Computational Biology and Chemistry 33 (2009) 224–230

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/compbiolchem

Brief Communication

Whole genome assembly from 454 sequencing output via modified DNA graph concept

Jacek Blazewicz[a,c], Marcin Bryja[a], Marek Figlerowicz[c], Piotr Gawron[a], Marta Kasprzak[a,c], Edward Kirton[b], Darren Platt[b], Jakub Przybytek[a], Aleksandra Swiercz[a,c,*], Lukasz Szajkowski[b]

[a] Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland
[b] Lawrence Livermore National Laboratory, Joint Genome Institute, 7000 East Avenue, Livermore, CA 94550, USA
[c] Institute of Bioorganic Chemistry, Polish Academy of Sciences, Noskowskiego 12/14, 61-704 Poznan, Poland

ARTICLE INFO

Article history:
Received 25 November 2008
Received in revised form 29 March 2009
Accepted 23 April 2009

Keywords:
DNA assembly
454 sequencing
DNA graph

ABSTRACT

Recently, 454 Life Sciences Corporation proposed a new biochemical approach to DNA sequencing (the 454 sequencing). It is based on the pyrosequencing protocol. The 454 sequencing aims to give reliable output at a low cost and in a short time. The produced sequences are shorter than reads produced by classical methods. Our paper proposes a new DNA assembly algorithm which deals well with such data and outperforms other assembly algorithms used in practice.

The constructed SR-ASM algorithm is a heuristic method based on a graph model, the graph being a modified DNA graph proposed for DNA sequencing by hybridization procedure. Other new features of the assembly algorithm are, among others, temporary compression of input sequences, and a new and fast multiple alignment heuristics taking advantage of the way the output data for the 454 sequencing are presented and coded. The usefulness of the algorithm has been proved in tests on raw data generated during sequencing of the whole 1.84 Mbp genome of *Prochlorococcus marinus* bacteria and also on a part of chromosome 15 of *Homo sapiens*. The source code of SR-ASM can be downloaded from http://bio.cs.put.poznan.pl/ in the section 'Current research' → 'DNA Assembly'.

Among publicly available assemblers our algorithm appeared to generate the best results, especially in the number of produced contigs and in the lengths of the contigs with high similarity to the genome sequence.

1. Introduction

The DNA sequence assembly process is one of the most important problems in computational biology, and is known for its high complexity. The huge amount of data makes the problem very difficult to solve. The fact that data are erroneous and incomplete makes the problem even more difficult. Exact algorithms cannot be used in practice because of unacceptable computation time. Many teams worldwide try hard to provide heuristics producing satisfying suboptimal outcomes (see for example Kececioglu and Myers, 1995; Idury and Waterman, 1995; Jiang and Li, 1996; Pevzner et al., 2001; Chaisson et al., 2004; Sundquist et al., 2007).

The errors present in the input sequences to assembly problem are generated during a wet experiment and their number and character depend on the applied sequencing method. Several biochemical approaches have been elaborated to determine nucleotide sequences of naturally occurring DNA molecules. For the long time Sanger method (Maxam and Gilbert, 1977; Sanger et al., 1977) was commonly used. It is based on a standard primer extension reaction involving fluorescently labeled terminators of DNA synthesis. The DNA sequence is established during consecutive capillary electrophoresis of the reaction products. An additional method developed during the last decades was sequencing by hybridization (SBH) (Southern, 1988; Bains and Smith, 1988; Lysov et al., 1988; Drmanac et al., 1989; Guénoche, 1992; Blazewicz et al., 1999a, 2002; Zhang et al., 2003; Blazewicz et al., 2004, 2006). However, SBH never reached a highly advanced level allowing its wider application. Recently, DNA sequencing has been revolutionized by an introduction of novel massively parallel sequencing systems: 454 (Life Sciences) (Margulies et al., 2005), Solexa (Illumina) (Bennett, 2004) and SOLID (Applied Biosystems) (Fu et al., 2008). They are capable of generating a huge amount of sequencing data (from 100 million bases per run (454) to billion bases per run (Solexa and SOLID)) at less than 1% of the cost of capillary-based methods. Each system employed different strategy to generate high quality data. 454 relies on pyrosequencing of clonally amplified DNA fragments fixed to beads placed in water-in-oil emulsion. It produces 200–300 nucleotide sequence reads. Solexa and SOLID

* Corresponding author.
E-mail address: aswiercz@cs.put.poznan.pl (A. Swiercz).

1476-9271/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiolchem.2009.04.005

sequencing approaches are built around a very large number of short sequence reads (ca. 30–50 nt). In both systems individual short DNA molecules are attached to a solid surface and clonally amplified. During the next stage Solexa uses especially designed labeled nucleotides which can reversible terminate DNA synthesis and SOLID is based on sequential ligation with dye-labeled oligonucleotides. In all three systems the necessity for DNA cloning and high G-C content are not as much of a problem as in Sanger method. Because new sequencing approaches are massively parallel, they allow for detecting mutations at a low sensitivity level. On the other hand, the length of sequence reads causes the problems when highly repetitive regions are analyzed. In addition, new systems are very sensitive to contaminations. In this paper, we focus only on the 454 sequencing approach.

