FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 38 (2013) No. 1

DOI: 10.2478/v10209-011-0019-4

DNA SEQUENCE ASSEMBLY INVOLVING AN ACYCLIC GRAPH MODEL

Jacek Blazewicz [ab] Wojciech Frohmberg [a] Piotr Gawron [ac]
Marta Kasprzak [ab*] Michal Kierzynka [ab] Aleksandra Swiercz [ab]
Pawel Wojciechowski [a]

Abstract. The problem of DNA sequence assembly is well known for its high complexity. Experimental errors of different kinds present in data and huge sizes of the problem instances make this problem very hard to solve. In order to deal with such data, advanced efficient heuristics must be constructed. Here, we propose a new approach to the sequence assembly problem, modeled as the problem of searching for paths in an acyclic digraph. Since the graph representing an assembly instance is not acyclic in general, it is heuristically transformed into the acyclic form. This approach reduces the time of computations significantly and allows to maintain high quality of produced solutions.

Keywords: DNA sequence assembly, acyclic graph model, GPU programming.

1 Introduction

The DNA sequence assembly, one of the most important problems of computational biology, is well known for its high complexity both on biological and computational levels. Huge amount of data, which are erroneous and incomplete, make the problem very hard to solve. Many teams worldwide put their efforts to provide time- and memory-efficient heuristics that produce satisfying results (see for example [13, 9, 12]).

The input data of the assembly problem come from the previous stage of the process of reading genetic information of organisms, namely the DNA sequencing.

[a]Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland
[b]Institute of Bioorganic Chemistry, Polish Academy of Sciences, Poznan, Poland
[c]Luxembourg Centre for Systems Biomedicine, University of Luxembourg
*The corresponding author: marta@cs.put.poznan.pl

A few years ago a new biochemical method of DNA sequencing, *454 sequencing* owned by Roche company (formerly by 454 Life Sciences Corporation), was introduced [10]. It gives highly reliable output of relatively low cost, in short time.

The specificity of the data from the 454 sequencing requires a specialized assembly algorithm. The aim of this paper is to present such an algorithm, partially implemented for GPU, which deals well with these data and outperforms other methods known from the literature. The algorithm is a heuristic based on a graph model, the graph being built on the set of input sequences. The novelty is present in the way the solution is searched for in the graph. Here, we utilize the polynomially-solvable problem of searching for paths in an acyclic digraph, unlike in common approaches to DNA assembly, where the graph is very tangled in general. Our graph also is not acyclic at the beginning, thus we use a heuristic transformation into the acyclic form.

The organization of the paper is as follows. After an introduction to the assembly problem and the 454 sequencing approach in Section 2, the description of the graph model is presented in Section 3. Section 4 contains details of the assembly algorithm. In Section 5, results of computational tests on real-world biological data are presented, together with a comparison of other assembly programs. The conclusion is provided in Section 6.

The source code of the algorithm is available at http://bioserver.cs.put.poznan.pl/#/Research/Download/SR-ASM/37.

2 The assembly problem

The process of reading genomic sequences can be divided into three stages depending on the length of the analyzed sequence: *sequencing* — which is determining a sequence of nucleotides in a DNA fragment of a length up to one thousand nucleotides, *assembly* — combining the sequenced fragments into longer contigs (continuous regions in genomes), and *mapping* — placing the assembled contigs in proper chromosome regions. For smaller genomes (viruses, bacteria) the stage of mapping is omitted. The whole process of reading genomic sequences is fundamental and underlies further genomic analysis, like understanding gene functions and phylogenetic relationship of species.

The problem of *DNA sequence assembly* can be generally defined in the following way. We have a multiset of sequences over alphabet {'A', 'C', 'G', 'T'} as input. These letters stand for four nucleotides composing DNA chains. The input sequences may have different lengths (although several next-generation sequencers produce DNA fragments of the same length) and they are outcomes of the DNA sequencing process, performed by traditional gel-based laboratory experiments or by modern next-generation sequencers (454 Roche, Illumina, SOLiD). Unfortunately, the results of the sequencing usually contain misreadings and other errors, coming from biochemical steps as well as from limitations of a given sequencing program. Thus, inexact matches of sequences have to be allowed at the assembly stage. The input data come from both strands of an assembled fragment of the DNA helix. Thus, some of the input sequences have the opposite orientation than the others, and they

should be matched in accordance with the rule of complementarity. Complementarity means, that A in one strand has T at the same position in the other strand, and C has G, respectively. For example, the reverse complement for AACATG is CATGTT. The goal of the assembly is to compose the input sequences into one sequence, or a series of contigs, in a proper order, where the criterion of evaluation of the solution can be its length (minimized during computations) or its likelihood (maximized). Example 1 provides an illustration of the general sequence assembly problem with sequences on input containing errors and coming from both DNA strands.

