FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 41 (2016) No. 2
DE GRUYTER OPEN
DOI: 10.1515/fcds-2016-0007
ISSN 0867-6356
e-ISSN 2300-3405

G-MAPSEQ – A NEW METHOD FOR MAPPING READS TO A REFERENCE GENOME

Pawel Wojciechowski [1,2,3]*, Wojciech Frohmberg[1,3], Michal Kierzynka[1,3,4], Piotr Zurkowski[1,3], Jacek Blazewicz[1,2,3]

Abstract. The problem of reads mapping to a reference genome is one of the most essential problems in modern computational biology. The most popular algorithms used to solve this problem are based on the Burrows-Wheeler transform and the FM-index. However, this causes some issues with highly mutated sequences due to a limited number of mutations allowed. G-MAPSEQ is a novel, hybrid algorithm combining two interesting methods: alignment-free sequence comparison and an ultra fast sequence alignment. The former is a fast heuristic algorithm which uses k-mer characteristics of nucleotide sequences to find potential mapping places. The latter is a very fast GPU implementation of sequence alignment used to verify the correctness of these mapping positions. The source code of G-MAPSEQ along with other bioinformatic software is available at: http://gpualign.cs.put.poznan.pl.

Keywords: computational biology, next generation sequencing, parallel computing, reads mapping

1 Introduction

Methods for reads mapping to a reference genome were developed at the beginning of NGS data era as a necessary step of data processing pipelines. A formal definition of the mapping problem is described in [8]. The input data for this problem are: a reference sequence (genome) $S = s_1, s_2, ..., s_n$ and a set of reads R. Let us assume that all reads $r_i \in R$ are substrings of S, $r_i = S[t, t + l_i - 1]$, where l_i is the length of read i ($|r_i| = l_i$). The result of mapping of a given read r_i to genome S is a position t within sequence S where r_i begins its course. Unfortunately, such a definition is not

*corresponding author: Pawel.Wojciechowski@cs.put.poznan.pl; [1]Institute of Computing Science, Poznan University of Technology, Poland; [2]Institute of Bioorganic Chemistry, Polish Academy of Sciences, Poland; [3]European Center for Bioinformatics and Genomics, Poland; [4]Poznan Supercomputing and Networking Center, Poland

sufficient, because sequence S is in practice only a *reference genome*, which means it is a representation of a typical DNA for a given species. As a consequence there may be differences between r_i and $S[t, t + l - 1]$. These differences result from variations among individuals of the same species which come from the process of DNA evolution, e.g. from events like substitutions, insertions or deletions. Therefore, in the problem of reads mapping a specialized algorithm searches for a substring of S which is the most similar to r_i, usually using *alignment* algorithms.

The results may vary with respect to the strategy of finding mapping positions. In the rest of the paper we will refer to these strategies as to *mapping modes*. According to [8] we may distinguish the following mapping modes:

1. finding all possible mapping positions with the quality above some defined threshold,
2. finding all the best mapping positions,
3. finding up to a defined number of the best mapping positions,
4. finding up to a defined number of the best scored mapping positions.

where the *best mapping positions* are understood as a set of positions in sequence S with the highest quality of mapping (e.g. calculated as alignment score). As a consequence, all possible results obtained in mode 2 are indistinguishable. The type of applied mapping mode depends mainly on the purpose one wants to achieve and the type of processing method. Additionally, depending on the type of analysis, the reads for which the mapping positions are not unique may be excluded from results.

1.1 Other methods

There are many established methods for reads mapping. However, a detailed description of individual algorithms is beyond the scope of this paper. A more inquiring reader can read detailed surveys comparing different mapping tools, e.g. [5]. Nevertheless, to give the reader a basic idea how these algorithms work, a brief description of one of the most popular method – bowtie2 [11] is given below. Likewise many other methods, bowtie2 uses FM-index which was proposed by Ferragina and Manzini [3]. This type of index is based on the Burrows-Wheeler transform, and is a compressed full-text substring index. Hence, it can be used to find the number and locations of a pattern within the compressed text. The method itself is very efficient, in terms of both memory and time. Bowtie2 takes advantage of the above-mentioned properties and performs the following steps:

1. several constant length substrings (called *seeds*) are extracted from the original read and its reverse-complementary (RC) version; seeds are separated from each other by a specified shift,
2. each seed is mapped to the reference genome using the FM-index (without any gaps or mismatches),

3. a priority is assigned to each mapping position of a seed depending on the distance between its neighboring seeds. Next, the method tries to extend a randomly chosen seed (with the probability depending on its priority) to a whole read using again the FM-index,

4. reads are aligned to the reference genome around the mapping positions found in the previous step; the alignment is done by a dynamic programming method parallelized with SIMD CPU instructions.