The specificity of the 454 sequencing data influences the assembly algorithm that is used at the computational stage of the sequencing process. In this paper, we propose a new assembly algorithm which deals well with these data and outperforms other assembly algorithms known from the literature and used in practice. The algorithm is a heuristics based on a graph model, the graph being built from the set of input sequences. The algorithm successfully utilizes some ideas coming from the procedures developed for the computational phase of the SBH (Blazewicz et al., 1999a). In particular, we use here a modified concept of the *DNA graph* (Blazewicz et al., 1999b), as well as Prize Collecting Traveling Salesman Problem (PCTSP), the latter being used for finding a consensus sequence (Blazewicz et al., 1999a). Since the assembly process is more complicated than SBH, we also use temporary compression of the input sequences, a new heuristic procedure for selecting promising pairs, operations repairing the lack of some arcs or excess of some vertices in the graph, and finally, the system of voting of sequences in creating the consensus sequence. The computational tests were performed on the data coming from real biochemical experiments at the Joint Genome Institute (JGI), which sought to sequence the whole genome of *Prochlorococcus marinus* bacteria (Chen et al., 2006)(accession number in GenBank is NC_007335) with the new 454 sequencing approach. Moreover, the performance of the algorithm was checked on a mammalian data—a part of chromosome 15 of *Homo sapiens* (accession number in GenBank is EU606050).

The organization of the paper is as follows. Section 2 contains the description of the new assembly algorithm proposed here—the SR-ASM algorithm. In Section 3 the computational results of the algorithm are compared with the outcomes of five other assembly algorithms used in practice and also with outcomes obtained at JGI, by a combination of two assembly programs and experts' finishing. The paper ends with Section 4.

2. Methods

In the classical approach to DNA assembly, algorithms merge sequences of lengths equal to a few hundreds of nucleotides in order to obtain an output sequence (or a series of disjoint contigs in case of lack of coverage in some places) up to a few million nucleotides long (see for example Kececioglu and Myers, 1995; Idury and Waterman, 1995; Jiang and Li, 1996; Pevzner et al., 2001; Blazewicz and Kasprzak, 2008). The input sequences come from both strands of the DNA. They are usually found by gel electrophoresis, sequencing by hybridization, or by capillary electrophoresis in one of its variations. The input sequences contain errors of several types: insertions, deletions, and substitutions of nucleotides, chimeras and contaminations. The average *depth of coverage* (i.e. the average number of reads covering a position) of the solution is usually 6–10.

The sequencing process in the novel approach invented by 454 Life Sciences Corporation, named the 454 sequencing, is performed automatically by a sequencer, which gives short DNA reads (usually 100–200 nucleotides long) with high average depth of coverage (even 30–40) (Margulies et al., 2005). The 454 sequencer protocol is based on measuring the intensities of light generated during the sequencing process. The sequences (reads) are presented as chains of nucleotides and in addition, the sequencer produces rates of confidence for every nucleotide. Similar to the previous sequencing methods, the produced sequences come from both DNA strands and include random insertions, deletions, substitutions of nucleotides, and contaminations. This time, however, the nature of the biochemical process excludes chimeras. Moreover, compared to other methods, the sequences have fewer insertion, deletion and substitution errors. The speed of generating the data gives a great opportunity to speed up the overall genome assembly process. However, the short reads need to be treated by a special assembly algorithm. In the last few years several new methods dedicated to short reads were implemented (e.g. Chaisson et al., 2004; Sundquist et al., 2007; Zerbino and Birney, 2008). In this paper we use two of them, i.e. VELVET from EMBL-EBI (Zerbino and Birney, 2008) and NEWBLER assembler (Margulies et al., 2005), for comparison with our method. VELVET is one of the newest assembly packages, designed for both Solexa and 454 technologies. The authors decided not to take into account the confidence rates of nucleotides, treating this information as rather unimportant. It will be interesting to compare this method with NEWBLER and our new algorithm, the more so as the latter methods exploit this information.

The motivation of our work was to invent a new assembly algorithm that performs well with data from the 454 sequencer together with the confidence rates. To check its usefulness we had raw data coming from real experiments using the 454 sequencer, undertaken at Lawrence Livermore National Laboratory (LLNL) operating jointly with the Joint Genome Institute (JGI). The data covered the whole genome of *Prochlorococcus marinus* bacteria, of length 1.84 Mbp. We also had results of the assembly process, obtained for this data by JGI, with the use of the programs Forge and Jazz together with experts' finishing (see Section 3). Moreover, additional tests on a part of chromosome 15 of *Homo sapiens* were done.

2.1. SR-ASM Algorithm

In the proposed SR-ASM algorithm (Short Reads ASseMbly) three phases can be distinguished. The first phase computes feasible overlaps for all input sequences (reads). The input parameters, among others, are the value of the minimum overlap between two sequences and the value of the *error bound*, the latter being the percentage of the mismatches allowed in the overlap of two sequences. These values are set by the user of the algorithm. *Feasible overlap* means here the overlap satisfying both parameters. The feasible overlaps are computed also for the sequences being reverse complementary to the input ones, due to the assumption that the reads come from both strands of a DNA helix. In the second phase, a graph is constructed with the reads and their reverse complementary counterparts as the vertices. Each read and its complementary counterpart create a pair of bound vertices. Here, we use a modified concept of the DNA graph invented in the context of the SBH (Blazewicz et al., 1999b). (The original *DNA graphs* (Blazewicz et al., 1999b) were defined as induced subgraphs of de Bruijn graphs (de Bruijn, 1946)) Two vertices being labeled by the corresponding reads are connected by a directed arc if there is a feasible overlap between these reads. We look for a path which passes through one of the vertices from every pair: either through the straightforward input sequence or its reverse complementary counterpart. In this procedure we use the idea of the Prize Collecting Traveling Salesman Problem (PCTSP), developed again in the context of the SBH approach (Blazewicz et al., 1999a). Usually it is not possible to find a single path in the graph and several paths are returned as solutions. At the end, in the third phase, a consensus sequence

(sequences) is determined with a new and fast multiple alignment heuristics.