Example 1. Let a set of the input sequences for the DNA assembly process be {AGCA, ATCAAGCAAC, GACTC, TAGAA, TTTGCC}. Due to the fact that we assume that the sequences may contain misreadings, we must allow for inexact matches. Furthermore, as they may come from any of the DNA strands, appearance of some reverse complements in the output is permitted. One of the possible results is shown in Figure 1. In printing the resulting sequence (the one above the line) the majority rule has been used, i.e. the character appearing the greatest number of times in a column is the winner.

```
TTAGCAAGGAACTCTA
----------------
TTTGCC  GA-CTC
  AGCA      TTCTA
  ATCAAGCAAC
```

Figure 1: A possible assembly of sequences from Example 1

The DNA sequence assembly problem is strongly NP-hard, even in the case of data without errors and derived from one DNA strand (compare with the shortest common superstring problem).

In the approach of 454 Roche, the sequencing process is performed automatically by a sequencer, which produces a set of DNA reads (DNA fragments) with great average depth of coverage and high single read accuracy [10]. Additionally to the reads, one gets rates of confidence for every nucleotide. Similarly as in other sequencing methods, the produced reads come from both DNA strands and include insertions, deletions, and substitutions of nucleotides with reference to the source. Dealing with these errors became the main goal of algorithms solving the computational part of the assembly process. Consequently, they cannot avoid incorporating some of these errors into their results.

3 Graph model

In a graph-theoretical model of the DNA assembly problem, used also in our approach, every DNA fragment (read) is represented by a vertex in a directed weighted graph. Two vertices are connected by an arc if they are likely to be neighbors in a potential

solution. It means that the fragments represented by the vertices overlap more or less accurately. Algorithms of more theoretical nature allow only for exact matches of the overlapping fragments. In practice, overlaps which are admitted are those whose number (or percentage) of mismatches between paired nucleotides is below an acceptable threshold. It can be calculated by classical dynamic programming algorithm for semi-global alignment (a variant of [11]). Weights on the arcs represent then the score of the alignment — the higher, the better.

An example of such a graph is shown in Figure 2. This graph is complete, but in practice — when a threshold on the score is defined — the weaker arcs are not present. Another practical problem, which is lack of knowledge which DNA fragments should be replaced by their reverse complements, is solved by doubling the graph. Every vertex has then a dual vertex standing for the reverse complement of its sequence, and arcs are added similarly. Finally, when searching for a path, only one vertex from every pair can be visited.

Example 2. In Figure 2 a graph modeling the following set of sequences is present: {ACCGTGT, CGGTGTG, GGTGTGG, GTGCTGGG}. To simplify the example, all sequences are assumed to belong to the same DNA strand (no reverse complement is taken into account), what in general cannot be done. Weights on arcs correspond to the scores of semi-global alignment of pairs of sequences, where every match of nucleotides gets +1 and every mismatch or gap gets −1. This instance can be assembled into one consensus sequence ACCGGTGTGGG, which can be obtained from the Hamiltonian path with the highest total weight.

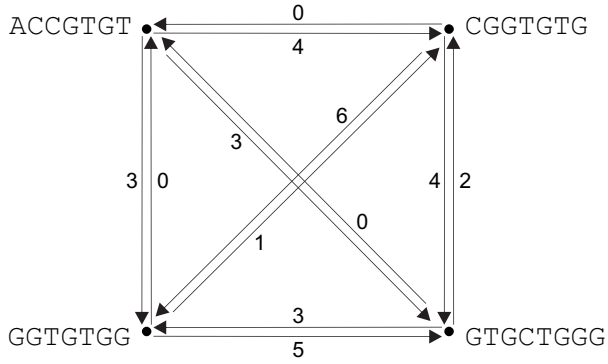

Figure 2: A graph representing the instance from Example 2

All of these properties imply that exact algorithms for the problem, or even heuristics, are highly inefficient for the usual number of input fragments — hundreds of thousands and more. A complete graph on n vertices requires $O(n^2)$ memory, hence some reduction in size is necessary. In our approach, a heuristic procedure chooses pairs of fragments which look similarly and computes alignments only for them. The incomplete graph makes no longer possible to find one continuous resulting path (another