1.2 The use of k-mer in biological sequence comparison

A k-mer can be defined as a substring of length k extracted from a longer sequence. K-mers are often used in comparison of biological sequences. Such techniques were proposed as a fast alternative to much more time-consuming alignment methods, but at the expense of accuracy. Some detailed reviews of k-mer algorithms for sequence comparison (as well as others approaches based on information theory) were presented by Vinga *et al.* [17], Reinert *et al.* [16] and Wan *et al.* [18]. The main idea of using k-mers in sequences comparison usually boils down to two stages. In the first phase each original sequence is transformed into a set of shorter substrings – k-mers. In the second step these sets are compared in order to evaluate the similarity of selected sequences. Such methods allow also the calculation of the distance between sequences. Furthermore, they were used to support the classification procedure [15, 21]. It is also worth noting that k-mer based methods are widely used in NGS-related research. For example, in [22] the authors show a highly accurate method for NGS reads comparison. Another example may be [13] where a method eliminating outlier reads during the data filtration stage of the processing pipeline was presented. To be more precise, the reads containing unique k-mers are treated by the method as invalid and are either corrected or discarded. Summing up, the k-mer based algorithms are widely exploited in the computational biology.

2 Methodology

G-MAPSEQ is an algorithm combining two interesting methods: ultra fast sequence alignment and alignment-free sequence comparison. Although the sequence alignment is very fast, mainly due to its efficient implementation on a graphics processing unit (GPU), still it would be impossible to align every single read to the whole genome. This is because of the size of real data sets. In order to save memory, one could try to align every read to every possible position within a given genome, but this would be highly intractable due to computation time. This is where the second module comes in – alignment-free heuristic for selection of so-called *promising pairs*. A promising pair is a pair of sequences that are supposed to be highly similar, or overlap each other. In our case, a promising pair is always a pair consisting of one read and one substring extracted from a reference genome. In order to find such promising pairs

our method computes so-called *k-mer characteristics* for all the sequences. The main assumption is that two similar sequences share similar characteristics. Therefore, the actual alignment, which is more precise but also much more computationally intensive, is applied only on a set of previously preselected pairs of sequences. With the alignment process the algorithm is able to precisely determine the similarity of a read and a given genome region, without any assumptions regarding seeds (which is typical for traditional mapping algorithms like bowtie2).

It should be stressed that the original idea of this method comes from an algorithm for graph construction in the DNA *de novo* assembly problem [9]. However, in this paper we propose a completely new application for this scheme. The steps of the proposed method are described below (see also Figure 1).

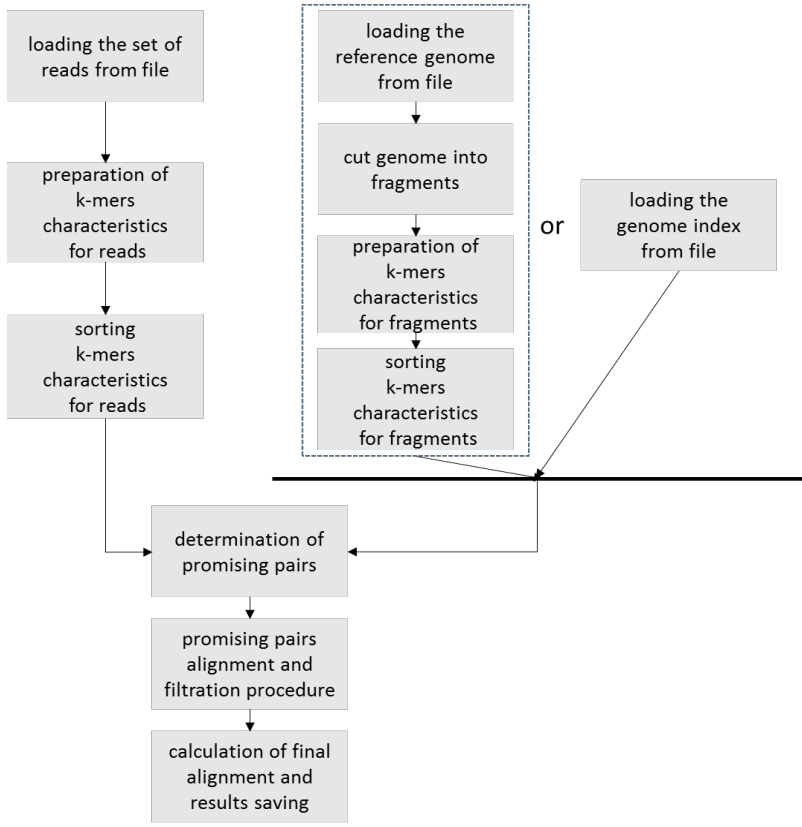

Figure 1. G-MAPSEQ - a block diagram of the method. Steps within the frame can be performed in a preprocessing stage and then load form a file.

Reference genome

In Next Generation Sequencing data reads come form both strands of the DNA double helix and no information is provided from which one.

Therefore, in order to find the correct mapping position, one of the following steps needs to be applied:

1. both versions of each read (forward and reverse complement) are be mapped to the reference genome, or
2. the genome itself is represented as two-stranded sequence.

The second option is more beneficial because the set of reads is usually much larger than the reference genome and therefore less time spent on conversion or memory is required.

The G-MAPSEQ uses the second strategy so the reference sequence is kept in two variants, forward and reverse complement. In order to save computational time, our method features genome preprocessing, the results of which, so-called indexes, can be saved to a file and reused in the future.