In order to find overlaps for all n sequences (*the first phase of the algorithm*), we should compare all pairs of sequences. The number of such pairs is $O(n^2)$. The alignment of two sequences (a calculation of a possible overlap of two sequences) is performed with the Smith–Waterman algorithm (Smith and Waterman, 1981; Waterman, 1995) and takes time $O(k^2)$, where k is the length of the longer sequence. The complexity of this problem makes the exact calculation of the complete overlap matrix impossible in practice for big instances (big n). For example, around 3×10^{11} sequence comparisons are necessary to reconstruct a 2 Mbp bacterial genome with reads of 100 bp and average depth of coverage equal to 15. The computation time is therefore extremely long, but can be shortened by applying some strategies. For instance, the number of necessary comparisons of every sequence with every other sequence can be initially decreased. Only the selected pairs are then compared by the time-consuming Smith–Waterman algorithm. On the other hand, it is possible to speed up the alignment procedure; for example, the procedure can work in subquadratic time (Crochemore et al., 2003).

In our approach we applied both strategies. We introduced a heuristic method which estimates the quality of the alignment of two sequences, and selects the so-called *promising pairs* of sequences. We also achieve the acceleration of the alignment algorithm, by imposing thresholds of minimum overlap between two sequences and maximum error rate of the alignment as defined above.

The new heuristic algorithm for selecting the promising pairs of sequences works as follows. The sequences contain relatively few random insertions, deletions and substitutions. The most frequent errors in the 454 output data are the lack or the excess of nucleotides of one type in a subsequence of the same consecutive nucleotides. If during the comparison of two sequences we omit the information about the number of consecutive nucleotides of the same type (remembering only one nucleotide of each type instead of a few) then we can compare the sequences which contain less errors. Thus, before the proper alignment, the sequences are temporarily compressed by deleting all but one repeated nucleotides (e.g. keeping "A" instead of "AAA"). Next, we compare all pairs of sequences by checking for common substrings.

Let us define a *window* as a fragment of a sequence of a given, constant length (another parameter of the algorithm, usually set to 4–7 nucleotides). For every pair of compressed sequences we compare their respective sets of windows. The *leading sequence* in the alignment (with the Smith–Waterman algorithm) is defined as the leftmost one. From the leading sequence we determine windows in the initial part of the sequence (the initial part is another parameter of the algorithm, and may vary between 20 and 40 nucleotides). The quality of a window is evaluated according to the confidence rates of the first nucleotides in the series of consecutive nucleotides of the same type present in a window. If any of the confidence rates of these nucleotides is below a given threshold, then such a window is rejected. We denote the set of windows from the leading sequence (A) as W_A. Next, we look for the windows from the next sequence (B) in the alignment, which are present in set W_A. These windows constitute set W_B. At the end, we check whether or not the order of the windows' appearance is the same in both sequences. In order to do this we define W_{AB} as:

$$W_{AB} = \{s \in W_B : pos_A(pred_B(s)) < pos_A(s)\} \quad (1)$$

where $pos_A(s)$ is the position where window s starts in sequence A, and $pred_B(s)$ is the feasible window preceding s in sequence B. Then, *measure p*, which reflects a fitness of two sequences A and B to be considered as overlapping ones in the algorithm, can be defined by the equation

$$p = \frac{|W_{AB}| + 1}{|W_B|} \quad (2)$$

If p exceeds a certain threshold (given as a parameter for the algorithm) then the pair of sequences A and B are marked as promising. A *shift of an overlap* of two sequences is defined as the number of nucleotides of the leading sequence not overlapped by the second one. For all the promising pairs of sequences the optimal semiglobal alignment, represented by the *shift* between two sequences in the alignment and by the score (similarity), is calculated by the Smith–Waterman algorithm (+1 for every match, −2 for every mismatch, and −1 for every gap).

Having the alignments, we can now construct the graph which allows to find the path (or paths) (*the second phase of the algorithm*). The vertices of the graph correspond to the sequences and its directed arcs connect the vertices for which the alignment has been calculated and for which the overlap is feasible. Now, some additional operations have to be done within the graph to help find the longest path in the graph. The procedure of selecting promising pairs of sequences can result in omitting some important arcs. These operations allow to restore some arcs (if the neighborhood suggests a good connection which should be present there) or eliminate some vertices (if one sequence is included in the another sequence) (see Fig. 1).

Now we search for a path in the graph. Here, the algorithm uses some ideas derived from PCTSP, developed for the SBH approach (Blazewicz et al., 1999a). Starting from any vertex, we construct a path by joining the vertices with the smallest shift of the overlap of the corresponding sequences. If there are two candidate arcs with the same shift values then the one with the highest similarity of a candidate sequence and the previously added one is chosen. The path is constructed as long as any vertices can be added in both directions: either predecessors or successors of the path. Such a path is constructed for every vertex as the starting point. The longest path is chosen, and all the vertices from the path (and their reverse complementary counterparts) are deleted from the graph. If there are still some vertices left in the graph, other paths are constructed, as long as the length of the new path is greater than the *minimum path length* (another parameter of the algorithm).