reason can be incompleteness of information coming from the sequencing stage). Now, instead of looking for one path in a graph with the highest score, the algorithm needs to look for many long and good paths. This problem is similar to the *minimum path covering problem*, which is NP-hard in general but easy for acyclic digraphs (see e.g. [1]).

There are, however, some differences in the path covering problem in DAGs and the model used for solving the DNA assembly problem. The main issue is that the assembly graph is not acyclic. The second problem concerns dual vertices: only one node from a pair can be used. To make the graph acyclic we propose the following heuristic approach. First of all, after computing semi-global alignments, results with scores that indicate many errors (more than 8) are rejected. Setting the threshold to a lower value cannot be done due to sequencing errors existing in the input data. This reduces the number of cycles in the graph, but does not remove all of them. Then, all strongly connected components are separately transformed into acyclic forms.

To handle this task, two heuristic methods have been checked. The first one is an intuitive approach: removing the worst arc from every strongly connected component as long as it contains a cycle. The second one is more sophisticated: from every strongly connected component one vertex is chosen and split into two vertices, v_s with all arcs leaving the initial vertex and v_t with all incoming arcs. Next, the max-flow min-cut algorithm [7] is run to find a minimal cut needed to disconnect the source v_s and the sink v_t.

The second problem with dual vertices is solved by a similar heuristic procedure. Directions of arcs are temporarily disregarded (arcs replaced by pairs of antiparallel arcs) and if in any connected component there are both vertices from a dual pair, they become the source and the sink, and a cut disconnecting them is looked for.

At this stage the algorithm for finding minimum path covering can be applied. Using dynamic programming we get paths from the longest to the shortest. The algorithm guarantees that a current path is as long as possible, but henceforth it can result in decreased length of next paths.

4 Algorithm

We have recently invented a new assembly algorithm working very well with the data from 454 sequencer [2], but consuming noticeable amount of time. In this algorithm our ideas from previous approaches to DNA sequencing by hybridization were partially applied [3, 6]. Now, basing on a series of tests on raw data coming from a real experiment with 454 sequencer, we decided to implement our new algorithm with partial parallelization for GPU. Using a GPU as a computational platform is relevant, due to huge amount of data being the input for the assembly process; many independent tasks — here alignment of pairs of sequences — can be executed simultaneously.

A set of sequences which are a few hundred nucleotides long and rates of confidence for every nucleotide from the set constitute the algorithm's input. As the sequences can come from either of the two complementary strands of the analyzed DNA fragment, at the beginning of the algorithm the input set is doubled by reverse

complements of the input sequences, which are added to the data. (Later on, only one of the complementary fragments is used in the assembly process, as described in the previous section.) The rates of confidence constitute positive numbers corresponding to the level of reliability of a given nucleotide, and vary from 0 to over 30. The highest values are reserved for single nucleotides read by the sequencer without any doubt.

The algorithm is composed of three phases: computing overlaps of input sequences, building the graph together with searching for paths, and finally printing the resulting consensus sequence. The first phase in previous implementation took the longest time and it was the main part aimed for the parallelization. We incorporated a new module of semi-global alignment implemented for GPU [4], which significantly shortened this computational phase. (Similar GPU approach was sucessfully used for other problem, multiple sequence alignment [5], what inspired us to apply it here.) Only promising pairs of input reads have been selected to the semi-global alignment, on the base of their similarity in "windows" and also on the confidential rates.

Basing on the computed overlaps, in the second phase of the algorithm a graph is constructed with arcs corresponding to feasible overlaps between sequences in vertices. The arcs are weighted by the remembered shift of the overlap and the error of the connection. The error is the total number of penalties for mismatches between nucleotides in the overlap, calculated by our GPU algorithm. Next, a series of additional operations is done, which helps repair the graph in case the heuristic procedure of selecting pairs in the first phase fails. These operations can add an arc for a new pair classified now as promising, or remove some vertices corresponding to sequences contained in other ones.

