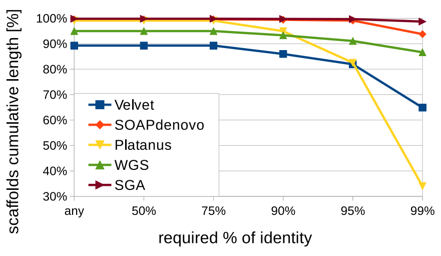
Charts presented in Figure 10 give some more insight into the quality of assemblies produced by each method. This time, we try to plot differences between individual methods based on the quality of all contigs/scaffolds, not only the longest ones. These charts present the percentage of cumulative contigs/scaffolds length mapped to the reference genome with a given percentage of identity. The higher quality is required, the shorter length of correctly mapped sequences. The largest drop can be observed in the case of Velvet and WGS assemblers. Interestingly, a huge drop was also reported for Platanus scaffolds of *H. sapiens*. Almost 49% of their cumulative length was between 95% and 99% of identity. This could also be seen in Table 2, where all the longest scaffolds were between 96.86% and 98.74% of identity. Additionally, the above-mentioned table shed a bad light on Velvet, but according to the current test over 89% of the scaffolds length produced by Velvet for the human chromosome was successfully mapped to the reference sequence. This indicates that Velvet has a problem mainly with the longest scaffolds. Another interesting finding is that both Velvet and WGS have a relatively low rate of total contigs/scaffolds length that map to the reference sequence at



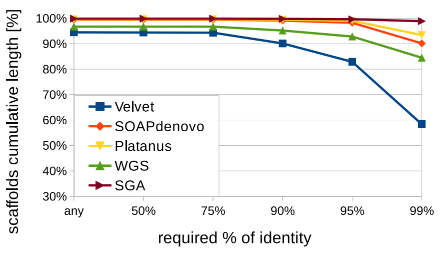
(a) *H. sapiens* contigs



(b) *C. elegans* contigs



(c) *H. sapiens* scaffolds



(d) *C. elegans* scaffolds

Figure 10: Percentage of total contigs/scaffolds length correctly mapped to the reference genome. Each contig/scaffold was required to map with a given percentage of identity (horizontal axis). Each time 100% on the vertical axis refers to the sum of all contig/scaffold lengths produced by a given assembler.

all. For example, in the case of *H. sapiens* contigs (Figure 10a), this is 95.61% and 95.65% for Velvet and WGS, respectively, whereas the other methods have between 99.81% and 99.98%. Similar results can be observed on the other charts. In contrast, Platanus, SOAPdenovo and SGA contigs represent a very high quality, as almost all of them map with more than 99% of identity. This trend is confirmed by the scaffolding results, with one exception – Platanus.

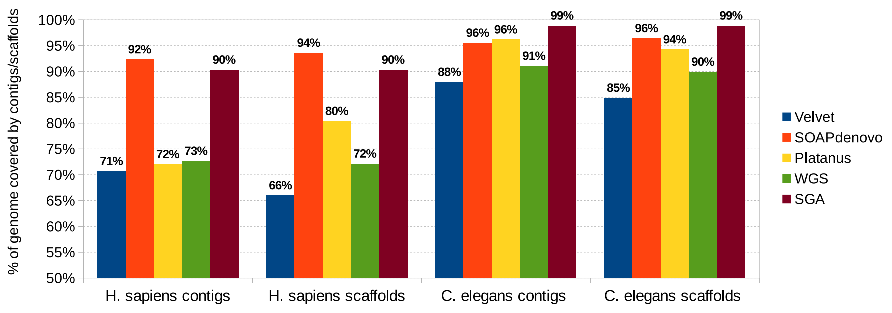


Figure 11: Percentage of the genome length covered by correctly mapped contigs/scaffolds.