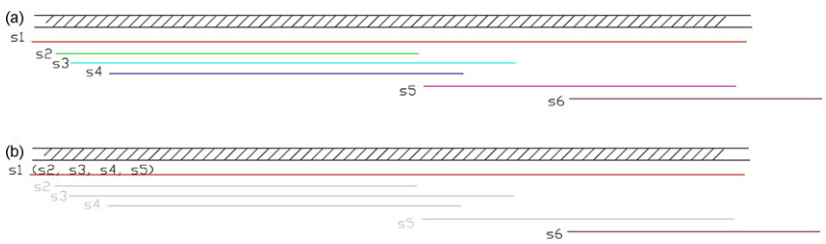

Fig. 1. An example of reducing some unnecessary vertices. (a) Sequences s2, s3, s4, s5 are subsequences of sequence s1. Consider a path constructed in the modified DNA graph which passes through vertices s1 → s2 → s3 → s4. Sequence s4 could not be connected with s5 due to weak overlap, but s1 could be possibly connected with s5 (in fact s5 is a subsequence of s1). (b) All the subsequences (s2, s3, s4 and s5) are included into the longer sequence (s1), and all the connections between other sequences are moved to sequence s1 (additional connection between s1 and s6).

Table 1
Results of the computational experiment for smaller instances (5×10^2–10^5 input sequences) of the genome of *Prochlorococcus marinus* for the PHRAP, TIGR-AS and CAP3 assemblers.

	PHRAP				TIGR-AS				CAP3			
Input seq.	No. of contigs	Qual [%]	Cov [%]	Time [s]	No. of contigs	Qual [%]	Cov [%]	Time [s]	No. of contigs	Qual [%]	Cov [%]	Time [s]
						98.70	22.34			99.50	41.42	
500	1	97.78	99.27	1.6	31	97.87	16.02	2.6	7	99.33	26.44	8.0
						98.52	14.01			88.08	24.90	
		96.93	90.43			98.56	12.78			91.55	27.26	
1000	4	99.32	9.66	3.5	52	99.26	11.91	6.6	11	99.93	21.97	17.3
		88.85	3.95			98.74	11.05			99.50	20.48	
		99.52	52.40			99.06	12.95			99.91	23.86	
2000	6	95.45	47.73	7.4	90	99.26	8.44	13.1	16	99.80	15.04	34.5
		88.85	1.98			99.21	7.42			99.93	11.01	
		97.78	99.27			99.00	5.68			99.91	9.67	
5000	7	88.85	0.80	19.2	198	99.06	5.25	44.4	33	99.83	8.80	88.6
		98.10	0.63			99.11	4.02			99.34	8.59	
		99.60	47.98			99.00	2.78			99.67	6.55	
10000	10	99.51	34.35	37.2	394	99.09	2.57	140.3	76	99.70	5.32	179.7
		99.52	10.38			99.56	2.34			99.91	4.73	
		99.60	24.20			99.00	1.40			99.76	3.39	
20000	22	96.48	18.63	84.5	817	99.06	1.29	604.1	184	99.65	2.96	408.7
		99.48	17.33			64.11	1.23			99.68	2.86	
		99.59	11.62			99.73	1.21			99.68	1.36	
50000	63	96.49	7.40	218.6	570	98.86	0.64	4767.2	493	99.75	1.34	1204.2
		97.95	6.94			98.81	0.59			99.69	1.31	
		73.17	6.74			98.31	0.37			99.57	1.01	
100000	135	98.69	6.48	487.7	622	99.34	0.30	24024.5	887	99.78	0.87	2822.9
		99.60	5.82			99.32	0.28			99.65	0.76	

After the completion of the above stage, we have a set of paths and a set of single sequences. The algorithm performs now the following steps as long as any changes can be made in the set of paths, in order to connect the disjoint parts of the current solution (both paths and single sequences). First, alignments for ends (last vertices) of the currently existing paths and single sequences are made both for their straightforward and reverse complementary encoding. Then, the ends with the best fit are overlapped. If one of the joined elements appears to be a single read, its reverse complementary counterpart is removed from further analysis.

The last phase of the algorithm involves composing the consensus sequence (or a series of contigs) from the obtained path (or set of paths, respectively). The multiple alignment algorithm is then trying to align optimally overlapping sequences (in order to create a contig). A leading sequence is chosen (at the beginning it is the first sequence in the path) together with these sequences which overlap it. Next, for each sequence the alignment with the leading sequence is constructed. All the alignments are put together, with respect to the leading sequence, and if there are any differences at a position, the nucleotides at that position in the sequences vote as follows. If a nucleotide of one type occurs at a given position in the alignment in at least 50% of sequences, then it is accepted in the consensus sequence, otherwise it is rejected. When the leading sequence finishes, it is replaced with the next one from the path, and the above procedure repeats until all the sequences (vertices) from the path are used. If there are more than one path the above steps are repeated for each path.

Paths composed of a fewer number of reads than the minimum path length are not considered. Our tests showed that due to this we lost only contigs of length not greater than 98 nucleotides which do not affect the 100% coverage of the genome.

3. Results and Discussion

The major part of our computational experiment consisted in tests with the use of raw data produced at JGI in cooperation with LLNL. The data cover the whole genome of *Prochlorococcus marinus* bacteria of length 1.84 Mbp. The output of the sequencer contains over 300,000 reads, each approximately 100 nucleotides long. The sequencer also provides rates of confidence of nucleotides.