Input data

The standard input file format for NGS data is *fastq*. It contains DNA reads and corresponding quality scores calculated for every single nucleotide. The latter describes the level of certainty of sequencing method while reading a given nucleotide. One of the first standard steps in NGS data analysis is reads filtering. The goal of this task is to remove low quality reads (or their parts) and artifact sequences coming from biochemical experiments, i.e. adapter sequences added during sample preparation. Moreover, due to the quality loss at the end on reads, tail cutting is considered individually for each read. Likewise in the case of other mapping tools, the reads filtering is considered as a separate problem and therefore reads should be preprocessed beforehand. In order to highlight this, our method accepts input sequences, both reads and the reference genome, in the *fasta* format, i.e. without quality scores.

2.1 Preparation of k-mers characteristics

In the next step the algorithm computes the *k-mer characteristics* of reads. Without loss of generality, let us assume that a given set of reads R consist of n reads r_i, where $i = 1, ..., n$, all of the same length. In turn, the reference genome is a sequence S of length m. Read r_i of length l consist of $l - k + 1$ k-mers each beginning at the positions j, where $j = 1, ..., l-k+1$. Within each individual read the k-mers are sorted according to their frequency of occurrence. Such an ordered list of k-mers is called k-mer characteristic of a read. Preparation of such characteristics for the reference genome is slightly different. First of all, the algorithm creates the reverse complement representation of sequence S. Then, both sequences are cut into fragments of length

s with offset o, where s and o are parameters of the method. Next, such subsequences are processed in the same way as reads. As a result of this step two lists of k-mer characteristics are prepared: one for reads and one for reference genome and its reverse complement version.

2.2 Sorting k-mer characteristics

In this phase, both lists of k-mer characteristics are sorted lexicographically. This resembles words sorted in a dictionary. In our case k-mer characteristics may be compared to words and individual k-mers – to letters. For further consideration let L^R be a sorted list of k-mers for a set of reads R and L^S a sorted list of k-mers for the reference genome. $|L|$ is the number of elements on list L and L_i denotes i-th element of list L, where $i = 0, ..., |L - 1|$.

2.3 Determination of promising pairs

In order to determine the set of promising pairs P, both list are traversed simultaneously. Let L^R_{cr} and L^S_{cs} denote the currently visited elements of list L^R and L^S, respectively. One of the G-MAPSEQ parameters is *window size* w, which defines the number of promising pairs generated for each read. Every iteration, L^R_{cr} is compared to L^S_{cs} and depending of the result, the following computations are performed:

- $L^R_{cr} > L^S_{cs}$ – all pairs $(L^R_{cr}; L^S_i)$, where $i = cs - \frac{w}{2}, ..., cs + \frac{w}{2}$, are added to list P. L^S_i is called *neighborhood* of read L^R_{cr}. The following requirements need to be satisfied: $|L^S_i| = w$, $cs - \frac{w}{2} \geq 0$ and $cs + \frac{w}{2} < |L^S|$. For real-life data sets $w \ll |L^S|$, so it is easy to manipulate the borders of neighborhood (index i) to satisfy these requirements, i.e. $\forall cs : cs \in (0, \frac{w}{2}\rangle \implies i = 0, ..., w$ and $\forall cs : cs \in \langle |L^S| - \frac{w}{2}, |L^S|) \implies i = |L^S| - w, ..., |L^S|$.

  Next, if $cr = |L^R|$, which means that the last element of list L^R was processed, the procedure is finished. Otherwise, the current element of list L_R i.e. L^R_{cr} is moved to the next element: $cr = cr + 1$.

- $L^R_{cr} <= L^S_{cs}$ – if $cs < |L^S|$, the current element L^R_{cs} of list L_S is switched to the next one $cs = cs + 1$. Otherwise, if $cs = |L^S|$ (which means that it was the last element of the genome list), for all remaining elements of L^R i.e. L^R_l, where $l = cr, ..., |L^R|$, the following pairs are added to P: (L^R_l, L^S_b), $b = |L^S| - w, ..., |L^S|$.

It may happen that a read and a fragment of the reference genome do not belong to the same neighborhood, even though they overlap on a relatively long distance. This may be the case especially if two sequences are shifted relative to each other by a number of nucleotides. In order to avoid such a situation, so-called *partial characteristics* were introduced [9]. A partial characteristic is a regular k-mer characteristic only computed for a selected subsequence of a read or a genome fragment. The subsequences are usually extracted from the beginning, center and the end of a given

sequence. If two sequences have similar partial characteristics, they are likely to share the same region. As a consequence, additional promising pairs are generated based on sorted lists of partial characteristics – the principles are the same like in the case of original k-mer characteristics.

Additionally, the set of promising pairs P is enriched by a method called *greatest lexicographical index*. This method extracts, as the name suggest, lexicographically greatest substring of a fixed length from each sequence. Those sequences that share identical indexes become a promising pair. The main purpose of the method is to be able to detect those pairs of sequences that share a specific pattern regardless of its position within a given sequence. The length of the index is a trade-off between speed of the algorithm and accuracy, and in our case is usually 13.

Once all the promising pairs are determined, there are at least $|L^R| * w$ pairs of sequences in set P. The next section explains how this information is further processed.

Promising pairs alignment and filtration procedure

The main goal of this step is to determine which promising pairs represent correct mapping positions. Often times, a promising pair connects two sequences that are not similar enough to constitute a correct mapping, and therefore all preselected pairs need to go through a strict verification process. In order to eliminate low-scored pairs a special version of semi-global alignment is applied here, i.e. modified algorithm of Needleman-Wunsch [14]. The main difference is that while the original semi-global alignment does not take into account terminal gaps when calculating score, in our algorithm terminal gaps associated with the read do contribute to the score. This does not apply to the terminal gaps on the reference genome side. This is because the whole read needs to be aligned. Otherwise, it would mean that most probably the next (or previous) fragment of the genome sequence should be checked.