An important characteristic of *de novo* assembly is the percentage of the reconstructed genome in terms of its length. For the data sets considered in this test the results are presented in Figure 11. In the case of the human chromosome, two methods stand out, namely SOAPdenovo and SGA, while the other methods performed much worse. For *C. elegans* the competition was much more tight. Not only Platanus has joined the best performing group, but both WGS and Velvet were able to cover much more reference sequence as compared to the first data set. However, the undisputed leader is SGA, covering 99% of the reference genome.

Another interesting property to look at is the total length of all contigs/scaffolds with respect to the genome size. This is presented in Figure 12. Not surprisingly, these values correspond quite well to the genome coverage, presented above. However, one can notice that for Velvet and WGS the genome coverage is always smaller than the total length of contigs and scaffolds. This is due to misassembled sequences. On the other hand, for SOAPdenovo and SGA this is the other way round, i.e. the genome coverage is always higher than the length of contigs/scaffolds. This is because some assemblies may map to the reference sequence more than once. It is considered that

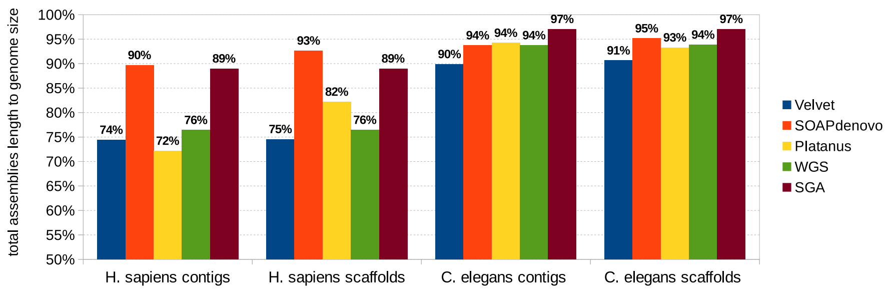


Figure 12: Total length of all contigs/scaffolds compared to the genome size.

the latter situation is more favorable for the assembly software. Ideally, however, the output from the software should cover every part of the reference sequence exactly once.

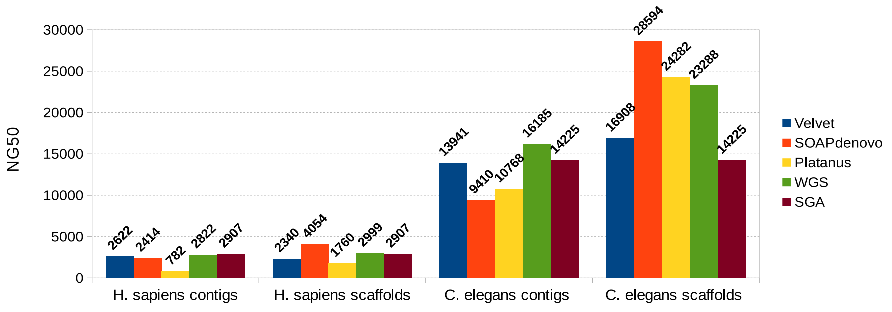
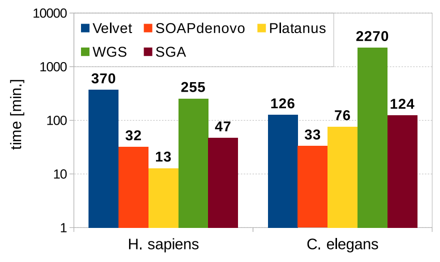
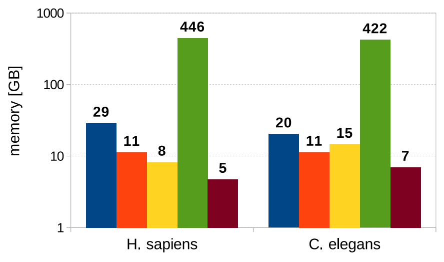


Figure 13: NG50 values calculated for each individual assembly.