In the computational experiment, apart from the complete set of data as described above, also smaller instances were prepared in order to do comprehensive comparison of various assemblers. For the complete set some methods were not able to perform correctly as either they did not finish the computations in a given time or their results were poor. The smaller instances are subsets of the complete set and cover continuous fragments of the genome. Thus, they theoretically should result in a small number of disjoint contigs. The model sequence of *Prochlorococcus marinus* is known and published (Chen et al., 2006)(accession number in GenBank is NC_007335), we used it only to evaluate the quality of produced contigs.

The tests were carried out on SUN Fire 6800 in Poznan Supercomputing and Networking Center. The assembly programs which were used for the comparison are widely known and publicly available: PHRAP (http://www.phrap.org/), CAP3 (Huang and Madan, 1999)(http://genome.cs.mtu.edu/), TIGR assembler (Sutton et al., 1995)(TIGR-AS, http://www.tigr.org/). From among the assemblers dedicated to short reads, which are publicly available, we chose VELVET as one of the newest ones (Zerbino and Birney, 2008)(http://www.ebi.ac.uk/~zerbino/velvet). The results obtained at JGI for the complete data set, with the use of programs Forge (http://combiol.org/talk/ForgeG/) and Jazz (the assembler of JGI) together with experts' finishing, are also presented here. For the same complete data we have had the results of NEWBLER assembler, being a 454 product attached to the sequencer, using flow signals of the sequencer instead of nucleotide sequences. To complete the comparison, we added our previous assembly program ASM (Blazewicz et al., 2004).

The criteria which have been used to compare the outcomes of the methods are: the number of contigs produced by the methods, the quality of the largest contigs, the coverage, N50 length, and the time of computations. The *quality*, i.e. the similarity of a contig to a fragment of the genome, was calculated by the Smith–Waterman algorithm with the score +1 for a match, −2 for every mismatch, and −1 for a gap. The *coverage* is the percentage of the total length of the reconstructed sequence which is covered by the contig. *N50 length* is the greatest possible value x such that 50% of the whole

Table 2
Results of the computational experiment for smaller instances (5×10^2–10^5 input sequences) of the genome of *Prochlorococcus marinus* for the ASM, VELVET and SR-ASM assemblers.

	ASM				VELVET				SR-ASM			
Input seq.	No. of contigs	Qual [%]	Cov [%]	Time [s]	No. of contigs	Qual [%]	Cov [%]	Time [s]	No. of contigs	Qual [%]	Cov [%]	Time [s]
		94.43	76.50			48.54	29.50					
500	8	84.26	25.73	7.7	11	49.53	15.76	0.1	1	99.86	100.00	2.0
		97.78	5.34			49.41	14.99					
		96.23	96.46			47.59	17.53					
1000	6	80.38	5.05	31.8	30	47.95	15.39	2.4	1	99.78	100.00	5.0
		97.92	2.11			49.42	9.87					
		97.20	48.86			47.76	10.70					
2000	18	90.15	26.86	128.0	65	47.80	7.69	3.9	1	99.75	100.00	10.0
		97.53	23.94			52.34	5.42					
		96.82	19.74			47.39	4.18					
5000	28	97.20	19.79	794.6	179	47.59	3.64	6.8	1	99.78	100.00	31.0
		97.08	17.92			48.72	3.55					
		96.91	17.68			47.90	2.57					
10000	74	96.82	9.66	3380.9	395	47.84	2.02	11.7	1	99.78	100.00	113.0
		97.20	9.68			47.39	2.01					
		97.42	10.72			99.79	1.71			99.69	43.65	
20000	147	90.83	9.23	11783.3	752	52.39	1.12	22.5	10	99.78	20.67	746.0
		97.04	8.31			47.03	1.05			99.76	14.80	
		97.23	4.84			99.90	0.57			99.82	10.17	
50000	360	97.10	4.78	81075.2	1959	99.95	0.53	53.7	31	99.63	10.10	5926.0
		97.40	3.69			47.90	0.52			99.79	9.13	
		97.36	4.91			99.96	0.39			99.82	10.42	
100000	700	97.52	2.96	351418.7	3873	99.95	0.29	108.0	76	99.72	9.60	22987.0
		97.30	2.79			99.90	0.29			99.66	6.94	

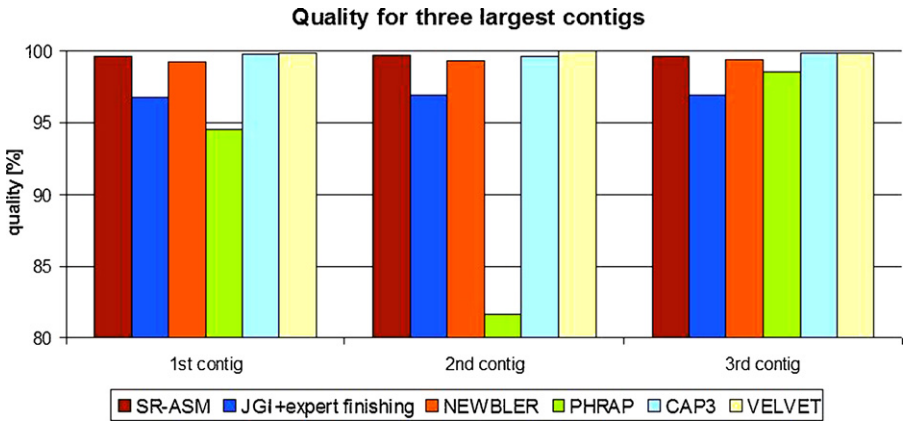

Fig. 2. Comparison of the quality of three largest contigs obtained by the SR-ASM, NEWBLER, PHRAP and VELVET assemblers, and obtained at JGI. The results concern the whole genome of *Prochlorococcus marinus* bacteria of length 1.86 Mbp. The input test set contains over 3×10^5 DNA reads.

reconstructed sequence is covered by produced contigs of length not less than *x*.