The regular semi-global alignment algorithm fills matrix H according to equation 1, where $i = 1, ..., n$, $j = 1, ..., m$, n, m are the lengths of sequences, $s_x(k)$ is the $k-th$ nucleotide of sequence x, SM is a substitution matrix, and g is a gap cost. The first row and the first column are in this case initialized according to equations 2 and 3.

$$H_{i,j} = \max \left\{ \begin{array}{c} H_{i,j-1} - g \\ H_{i-1,j} - g \\ H_{i-1,j-1} + SM(s_1(i), s_2(j)) \end{array} \right\} \tag{1}$$

$$H_{i,0} = 0 \tag{2}$$

$$H_{0,i} = 0 \tag{3}$$

The score of the final semi-global alignment is the maximum value from the last column or the last row of matrix H. It is worth noting that the model presented here uses linear gap cost function instead of affine gap penalty function. The latter is more popular in the context of proteins.

The modified version of semi-global alignment implemented in G-MAPSEQ computes matrix H in the same way, i.e. according to equation 1. However, the first row is initialized according to equation 4. Moreover, the final score is the maximum value from the last column of matrix H, while the last row is not considered.

$$H_{i,0} = -i * g \tag{4}$$

In order to speed up the filtration step, calculations are performed on graphics processing unit (GPU). G-MAPSEQ is able to utilize multiple GPUs installed in a single system. Moreover, a load balancer takes care of on-line work distribution among all available devices. More details of this implementation can be found in [6].

Once the alignment is computed, the similarity score between a read and a reference genome fragment is compared to a predefined threshold t, which is a parameter of the method. Threshold t can be expressed as a minimal alignment score or as a percentage of the maximum score between two identical sequences, each of length l.

It should be stressed that contrary to intuition the alignment itself is not calculated here – the result consists only of the score and the corresponding overlap value for individual pairs of sequences. This information is sufficient to determine whether a pair of sequences is similar enough to constitute a correct mapping position or not. Moreover, this way the method saves a lot of computational time. The whole alignment, i.e. with a backtracking step, is calculated afterwards, but only for those pairs of sequences that have already passed the quality verification. This is described in more detail later on.

The next problem comes from the way the reference genome is cut into fragments. At the beginning of the algorithm, the reference genome is fragmented according to the above-described parameters o and s. The lower value of o, the greater chance that one of the genome fragments will be identical with a given read. On the other hand, it causes many genome fragments to partially overlap with the read sequence, and hence, there is an increased chance that their k-mer characteristics will be adjacent to the k-mer characteristic of a given read on list L^S. As a result, in the set of promising pairs there are many pairs pointing to the same mapping position (often with very high scores because the length of genome fragments is usually 10-20% greater than the length of individual reads). Such duplicates need to be filtered. This idea is depicted in Figure 2.

The filtration procedure is slightly different for pair-end reads. In this case, each paired-end read is treated as an ordered pair of reads (they are ordered at the preprocessing stage). The mapping position is evaluated for every pair, not for individual reads. The mapping candidates must satisfy additional conditions, which are described by two parameters: *insert size* and the standard deviation of the insert size. The meaning of these parameters is as follows. The distance between mapping positions of the paired reads must be equal to the insert size with the permissible deviation from the reference value. In our algorithm the choice of mapping positions for paired-end reads is based on the sum of alignment scores of both reads which satisfy the distance requirements. If this condition can not be satisfied, the reads are marked as unpaired and are further processed as regular, non paired-end reads.

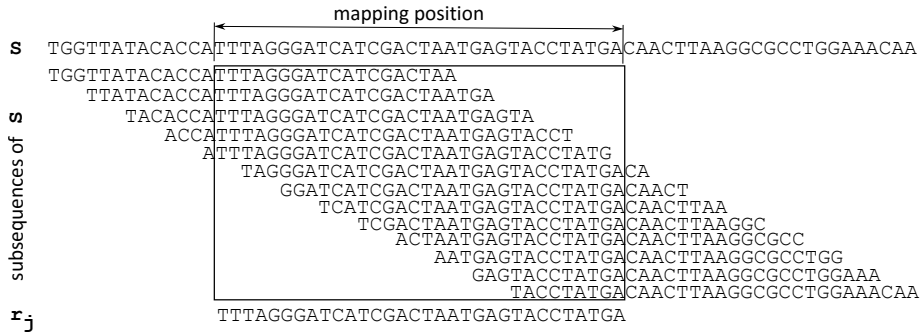

Figure 2. An example of filtration problem. Read r_j is a subsequence of the reference genome S. Multiple fragments cut from the genome with offset $o = 3$ overlap with a given read. At the filtration phase the algorithm selects the best scoring variant. Note that due to offset o and the length of fragments s there is no fragment identical to read r_j.