Figure 13 presents the standard NG50 metric, calculated for each method. Clearly, much better values were obtained for the *C. elegans* data set. This is due to a better quality of reads and a higher coverage depth. The method that outperformed all the others when it comes to scaffold lengths is SOAPdenovo. Apparently, what it lacks in contigs it makes up for in scaffolds. Note that this method has the lowest NG50 value for *C. elegans* contigs, but it implements a very good scaffolding stage. Hence the good results. Because only those contigs/scaffolds that were correctly mapped to the reference sequence are considered here for NG50, it might be that this value is smaller for scaffolds than for contigs, as is the case with Velvet and the *H. sapiens* data set. Additionally, it is plain to see that for SGA the NG50 value remains the same for



(a) Time



(b) Memory

Figure 14: The amount of time and memory needed for each tested implementation to complete the whole computational process leading to scaffolds. Note: scales on both charts are logarithmic.

scaffolds as it is for contigs. This is the consequence of not extending scaffolds over the contig length, as discussed before.

Finally, we investigate the amount of time and memory needed to perform the calculations, cf. Figure 14. Whereas the implementations are quite comparable when it comes to the quality of results, they have very different consumption of resources. Both SOAPdenovo and Platanus might be considered as very fast implementations. For SGA and Velvet it takes a bit longer to compute the results. However, a really long run time was reported for WGS processing the *C. elegans* data set, nearly 38 hours. This is very long compared to 33 minutes needed for SOAPdenovo. On the other front, the leader of low memory consumption is SGA. However, SOAPdenovo, Platanus and Velvet are reasonable too. By far the highest amount of memory was used by WGS, up to 446 GB. Whereas some laboratories may sacrifice more time to finally obtain high quality results, the memory issues may be much harder to overcome, due to lack of specific hardware. This needs to be carefully considered, especially for large genomes.

4.4. Summary

The results of our comparative study presented above include representatives of both models: decomposition-based graphs and overlap graphs. Even though both models have their theoretical pros and cons, in practice their performance depends more on details of a particular implementation. As we saw, regardless of the graph model, we have both good and poor methods when it comes to the length and quality of assemblies. Similarly, when it comes to resources, the overlap graphs were traditionally considered more resource-hungry, which does not have to be true (cf. SGA). Moreover, the performance of the assembly software depends also on the properties of a given data set. As a result, it is hard to decide which model better suits the requirements of modern sequencing. Consequently, one is always advised to combine results from multiple assemblers when doing *de novo* assembly.

5. Conclusions and future trends

As stated at the beginning of this paper, computational biology is an interesting intersection of biology and computer science with a synergy effect. It was not only the biology that became a beneficiary of research in that field. The new models and algorithms developed in this context brought also a fresh insight into some aspects of computer science, especially graph theory. For example, a new class of graphs was defined, namely DNA graphs, for which the Hamiltonian cycle/path problem can be solved in a polynomial time. The properties of such graphs were further studied and the polynomial solvability of HCP was extended to the quasi-adjoint graphs (cf. Figure 2). Likewise, a lot of research was done to study the computational complexity of problems related to properties of the DNA graphs.

Nevertheless, we started our journey from the sequencing by hybridization problem, which was the first type of sequencing that involved an algorithmic approach, back in 1988. We thoroughly described the properties of both Lysov and Pevzner graph models, including erroneous data, in which case the problem becomes strongly NP-hard. We also presented how these models evolved to adapt to the next-generation sequencing. Both overlap and decomposition-based graphs constitute nowadays a robust foundation for modern DNA assembly methods. In order to find out which of these better suits the NGS data, we also carried out a practical comparative study. However, the presented results suggest that the quality of assemblies and even the resource requirements depend more on a particular implementation than on the type of underlying model. We also have concluded that with *de novo* assembly one is advised to use multiple assemblers and combine the results rather than rely on a single tool.