In Tables 1 and 2 results for the smaller instances of *Prochlorococcus marinus* are presented. Table 1 contains the results of PHRAP, TIGR-AS, and CAP3, Table 2 of ASM, VELVET, and SR-ASM. For the cases where the algorithms resulted in more than one contig, the quality and coverage values are given for three longest contigs. In the tables, "input seq." means the number of reads in instances, "qual" stands for the quality, "cov" for the coverage, and "no. of contigs" for the number of contigs produced by the assemblers.

Results of the tests performed for the whole bacteria's genome are shown in Table 3. Not all compared algorithms are presented there—ASM and TIGR-AS had unacceptable computation time. On the other hand, results from both sources from JGI (JGI after experts' finishing and NEWBLER) are added. The results from Table 3 are better visualized in Figs. 2–5 .

As we can see from the tables, the computation time of ASM grows rapidly with the instance size. The algorithm could generate a solution for 100,000 reads at most. However, it surpasses CAP3 and TIGR-AS in the length of produced contigs, with slightly worse similarity to the model sequence. ASM, similarly as CAP3, TIGR-AS, and PHRAP, does not use the information about confidence rates provided by the 454 sequencer. Taking into account all the "older" algorithms not dedicated to the 454 data, PHRAP seems to be the best one. PHRAP beats them in all the measures except the quality.

The small number of contigs obtained at JGI by the use of Jazz and Forge is due mainly to the finishing phase, where inappropriate contigs were rejected by experts. On the other hand, NEWBLER was not supported by the experts and generated very good solutions concerning all the criteria. However, SR-ASM reached almost so good results in the number of contigs, surpassing NEWBLER in both the quality and the coverage of the contigs. Solutions of PHRAP are quite acceptable except the number of contigs, but the behavior of VELVET was surprising. VELVET from EMBL-EBI is designed for both Solexa and 454 technologies, executing the program we set its parameters appropriately for the kind of input data we had (according to the personal communication with an author).

The second part of the computational experiment was done with the use of "mammalian" data, being of different characteristic than the bacteria ones. The data are stored in NCBI Short Reads Archive

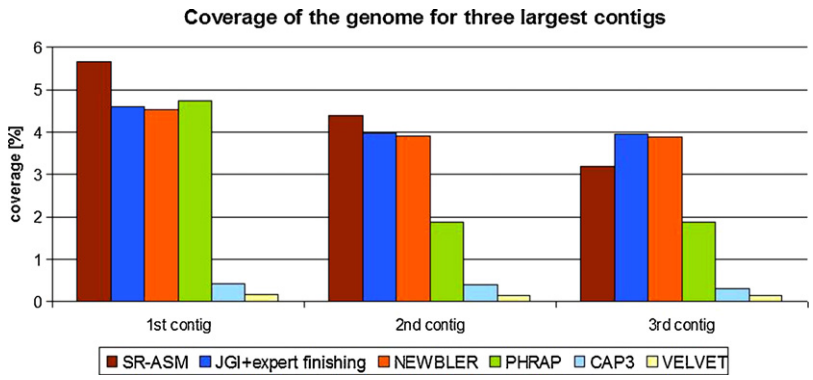

Fig. 3. Comparison of the coverage of three largest contigs obtained by the SR-ASM, NEWBLER, PHRAP and VELVET assemblers, and obtained at JGI. The results concern the whole genome of *Prochlorococcus marinus* bacteria of length 1.86 Mbp. The input test set contains over 3×10^5 DNA reads.

Table 3
Results of the computational experiment for the whole genome of *Prochloroccocus marinus* bacteria (over 3×10^5 input sequences) for the PHRAP, NEWBLER, CAP3, VELVET and SR-ASM assemblers, and the results obtained at JGI.

	No. of contigs	Quality [%]	Coverage [%]	N50 [bp]	Time [s]
		94.56	4.74		
PHRAP	5460	81.66	1.89	5241	2067
		98.56	1.88		
		96.85	4.60		
JGI after experts' finishing	150	96.94	3.97	15148	–
		97.00	3.95		
		99.29	4.53		
NEWBLER	149	99.38	3.91	22745	–
		99.45	3.89		
		99.85	0.42		
CAP3	7937	99.69	0.38	1517	12376
		99.91	0.32		
		99.90	0.17		
VELVET	13769	99.96	0.15	518	328
		99.90	0.14		
		99.71	5.66		
SR-ASM	171	99.76	4.38	10157	289513
		99.68	3.19		

(at http://www.ncbi.nlm.nih.gov/Traces/home/) and come from a biochemical experiment which was to cover a gap closure in chromosome 15 of *Homo sapiens*. The final sequence of length 11717 nucleotides has accession number EU606050 in GenBank. There was around 14,000 input reads, each one of length between 58 and 669 nucleotides. Because of significantly contaminated 3′ ends of the reads, we had to cut them from this side. Finally, we took

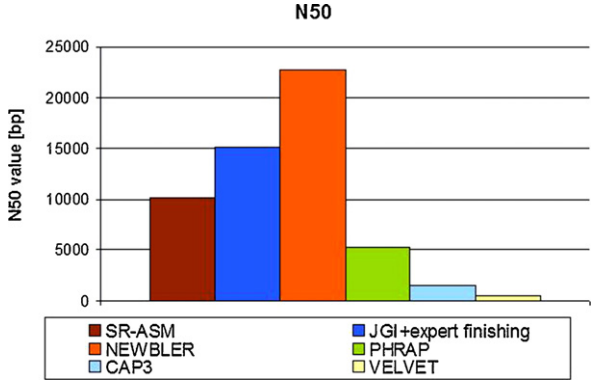

Fig. 4. Comparison of the N50 value obtained for SR-ASM, NEWBLER, PHRAP and VELVET assemblers, and obtained at JGI. The results concern the whole genome of *Prochlorococcus marinus* bacteria of length 1.86 Mbp. The input test set contains over 3×10^5 DNA reads.