Calculation of the final alignments and results saving

The last step of the algorithm involves mainly saving the mapping positions to the file. The standard file format for such an analysis is SAM (or its binary version – BAM). This format contains many information about the mappings, some of them are optional. One of the optional information is the so-called CIGAR string, which is a condensed form of the alignment. However, in order to compute the CIGAR string the whole alignment with the backtracking step needs to be performed. For this task G-MAPSEQ uses a well-known implementation of the Needleman-Wunsch algorithm from the SeqAn [1] library. Although this is a quite time-consuming task, CIGAR string is very useful, especially for evaluation of mapping quality [8].

3 Results

In order to evaluate the quality of the presented method we performed several computational tests. The tests were designed in a way to examine different values of individual parameters. All the parameters were already described in Section 2, but for clarity are also listed below:

- s – length of the genome fragments,
- o – offset used to extract successive fragments of a reference genome,
- k – length of a k-mer,
- w – window/neighborhood size,

[1] SeqAn library: https://www.seqan.de/

- t – threshold – the percentage of the maximum possible score a promising pair needs to obtain to become a valid mapping position (depends also on the mapping mode, see Section 1, and the alignment scores of other pairs).

In order to perform reliable tests, unless stated otherwise, a real genomic data set was used (*Saccharomyces cerevisiae*). It consists of 17 chromosomes with total length of 12 million of base pairs. Because it would be very hard to evaluate the quality of method for reads coming from a biochemical experiment, an application for realistic reads extraction was used – *mason* [7]. The prepared test cases were finally assessed with a well-known Rabema software tool [8].

All tests were performed on the following hardware platform:

- CPU: Intel Core i7-3820, 3.6 GHz,
- GPU: 2 x NVIDIA GeForce GTX 680 with 2 GB of RAM,
- RAM: 64 GB,
- OS: CentOS release 7.1.1503.

The first part of tests concerns parameter tuning whereas the later part represents comparison of G-MAPSEQ with *bowtie* [12] – one the most popular mapping methods.

3.1 Window size

The value of window size parameter w defines how large is the neighborhood of each read. In other words, it determines the number of promising pairs verified by the GPU alignment algorithm for each read. The maximum window size is approximately equal to $(g - s)/o$, where g is the length of a genome. However, such a window would result in comparing every single read to all generated fragments and despite the high performance of the GPU alignment algorithm it would be very time-consuming.

The goal of the fist test was to estimate a reasonable value of window size. For this purpose, sorted lists of full k-mers characteristics were searched, and for each read its distance to the closest fragment (comprising it entirely) was computed. In other words, we computed the minimum window size that guarantees all reads to find the correct mapping position. It should be stressed that the number of fragments that contain a given read depends on: the difference between the lengths of a fragment and a read, and the offset o. The distance to the closest fragment was taken into account.

The tests were performed for 1 million of generated reads, each of length 100 nucleotides. Note that because the reads were extracted by *mason* toolkit, the correct mapping positions were known. The genome was cut into fragments of length $s = 120$ with offset $o = 20$. Therefore, the number of fragments comprising each read was 2 or 3 depending on the actual mapping position of a given read. The k-mer length was set to $k = 3, 4, 5, 6$.

The number of correctly aligned pairs in a given window vs. the window size is presented in Figure 3, whereas a closer look at the small windows is presented in Figure 4. If there are more fragments containing the entire read, chances of finding

one of the correct mappings increase. Results for the same experiment parameters but with the offset equal to 5 are shown in Figures 5 and 6.

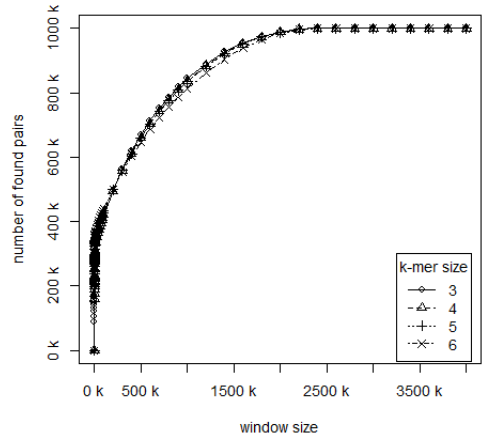

Figure 3. The number of correct pairs found depending on the window size for reads of length 100 bp, fragments – 120 bp and the offset value – 10.

From the performed test we may draw three main conclusions concerning the window size. First of all, the size of the k-mer has substantial influence on the quality of results. Longer k-mers are more informative and distinguishing and therefore more precise. The next conclusion is that there is no point in extending the window to an enormous size. The number of pairs found grows rapidly for a very small window size and then its further increase is rather small (see Figures 4 and 6). The last conclusion is that full k-mer characteristics are insufficient, because results which are comparable to other known methods would only be achieved for a window of a large size (see Figures 3 and 5). As a result, the window size for all further test was set to 2000.