Let us now have a glimpse into the future trends. First of all, based on the exponential decrease of sequencing costs over the last years[2] we may expect the prices to become even lower in the near future. As a result, we may continue to observe exponential increase in the amount of generated data sets that need to be processed and analyzed. On the other hand, one may observe an exponential increase in the computing power of the world's fastest supercomputers too [3]. However, hardly any software can scale up to such large installations. They are not easily available for everyone either. On contrary, the computing power and amount of memory in smaller servers tend to increase much slower, more according to the Moore's law. Therefore, we perceive the stress for *de novo* assembly is more likely to be placed on highly time and memory effective software implementations, to allow for large instances on relatively affordable hardware. Even though, the current hardware limits of assembly software may be to some extent solved by future systems with more memory and computing power, algorithmic improvements seem to be indispensable as future data sets will become larger too.

Importantly, the current limitations of *de novo* assembly tools are not only associated with the large size of data sets, but also with the sequencing error rate and target genome repetitiveness. We anticipate that the latter may be overcome with the advent of much longer reads. For example, methods like PacBio RS II feature reads length of several thousands base pairs. Yet, this is achieved at the expense of high sequencing cost and relatively poor quality of reads. The short read sequencers have a raw error rate below 1% (before a typical preprocessing), whereas in the case of long reads this rate rises up to 10%. Fortunately, the error rate is being constantly reduced and the prices of new sequencing methods tend to drop rather quickly. Therefore, in the future we may expect to obtain data sets with both long reads and low error rate, which would be perfect for DNA assembly. However, before this happen, the current algorithms are not likely to deal well with such highly erroneous data, as they were designed for good quality reads.

Nevertheless, the above-mentioned high error rate of currently available long reads

[2]DNA sequencing costs: https://www.genome.gov/27541954/dna-sequencing-costs/
[3]The TOP500 project: http://www.top500.org/

does not necessarily mean that such data are useless. Several methods combining the advantages of both long but noisy and short and accurate reads were described by Koren and Phillippy [54]. For example, short reads mapped to the long ones may help to correct the sequencing errors. Obviously, such hybrid methods are costly, because they require two sequencing experiments. Nevertheless, this may currently be one of the best ways to go for difficult assemblies. Another option is to use a few additional mate-pair libraries with different insert sizes [55].

The high popularity of the decomposition-based graphs (as compared to overlap graphs) observed in the past years was mainly due to its favorable computation time and memory consumption. However, with longer reads becoming available this trend may change. Increasing market share of sequencers offering such reads should lead again to methods based on the overlap graphs. This is because the construction of the decomposition-based graphs basically results in a loss of the precious information contained in the long reads. Moreover, with parallel computing virtually becoming a commonplace and highly optimized software tools, the higher computational cost associated with the overlap graph methods no longer needs to be a discouraging factor.

Another issue that is becoming increasingly important, though not only in the context of DNA assembly, is the energy efficiency of computational methods. As the number of sequencing experiments is rapidly growing, the amount of corresponding computations is raising as well. Nowadays, it is crucial to minimize the energy consumption, because of both economical and environmental issues [56]. Although the energy efficient hardware architectures, e.g. GPUs, have been widely exploited over the past years in various research areas like video processing [57], medical imaging [58] or physical simulations [59], they were not yet sufficiently explored in the problem of DNA assembly. Therefore, we suspect that this will be one of the directions for development of new computational tools.

Finally, obtaining sequences of unknown genomes is not the only application for DNA *de novo* assembly. The genome of every single individual within a given species varies and it seems natural to test such unique arrangements, especially in the case of rare diseases. Nowadays, the majority of methods are based on the resequencing. However, developing very accurate and fast *de novo* assembly methods could revolutionize many NGS-based research areas such as: RNA-seq (transcriptome study) or structural variation of DNA, i.e. the variation in the structure of individual chromosomes. Significant improvement in this area would have a high impact on medical diagnostics and the medicine as a whole.

Acknowledgment

We are grateful to anonymous referees for their valuable remarks which significantly improved the presentation of the paper.