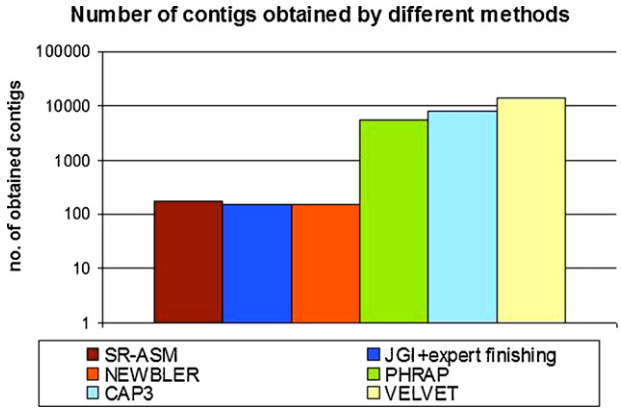

Fig. 5. Comparison of the number of contigs obtained by the SR-ASM, NEWBLER, PHRAP and VELVET assemblers, and obtained at JGI. The results concern the whole genome of *Prochlorococcus marinus* bacteria of length 1.86 Mbp. The input test set contains over 3×10^5 DNA reads.

200 leftmost nucleotides from the reads; reads shorter than 100 nucleotides were rejected.

The results of the second series of tests are presented in Table 4. This time we do not have NEWBLER's results since the data are stored in the form of nucleotide sequences (NEWBLER works only on flow signals from 454 sequencer, we had no access to such experiments). Also PHRAP is not taken into account, execution of it ended with an error ('fatal error: requested memory unavailable'; in fact, not more than 4 GB of memory can be allocated to the computational process on the 32 bit processor). From among the remaining

Table 4
Results of the computational experiment for the gap closure in chromosome 15 of *Homo sapiens* for the CAP3, TIGR-AS, VELVET and SR-ASM assemblers. Data are obtained from NCBI Short Reads Archive (at http://www.ncbi.nlm.nih.gov/Traces/home/). There are around 14,000 input reads.

	No. of contigs	Quality [%]	Coverage [%]	N50 [bp]	Time [s]
		98.72	89.04		
CAP3	330	95.52	13.89	10515	9582
		46.70	4.40		
		97.98	15.21		
TIGR-AS	3086	98.59	9.99	910	97202
		97.54	9.71		
		48.10	2.69		
VELVET	554	50.28	1.54	132	28
		50.75	1.71		
		98.56	100.00		
SR-ASM	48	79.09	5.27	11863	2814
		89.53	3.99		

four approaches, SR-ASM appeared to outperform the others. Our algorithm generated the whole resulting sequence of 100% of coverage and 98.56% of similarity to the model sequence. Concerning two other criteria, the number of produced contigs and N50, the results of SR-ASM are also the best.

4. Conclusions

In this paper, a new SR-ASM algorithm, dedicated to DNA assembly based on short reads, has been proposed. Computational tests with the use of raw data generated by 454 sequencer at JGI, covering the whole 1.84 Mbp genome of *Prochlorococcus marinus*, have shown its performance to be very good. Our algorithm produced better results, with respect to the coverage and to the similarity to the model sequence, than the NEWBLER's results or the results obtained at JGI with the support of experts' finishing. The performance has been further confirmed by additional tests on a fragment of chromosome 15 of human genome.

Summing up the results presented in Tables 1–4 and in Fig. 2–5, our algorithm outperformed all other approaches used in the tests. The strength of SR-ASM comes from the combination of the ideas coming from our SBH procedures and new propositions: temporary compression of input sequences, a new method of selecting promising pairs, operations adding some arcs or deleting some vertices in the graph, and finally, the system of voting of sequences in creating the consensus sequence.

The computation time of SR-ASM is rather long, but time is not crucial for assembly algorithms. SR-ASM solved the whole bacteria genome in 80 h, which is quite acceptable with respect to the time of obtaining these data in biochemical experiments. Currently we are working on a parallel implementation of this algorithm, which will significantly reduce the computational time.

SR-ASM is a new proposition, which can be used in practice in the whole genome assembly based on the 454 sequencing. It outperforms other assembly algorithms both dedicated and not dedicated to short reads. Moreover, combined with experts' finishing stage it has a chance to produce better results than the approaches for short-reads assembly currently used in laboratories.

Acknowledgments

We are very grateful to Krystyna Ciesielska and Richard Law for their helpful comments concerning the paper.

The research has been partially supported by the Polish Ministry of Science and Higher Education grant N N519 314635.