3.2 K-mer size, offset and fragments lengths

For testing purposes, five data sets were generated, each with 10,000 paired-end reads of lengths $l = 36, 72, 100, 400, 800$ bp, respectively. Each read contained up to $3, 3, 5, 12, 23$ sequencing errors (substitutions, insertions or deletions). The number of reads considered here differs significantly from that of the real biological experiments, but the main goal of this test was to find out how changes in parameters affect the quality of mappings. The results for G-MAPSEQ are compared with *bowtie* and *bowtie2*. Tests were performed for paired-end as well as for single-end reads mapping. In the latter case each paired-end read was treated as two independent reads. It is worth noting that paired-end mappings are computed in the same way as single-end,

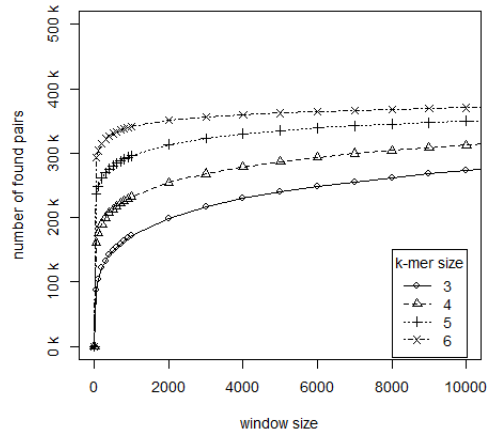

Figure 4. The number of correct pairs found depending on the window size – chart enlarged for small windows. Reads were of length 100 bp, fragments – 120 bp and the offset value – 10.

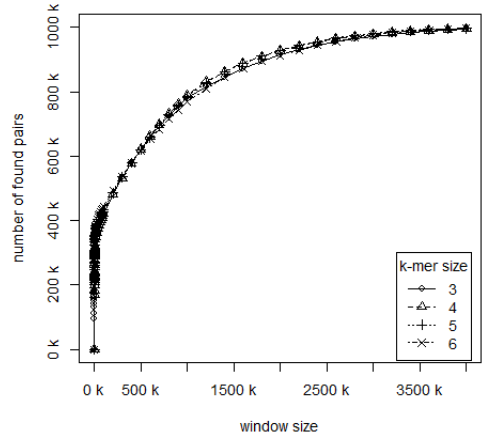

Figure 5. The number of correct pairs found depending on the window size for reads of length 100 bp, fragments – 120 bp and the offset value – 5.

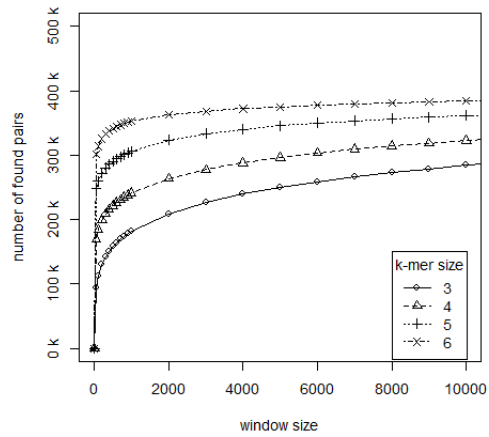

Figure 6. The number of correct pairs found depending on the window size – chart enlarged for small windows for reads of length 100 bp, fragments – 120 bp and the offset value – 5.

with an exception in the filtration procedure where additional restrictions are imposed. Therefore, some high-score mapping positions may be discarded, because the maximum distance between paired-end reads is exceeded. As a consequence, G-MAPSEQ cannot map more paired-end reads compared to their regular counterparts.

Table 1. Percentage of reads mapped to the *Saccharomyces cerevisiae* genome. All 10,000 reads were of length 36 bp. The length of partial subsequences was set to $p = 15$ and k refers to k-mer size. *PE* indicates that reads were mapped as paired-end.

k	o	G-MAPSEQ $s = 40$		bowtie		bowtie2	
			PE		PE		PE
3	2	89.04	87.63	92.29	84.18	94.37	96.28
	5	83.79	82.68				
	10	69.72	68.48				
4	2	91.30	89.72				
	5	87.77	86.34				
	10	77.88	76.17				
5	2	93.12	91.40				
	5	90.51	88.76				
	10	83.32	80.99				
6	2	93.90	91.81				
	5	91.66	89.61				
	10	85.94	82.98				

Table 2. Percentage of reads mapped to the *Saccharomyces cerevisiae* genome. All 10,000 reads were of length 72 bp. The length of partial subsequences was set to $p = 25$ and k refers to k-mer size. PE indicates that reads were mapped as paired-end.

k	o	G-MAPSEQ $s = 85$		bowtie		bowtie2	
			PE		PE		PE
3	2	92.85	89.77	87.71	81.37	97.1	97.22
	5	92.23	89.19				
	10	90.63	87.65				
	20	81.73	78.88				
4	2	94.40	91.02				
	5	93.89	90.57				
	10	92.45	89.24				
	20	85.76	82.52				
5	2	95.50	92.19				
	5	95.02	91.68				
	10	93.83	90.60				
	20	88.58	84.71				
6	2	96.41	92.67				
	5	96.08	92.27				
	10	95.38	91.64				
	20	90.79	86.54				

For all but one test set (i.e. with $l = 100$) constant fragment lengths were set. The tests were performed with different k-mer sizes (from 3 to 6) and different fragment offsets (adjusted to the read length). The quality of results for read lengths $l = 36, 72, 100, 400, 800$ bp are presented in Tables 1, 2, 3, 4 and 5, respectively.

The performed test shows that in general longer k-mers lead to better quality of mapping, which was expected. However, a significant improvement, i.e. from 4% up to 10%, which can be achieved for longer k-mers in the case of short reads (36 - 72 bp) cannot be observed for longer sequences (400 or 800 bp). This is a very meaningful observation as the computational time grows with increasing k-mer size. Therefore, the conclusion is that for longer reads small k-mers can be used with very little reduction of mapping quality.