The research has been supported by grant No. 2012/05/B/ST6/03026 from the National Science Centre, Poland, and also by the PL-Grid Infrastructure in Poland.

References

[1] F. Sanger, S. Nicklen, A. Coulson, DNA sequencing with chain-terminating inhibitors, Proceedings of the National Academy of Sciences, U.S.A. 74 (1977) 5463–5467.

[2] E. Southern, Analyzing polynucleotide sequences, International patent application PCT/GB89/00460.

[3] Y. Lysov, V. Florent'ev, A. Khorlin, K. Khrapko, V. Shik, A. Mirzabekov, Determination of the nucleotide sequence of DNA using hybridization with oligonucleotides. A new method., Doklady Akademii Nauk SSSR 303 (1988) 1508–1511.

[4] P. Pevzner, l-Tuple DNA sequencing: computer analysis, Journal of Biomolecular Structure and Dynamics 7 (1989) 63–73.

[5] M. Margulies, M. Egholm, W. Altman, Genome sequencing in open microfabricated high density picoliter reactors, Nature 7057 (437) (2005) 376–380.

[6] V. Phan, S. Skiena, Dealing with errors in interactive sequencing by hybridization, Bioinformatics 17 (2001) 862–870.

[7] J. Blazewicz, P. Formanowicz, M. Kasprzak, W. Markiewicz, Sequencing by hybridization with isothermic oligonucleotide libraries, Discrete Applied Mathematics 145 (2004) 40–51.

[8] F. Preparata, E. Upfal, Sequencing-by-hybridization at the information-theory bound: an optimal algorithm, Journal of Computational Biology 7 (2000) 621–630.

[9] F. Preparata, J. Oliver, DNA Sequencing by hybridization using semi-degenerate bases, Journal of Computational Biology 11 (2004) 753–765.

[10] J. Blazewicz, A. Hertz, D. Kobler, D. de Werra, On some properties of DNA graphs, Discrete Applied Mathematics 98 (1999) 1–19.

[11] N. de Bruijn, A combinatorial problem, Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen 49 (1946) 758–764.

[12] J. Blazewicz, P. Formanowicz, M. Kasprzak, D. Kobler, On the recognition of de Bruijn graphs and their induced subgraphs, Discrete Mathematics 245 (2002) 81–92.

[13] R. Pendavingh, P. Schuurman, G. Woeginger, Recognizing DNA graphs is difficult, Discrete Applied Mathematics 127 (2003) 85–94.

[14] J. Hao, The adjoints of DNA graphs, Journal of Mathematical Chemistry 37 (2005) 333–346.

[15] X. Li, H. Zhang, Characterizations for some types of DNA graphs, Journal of Mathematical Chemistry 42 (2007) 65–79.

[16] X. Li, H. Zhang, Embedding on alphabet overlap digraphs, Journal of Mathematical Chemistry 47 (2010) 62–71.

[17] J. Blazewicz, M. Kasprzak, B. Leroy-Beaulieu, D. de Werra, Finding Hamiltonian circuits in quasi-adjoint graphs, Discrete Applied Mathematics 156 (2008) 2573–2580.

[18] J. Blazewicz, M. Kasprzak, Reduced-by-matching graphs: toward simplifying Hamiltonian circuit problem, Fundamenta Informaticae 118 (2012) 225–244.

[19] C. Berge, Graphs and Hypergraphs, North-Holland Publishing Company, London, 1973.

[20] N. Apollonio, P. Franciosa, A characterization of partial directed line graphs, Discrete Mathematics 307 (2007) 2598–2614.

[21] J. Blazewicz, M. Kasprzak, Computational complexity of isothermic DNA sequencing by hybridization, Discrete Applied Mathematics 154 (2006) 718–729.

[22] J. Blazewicz, M. Kasprzak, Complexity of DNA sequencing by hybridization, Theoretical Computer Science 290 (2003) 1459–1473.