Next, the graph is submitted to the procedure of making it acyclic, which has been described in the previous section. A solution for the minimum path covering problem, which is looked for in the reduced graph, is a set of paths passing through one of the vertices representing either the straightforward fragment provided in the input, or its reverse complementary counterpart. Every path corresponds to a contig and is constructed with the following optimization criteria used sequentially: first the minimum shift between neighboring sequences is used, then the minimum error of traversed arcs.

The third phase of the algorithm consists in printing the series of obtained contigs. Instead of the most frequently used majority rule, we decided to implement another one. Namely, if a nucleotide of one type occurs at a given position in the alignment in over 50% of sequences, then it is printed in the consensus sequence, otherwise, it is rejected.

5 Results

Computational tests have been performed on data coming from a real biochemical experiment with 454 sequencer, done in Lawrence Livermore National Laboratory cooperating with Joint Genome Institute. The aim of that experiment was to sequence the whole 1.84 Mbp genome of bacteria *Prochlorococcus marinus*. The output

Table 1: Results of DNA assembly of the whole genome of bacteria *Prochlorococcus marinus* obtained by newly implemented approaches

	Basic		GPU		Acyclic		Flow	
Contig No.	length	quality [%]	length	quality [%]	length	quality [%]	length	quality [%]
1	50362	48.18	42121	98.70	68088	68.75	70949	52.48
2	42865	97.84	40543	79.29	49012	98.59	68088	68.75
3	38290	64.02	36787	99.18	45484	99.28	49012	98.59
4	38271	99.09	35341	98.77	38634	96.62	45484	99.28
5	38212	99.06	35021	98.65	37043	61.09	38634	96.62
6	37920	96.80	33826	98.92	36644	98.73	37043	61.09
7	34906	99.35	33111	74.23	34881	99.14	36644	98.73
8	34248	99.33	31872	98.83	33623	99.57	34881	99.14
9	32959	98.10	29886	98.92	31807	98.56	33623	99.57
10	32855	99.42	28268	98.45	28804	98.57	31807	98.56
average	38089	90.12	34678	94.39	40402	91.89	44617	87.28
n50	16274		12915		17509		16227	
time [s]	13054		11251		2436		2520	

of the sequencer contained over 300000 sequences of length of about 100 nucleotides. Together with the sequences, their rates of confidence were provided.

In Table 1 collective results for all newly implemented approaches are shown. "Basic" is a modernized algorithm from [2], "GPU" differs from it in GPU procedure for semi-global alignment and distinct way of selecting promising pairs. "GPU" enhanced by the intuitive method of making the graph acyclic is named "Acyclic", and "Flow" when realizes max-flow min-cut algorithm. Ten longest contigs obtained by the programs are taken into account.

The quality, i.e. the similarity of the produced contigs to fragments of the genome, is quite high for all the implementations, but the best (on average) for "GPU". On the other hand, the mean length of these contigs is the smallest. The length and the quality are somehow connected, and the best pair of average values of these parameters is present for "Acyclic". The most advanced method "Flow" gives the longest contigs, but of lowest quality.

N50 is a popular biological measure for evaluating assembly results. If we order all the generated contigs with decreasing length, n50 is equal to the length of the first contig in this order satisfying the condition, that its length and lengths of longer contigs summed up, give at least half of the assembled DNA. So, the greater n50, the better the results are. Again "Acyclic" has the best value of this measure, also its computation time is the shortest.

Table 2: Results of DNA assembly of the whole genome of bacteria *Prochlorococcus marinus* obtained by other approaches

	PHRAP		NEWBLER		CAP3		VELVET	
Contig No.	length	quality [%]	length	quality [%]	length	quality [%]	length	quality [%]
1	19534	93,60	76956	99.29	7139	99.85	3121	99.90
2	18010	99,54	66481	99.38	6487	99.69	2737	99.96
3	17468	99,52	66153	99.45	5361	99.91	2551	99.90
4	14367	85,35	50293	99,31	5266	99,85	2468	100,00
5	14286	99,71	45002	99,49	5034	99,76	2305	100,00
6	14174	99,29	41745	99,40	4756	99,74	2148	99,88
7	14106	99,56	41642	99,48	4690	99,89	2002	99,90
8	14055	99,45	40488	99,41	4676	99,74	1958	99,90
9	12816	99,66	34780	99,45	4663	99,83	1916	99,95
10	12711	99,23	33585	45,71	4609	99,87	1911	99,82
average	15153	97,49	49713	94,04	5268	99,81	2312	99,92
n50	5241		22745		1517		518	
time [s]	2067		—		12376		328	