The influence of the fragment length and the offset size was tested for reads of length 100 bp and is presented in Table 3. It is worth noting that for some combinations of parameters the method is unable to find correct mapping positions, because there is no fragment containing the entire read. For these parameters a significant decrease of the quality of mapping is observed. These combinations are: $(105, 20); (105, 50); (110; 50); (120; 50)$ (denoted as pairs (s, o)).

Table 3. Percentage of reads mapped to the *Saccharomyces cerevisiae* genome. All 10,000 reads were of length 100 bp. The length of partial subsequences was set to $p = 35$ and k refers to k-mer size. PE indicates that reads were mapped as paired-end.

k	o	G-MAPSEQ								bowtie		bowtie2	
		$s = 105$		$s = 110$		$s = 120$		$s = 150$					
			PE		PE		PE		PE		PE		PE
3	5	94.83	90.49	94.44	90.05	93.56	89.42	91.19	87.76	83.59	76.70	97.07	97.46
	10	92.98	87.35	93.73	89.25	92.50	88.51	90.15	86.88				
	20	86.53	80.68	87.74	83.53	88.47	84.97	86.71	83.36				
	50	47.89	46.55	54.09	52.57	63.63	61.81	74.09	71.52				
4	5	95.33	90.85	95.21	90.73	94.23	89.96	91.71	88.42				
	10	94.05	88.15	94.85	90.42	93.41	89.40	90.81	87.46				
	20	88.13	82.12	90.39	85.19	90.30	86.49	88.02	84.85				
	50	50.05	48.62	56.79	55.14	66.41	64.45	76.16	73.77				
5	5	96.25	91.68	96.15	91.50	95.26	90.83	92.56	88.94				
	10	95.12	88.89	95.95	91.31	94.88	90.47	91.75	88.23				
	20	90.47	83.69	92.42	86.41	92.81	88.26	89.50	86.09				
	50	51.70	50.03	58.94	56.97	70.14	67.72	79.15	76.30				
6	5	96.68	91.98	96.73	91.98	96.22	91.74	94.08	90.11				
	10	95.58	89.20	96.44	91.69	95.75	91.31	93.36	89.51				
	20	91.31	84.37	93.01	86.78	94.04	89.72	90.98	87.38				
	50	52.63	51.07	60.21	58.08	72.09	69.62	81.15	78.28				

The computation time was not taken into account during this part of tests. However, it grows significantly for longer reads which is caused by the final alignment procedure. This is because of the size of the dynamic programming matrix which increases quadratically with growing sequence lengths. An improvement implemented in G-MAPSEQ minimizes the number of cells that need to be computed by dynamic programming procedure. This was possible as the score of the alignment is calculated beforehand, during the procedure of promising pairs verification. This value is used to precisely limit the number of cells calculated in the final alignment. Even though it substantially speeds up the whole process, it still is quite time consuming.

Table 4. Percentage of reads mapped to the *Saccharomyces cerevisiae* genome. All 10,000 reads were of length 400 bp. The length of partial subsequences was set to $p = 100$ and k refers to k-mer size. PE indicates that reads were mapped as paired-end.

k	o	G-MAPSEQ $s = 420$		bowtie		bowtie2	
			PE		PE		PE
3	5	96.96	88.73	71.91	66.30	98.18	98.25
	20	97.01	88.65				
	50	94.34	84.91				
	100	84.42	76.85				
4	5	97.06	88.82				
	20	97.07	88.72				
	50	94.84	85.32				
	100	85.44	77.73				
5	5	96.98	88.66				
	20	97.27	88.83				
	50	95.18	85.26				
	100	86.72	78.33				
6	20	97.52	89.12				
	50	95.65	85.63				
	100	87.54	78.88				

3.3 Comparison to bowtie and bowtie2

A test was also performed for reads obtained from real biochemical experiment for *Caenorhabditis elegans* genome (accession number SRR065390). In order to evaluate the quality of mapping algorithms a special test case was prepared with the *rabema* software [8]. This software computes a special reference index with all possible mapping positions up to a defined number of errors (here 8). For this purpose *rasers3* [19] tool was used. Unfortunately, the preparation of such a reference index is very computationally heavy. Therefore, the input set originally containing over 60 millions of paired-end reads of length 2×100, was limited to 100,000 single-end reads. Addi-

Table 5. Percentage of reads mapped to the *Saccharomyces cerevisiae* genome. All 10,000 reads were of length 800 bp. The length of partial subsequences was set to $p = 280$ and k refers to k-mer size. PE indicates that reads were mapped as paired-end.

k	o	G-MAPSEQ $s = 850$		bowtie		bowtie2	
			PE		PE		PE
3	50	97.24	90.54	73.77	66.94	98.64	98.69
	100	94.71	87.80				
4	50	97.38	90.53				
	100	95.43	87.82				
5	50	97.42	90.64				
	100	95.53	87.63				
6	50	97.65	90.81				
	100	95.86	87.87				

tionally, the considered input data set contains a lot of low quality reads with many unrecognizable nucleotides (N's) which are not filtered. The preprocessing stage of this test case showed that from 100k reads, only 92k can be mapped with up to 8 errors.