[23] J. Blazewicz, M. Kasprzak, Complexity issues in computational biology, Fundamenta Informaticae 118 (2012) 385–401.

[24] J. Blazewicz, P. Formanowicz, M. Kasprzak, W. Markiewicz, J. Weglarz, DNA sequencing with positive and negative errors, Journal of Computational Biology 6 (1999) 113–123.

[25] J. Błażewicz, P. Formanowicz, F. Guinand, M. Kasprzak, A heuristic managing errors for DNA sequencing, Bioinformatics 18 (2002) 652–660.

[26] J. Blazewicz, E. Burke, G. Kendall, W. Mruczkiewicz, C. Oguz, A. Swiercz, A hyper-heuristic approach to sequencing by hybridization of DNA sequences, Annals of Operations Research 207 (2013) 27–41.

[27] J. Blazewicz, C. Oguz, A. Swiercz, J. Weglarz, DNA sequencing by hybridization via genetic search, Operations Research 54 (2006) 1185–1192.

[28] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, A series of books in the mathematical sciences, W.H. Freeman, San Francisco, 1979.

[29] J. Blazewicz, W. Frohmberg, M. Kierzynka, E. Pesch, P. Wojciechowski, Protein alignment algorithms with an efficient backtracking routine on multiple GPUs, BMC Bioinformatics 12 (1) (2011) 1–17.

[30] J. Blazewicz, W. Frohmberg, P. Gawron, M. Kasprzak, M. Kierzynka, A. Swiercz, P. Wojciechowski, DNA sequence assembly involving an acyclic graph model, Foundations of Computing and Decision Sciences 38 (2013) 25–34.

[31] J. Bang-Jensen, G. Gutin, Digraphs. Theory, Algorithms and Applications, Springer-Verlag, Berlin, 2007.

[32] P. Pevzner, H. Tang, M. Waterman, An Eulerian path approach to DNA fragment assembly, Proceedings of the National Academy of Sciences 98 (2001) 9748–9753.

[33] P. Medvedev, K. Georgiou, G. Myers, M. Brudno, Computability of models for sequence assembly, Lecture Notes in Computer Science 4645 (2007) 289–301.

[34] E. Kapun, F. Tsarev, De Bruijn superwalk with multiplicities problem is NP-hard, BMC Bioinformatics 14 (5) (2013) 1–4.

[35] D. Zerbino, E. Birney, Velvet: Algorithms for de novo short read assembly using de Bruijn graphs, Genome Res. 18 (2008) 821–829.

[36] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, J. Wang, De novo assembly of human genomes with massively parallel short read sequencing, Genome Research 20 (2010) 265–272.

[37] P. Medvedev, S. Pham, M. Chaisson, G. Tesler, P. Pevzner, Paired de Bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers, Lecture Notes in Computer Science 6577 (2011) 238–251.

[38] W. Frohmberg, M. Kierzynka, J. Blazewicz, P. Gawron, P. Wojciechowski, G-DNA – a highly efficient multi-GPU/MPI tool for aligning nucleotide reads, Bulletin of the Polish Academy of Sciences: Technical Sciences 61 (2013) 989–992.

[39] Y. Liu, A. Wirawan, B. Schmidt, CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions, BMC Bioinformatics 14 (1) (2013) 1–10.

[40] W. Frohmberg, M. Kierzynka, J. Blazewicz, P. Wojciechowski, G-PAS 2.0 – an improved version of protein alignment tool with an efficient backtracking routine on multiple GPUs, Bulletin of the Polish Academy of Sciences: Technical Sciences 60 (2012) 491–494.

[41] T. Rognes, Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation, BMC Bioinformatics 12 (1) (2011) 1–11.

[42] J. Blazewicz, W. Frohmberg, M. Kierzynka, P. Wojciechowski, G-MSA – A GPU-based, fast and accurate algorithm for multiple sequence alignment, J. Parallel. Distr. Com. 73 (2013) 32–41.