References

Bains, W., Smith, G.C., 1988. A novel method for nucleic acid sequence determination. J. Theor. Biol. 135, 303–307.
Bennett, S., 2004. Solexa Ltd. Pharmacogenomics 5, 433–438.
Blazewicz, J., Figlerowicz, M., Formanowicz, P., Kasprzak, M., Nowierski, B., Styszynski, R., Szajkowski, L., Widera, P., Wiktorczyk, M., 2004. Assembling the SARS-CoV genome—new method based on graph theoretical approach. Acta Biochim. Polonica 51, 983–993.
Blazewicz, J., Formanowicz, P., Guinand, F., Kasprzak, M., 2002. A heuristic managing errors for DNA sequencing. Bioinformatics 18, 652–660.
Blazewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W.T., Weglarz, J., 1999a. DNA sequencing with positive and negative errors. J. Comput. Biol. 6, 113–123.
Blazewicz, J., Formanowicz, P., Swiercz, A., Kasprzak, M., Markiewicz, W.T., 2004. Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries. Comput. Biol. Chem. 28, 11–19.
Blazewicz, J., Glover, F., Swiercz, A., Kasprzak, M., Markiewicz, W.T., Oguz, C., Rebholz-Schuhmann, D., 2006. Dealing with repetitions in sequencing by hybridization. Comput. Biol. Chem. 30, 313–320.
Blazewicz, J., Hertz, A., Kobler, D., de Werra, D., 1999b. On some properties of DNA graphs. Discrete Appl. Math. 98, 1–19.
Blazewicz, J., Kasprzak, M., 2008. Graph reduction and its application to DNA sequence assembly. Bull. Polish Acad. Sci. Tech. Sci. 56, 65–70.
Chaisson, M., Pevzner, P., Tang, H., 2004. Fragment assembly with short reads. Bioinformatics 20, 2067–2074.
Chen, F., Alessi, J., Kirton, E., Singan, V., Richardson, P., http://www.jgi.doe.gov/science/posters/chenPAG2006.pdf 2006. Comparison of 454 sequencing platform with traditional Sanger sequencing: a case study with de novo sequencing of *Prochlorococcus marinus* NATL2A genome. In: Plant and Animal Genomes Conference.
Crochemore, M., Landau, G.M., Ziv-Ukelson, M., 2003. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. SIAM J. Comput. 32, 1654–1673.
de Bruijn, N.G., 1946. A combinatorial problem. Koninklijke Nederlandse Akademie van Wetenschappen 49, 758–764.
Drmanac, R., Labat, I., Brukner, I., Crkvenjakov, R., 1989. Sequencing of megabase plus DNA by hybridization: theory of the method. Genomics 4, 114–128.
Fu, Y., Peckham, H.E., McLaughlin, S.F., Rhodes, M.D., Malek, J.A., McKernan, K.J., Blanchard, A.P., 2008. SOLID sequencing and Z-Base encoding. In: The Biology of Genomes Meeting. Cold Spring Harbour Laboratory, http://www.appliedbiosystems.com.
Guénoche, A., 1992. Can we recover a sequence, just knowing all its subsequences of given length? Comput. Appl. Biosci. 8, 569–574.
Huang, X., Madan, A., 1999. CAP3: a DNA sequence assembly program. Genome Res. 9, 868–877.
Idury, R., Waterman, M., 1995. A new algorithm for DNA sequence assembly. J. Computat. Biol. 2, 291–306.
Jiang, T., Li, M., 1996. DNA sequencing and string learning. Math. Syst. Theor. 29, 387–405.
Kececioglu, J.D., Myers, E.W., 1995. Combinatorial algorithms for DNA sequence assembly. Algorithmica 13, 7–51.
Lysov, Y.P., Florentiev, V.L., Khorlin, A.A., Khrapko, K.R., Shik, V.V., Mirzabekov, A.D., 1988. Determination of the nucleotide sequence of DNA using hybridization with oligonucleotides. A new method. Doklady Akademii Nauk SSSR 303, 1508–1511.
Margulies, M., Egholm, M., Altman, W.E., Attiya, S., Bader, J.S., 2005. Genome sequencing in microfabricated high-density picolitre reactors. Nature 437, 376–380.
Maxam, A.M., Gilbert, W., 1977. A new method for sequencing DNA. Proc. Natl. Acad. Sci. U.S.A. 74, 560–564.
Pevzner, P.A., Tang, H., Waterman, M.S., 2001. A new approach to fragment assembly in DNA sequencing. In: Proc. of the Fifth Annual International Conference on Computational Molecular Biology RECOMB'01, 256–267, Montreal. ACM Press.
Sanger, F., Nicklen, S., Coulson, A.R., 1977. DNA sequencing with chain-terminating inhibitors. Proc. Natl. Acad. Sci. U.S.A. 74, 5463–5467.
Smith, T.F., Waterman, M.S., 1981. Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.
Southern, E.M., Analyzing polynucleotide sequences. International Patent Application PCT/GB8900460, 1988.
Sundquist, A., Ronaghi, M., Tang, H., Pevzner, P., Batzoglou, S., 2007. Whole-genome sequencing and assembly with high-throughput, short-read technologies. PLoS ONE 2, e484.
Sutton, G., White, O., Adams, M., Kerlavage, A., 1995. TIGR assembler: a new tool for assembling large shotgun sequencing projects. Genome Sci. Technol. 1, 9–19.
Waterman, M.S., 1995. Introduction to Computational Biology. Maps, Sequences and Genomes. Chapman & Hall, London.
Zerbino, D.R., Birney, E., 2008. Velvet: algorithms for de novo short reads assembly using de Bruijn graphs. Genome Res. 8, 821–829.
Zhang, J.H., Wu, L.Y., Zhang, X.S., 2003. Reconstruction of DNA sequencing by hybridization. Bioinformatics 19, 14–21.