In Table 2 results of other widely known assembly programs are presented. The algorithms are PHRAP (www.phrap.org), NEWBLER (a commercial package), CAP3 (seq.cs.iastate.edu [8]), and VELVET (www.ebi.ac.uk/∼zerbino/velvet/ [13]).

The tests from Table 2 were performed in a different computational environment, thus the times cannot be directly compared with the ones from Table 1. As one can observe, NEWBLER produces the best results. Among academic approaches, our algorithms (in all presented variants) have the best achievements in the lengths of contigs, while other authors' programs have contigs of better average quality.

6 Conclusion

In this paper we have proposed a new algorithm for DNA sequence assembly, involving the acyclic graph model, which generally outperforms other widely known and previously published assembly algorithms. Only a commercial package, NEWBLER, distributed together with 454 sequencing equipment, has produced better results. Considering non-commercial academic methods, our algorithm in all its variants has proven to be the best in length of generated contigs. The contigs' similarities to the original genome are generally high with few exceptions, which are to be elim-

inated in our further work. Partial implementation for GPU reduced the overall computation time, however, further reduction in time could be made. Our research plans include, among others, adapting the whole program to a parallel environment without any loss of quality of its results.

7 Acknowledgements

The authors would like to thank Rudi Balling for his valuable help in the realization of this project. The research has been supported by grant No. FNR/11/AM2c/20 from the National Research Fund, Luxembourg, and grants No. DEC-2011/01/B/ST6/07021 and No. 2012/05/B/ST6/03026 from the National Science Centre, Poland.

References

[1] J. Bang-Jensen and G. Gutin, *Digraphs. Theory, Algorithms and Applications*, Springer-Verlag (2007).

[2] J. Blazewicz, M. Bryja, M. Figlerowicz, P. Gawron, M. Kasprzak, E. Kirton, D. Platt, J. Przybytek, A. Swiercz, and L. Szajkowski, Whole genome assembly from 454 sequencing output via modified DNA graph concept, *Computational Biology and Chemistry* 33 (2009) 224–230.

[3] J. Blazewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, and A. Swiercz, Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries, *Computational Biology and Chemistry* 28 (2004) 11–19.

[4] J. Blazewicz, W. Frohmberg, M. Kierzynka, E. Pesch, and P. Wojciechowski, Protein alignment algorithms with an efficient backtracking routine on multiple GPUs, *BMC Bioinformatics* (2011) 12:181.

[5] J. Blazewicz, W. Frohmberg, M. Kierzynka, P. Wojciechowski, G-MSA — a GPU-based, fast and accurate algorithm for multiple sequence alignment, *Journal of Parallel and Distributed Computing* (2012), in press.

[6] J. Blazewicz, F. Glover, M. Kasprzak, W.T. Markiewicz, C. Oguz, D. Rebholz-Schuhmann, and A. Swiercz, Dealing with repetitions in sequencing by hybridization, *Computational Biology and Chemistry* 30 (2006) 313–320.

[7] L.R. Ford, Jr. and D.R. Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics* 8 (1956) 399–404.

[8] X. Huang, A. Madan, CAP3: a DNA sequence assembly program *Genome Research* 9 (1999) 868-877.

[9] 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, and J. Wang, De novo assembly of human genomes with massively parallel short read sequencing, *Genome Research* 20 (2010) 265–272.

[10] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* 437 (2005) 376–380.

[11] S.B. Needleman and C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology* 48 (1970) 443-453.

[12] J.T. Simpson and R. Durbin, Efficient de novo assembly of large genomes using compressed data structures, *Genome Research* 22 (2012) 549–556.

[13] D.R. Zerbino and E. Birney, Velvet: Algorithms for de novo short read assembly using de Bruijn graphs, *Genome Research* 18 (2008) 821-829.

Received November, 2012