[43] S. Vinga, J. Almeida, Alignment-free sequence comparison – a review, Bioinformatics 19 (4) (2003) 513–523.

[44] C. Blum, J. Lozano, P. Davidson, Mathematical programming strategies for solving the minimum common string partition problem, European Journal of Operational Research 242 (2015) 769–777.

[45] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, J. Tang, G. Wu, H. Zhang, Y. Shi, Y. Liu, C. Yu, B. Wang, Y. Lu, C. Han, D. Cheung, S. Yiu, S. Peng, Z. Xiaoqian, G. Liu, X. Liao, Y. Li, H. Yang, J. Wang, T. Lam, J. Wang, SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler, Giga Science 18 (1) (2012) 1–6.

[46] R. Kajitani, K. Toshimoto, H. Noguchi, A. Toyoda, Y. Ogura, M. Okuno, M. Yabana, M. Harada, E. Nagayasu, H. Maruyama, Y. Kohara, A. Fujiyama, T. Hayashi, T. Itoh, Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads, Genome Res. 24 (2014) 1384–1395.

[47] L. Ruiqiang, F. Wei, T. Geng, The sequence and de novo assembly of the giant panda genome, Nature 7279 (463) (2010) 311–317.

[48] J. Butler, I. MacCallum, M. Kleber, I. Shlyakhter, M. Belmonte, E. Lander, C. Nusbaum, D. Jaffe, ALLPATHS: de novo assembly of whole-genome shotgun microreads, Genome Research 18 (2008) 810–820.

[49] E. Myers, G. Sutton, A. Delcher, A whole-genome assembly of Drosophila, Science 287 (5461) (2000) 2196–2204.

[50] J. Simpson, R. Durbin, Efficient de novo assembly of large genomes using compressed data structures, Genome Res. 22 (2012) 549–556.

[51] P. Ferragina, G. Manzini, Opportunistic Data Structures with Applications, Proceedings of the 41st Annual Symposium on Foundations of Computer Science (2000) 390–398.

[52] S. Salzberg, A. Phillippy, A. Zimin, GAGE: A critical evaluation of genome assemblies and assembly algorithms, Genome Research 22 (2012) 557–567.

[53] B. Langmead, S. Salzberg, Fast gapped-read alignment with Bowtie 2, Nature Methods 9 (2012) 357–359.

[54] S. Koren, A. Phillippy, One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly, Current Opinion in Microbiology 23 (2015) 110 – 120.

[55] C. Albertin, O. Simakov, T. Mitros, Z. Wang, J. Pungor, E. Edsinger-Gonzales, S. Brenner, C. Ragsdale, D. Rokhsar, The octopus genome and the evolution of cephalopod neural and morphological novelties, Nature 7564 (524) (2015) 220–224.

[56] M. Kierzynka, L. Kosmann, M. vor dem Berge, S. Krupop, J. Hagemeyer, R. Griessl, M. Peykanu, A. Oleksiak, Energy efficiency of sequence alignment tools – software and hardware perspectives, Future Generation Computer Systems (2016)`doi:doi:10.1016/j.future.2016.05.006`.

[57] S. Mahmoudi, M. Kierzynka, P. Manneback, K. Kurowski, Real-time motion

tracking using optical flow on multiple GPUs, Bulletin of the Polish Academy of Sciences: Technical Sciences 62 (2014) 139–150.

[58] M. Ciznicki, M. Kierzynka, K. Kurowski, B. Ludwiczak, K. Napierala, J. Palczynski, Efficient isosurface extraction using marching tetrahedra and histogram pyramids on multiple GPUs, Lecture Notes in Computer Science 7204 (2012) 343–352.

[59] M. Blazewicz, I. Hinder, D. Koppelman, S. Brandt, M. Ciznicki, M. Kierzynka, F. Loffler, S. E., J. Tao, From physics model to results: An optimizing framework for cross-architecture code generation, Scientific Programming 21 (2013) 1–16.