The quality measure for this test was the number of reads mapped correctly. G-MAPSEQ was able to align 97.4% of reads (out of 92k), whereas the results for the other methods were 96.72%, 99.848% and 99.865% for bowtie, bowtie2 and bowtie2 with very-sensitive option enabled, respectively. We can conclude that the quality of G-MAPSEQ is slightly better as compared to bowtie, whereas bowtie2 outperforms both methods.

The computation time for G-MAPSEQ was less than 6 minutes. The same problem was solved by bowtie and bowtie2 within 50s and 8s, respectively. Unfortunately, G-MAPSEQ was significantly slower than the other methods. However, its computation time is still acceptable. Additional tests were performed to measure the processing time for G-MAPSEQ for 1M and 2M of reads. The results show that the computation time increases slower than linearly with the size of the input data.

To give the reader a better overview of the G-MAPSEQ performance, we measured the run time of each computational step of the algorithm. We tested the version with indexed genome, as it is more likely to be used in real-life scenarios. The results were collected for several test cases, but the main trends are the same regardless of the size of the input data. All the measurements were performed for k-mer length equal to 6. The preparation of k-mer characteristics takes around 25% of the entire computational time. Then 38% of the run time is consumed by the selection of promising pairs, including the alignment and filtration procedures. The greatest lexicographical index procedure takes around 12%. Calculation of the final alignment and results saving – around 23%. The rest of the time is spend on the remaining steps. It is worth noting that the genome index loading (including preparation of all necessary data structures) is quite quick (it takes less than 1% of the computational time) and independent of

the number of reads. Moreover, the smaller the offset size o, the more time is needed for genome index loading. Additionally, with small o the time needed for computation of the greatest lexicographical index grows (due to potentially much higher number of pairs to compare) but less than linearly.

The influence of extending the k-mer size on the computational time was also investigated. This parameter has a direct connection only with k-mer characteristics preparation stage. The greater k the more time is consumed. This can be estimated by the following formula: $t_k = k * t_{k-1}$, where t_k is the computational time for a given k.

4 Conclusions

The main goal of this work was to design and implement a new version of reads mapping algorithm which would be able to deal with noisy input data. We decided to use our ultra fast alignment algorithm designed for GPUs. However, it was clear that performing alignment of all reads with all possible mapping positions in a genome is not tractable. Even though there are some GPU-based tools that support similar scenarios [1], they were designed to deal with other problems, e.g. MSA [2]. Our brief estimation of the computational time of such an algorithm applied for real-life reads mapping scanario was that this would take a few years. Therefore, our method needed a rough, but fast indication where individual reads can be mapped. We have employed k-mer characteristics which represent information contained in the sequences but in a compressed form. With this information we were able to quickly compare sequences without even performing the actual alignment. Based on this, the algorithm computes a set of possible mapping positions which are then verified by the above-mentioned sequence alignment algorithm on GPU.

In order to evaluate the quality of the proposed solution we have conducted a number of tests. The results show that G-MAPSEQ produces reasonable mappings and is to some extent robust to sequencing errors. Unfortunately, it was outperformed by a leading tool available on the market – bowtie2. One of the current minor limitations of our method is that it can be used only for reads which are of similar size, i.e. for most of the currently available data sets. However, with small modifications G-MAPSEQ could be used for input data sets containing sequences with substantially different lengths, like in the case of emerging Pacific Bioscience technology. Importantly, we perceive it could be very useful for longer reads which tend to contain multiple reading errors. Such technologies are entering the market and are likely to become popular in the near future. Finally, we believe that the presented algorithm is the first working implementation of a completely new approach to the problem of reads mapping which may further stimulate development of algorithms in this area.

Acknowledgment

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.

System requirements

Software and hardware requirements are: Linux operating system, g++ compiler, make, CUDA 2.0 or higher, CUDA-compliant GPU, SeqAn library ver. 1.4.2 or newer, bison, flex.

References

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

[2] Blazewicz, J., Frohmberg, W., Kierzynka, M., Wojciechowski, P., G-MSA – A GPU-based, fast and accurate algorithm for multiple sequence alignment, *J. Parallel. Distr. Com.*, **73**, 1, 2013, 32–41.

[3] Ferragina, P., Manzini, G., Opportunistic Data Structures with Applications, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000.

[4] Fiannaca, A., La Rosa, M., Rizzo, R., Urso, A., A k-mer-based barcode DNA classification methodology based on spectral representation and a neural gas network, *Artificial intelligence in medicine*, **64**, 3, 2015, 173–184.

[5] Fonseca, N.A., Rung, J., Brazma, A., Marioni, J.C., Tools for mapping high-throughput sequencing data, *Bioinformatics*, **28**, 24, 2012, 3169–3177.

[6] Frohmberg, W., Kierzynka, M., Blazewicz, J., Gawron, P., Wojciechowski, P., G-DNA – a highly efficient multi-GPU/MPI tool for aligning nucleotide reads, *Bulletin of the Polish Academy of Sciences Technical Sciences*, **61**, 4, 2013, 989–992.

[7] Holtgrewe, M., Mason – a read simulator for second generation sequencing data, *Technical Report Institut für Mathematik und Informatik, Freie Universität Berlin*, TR-B-10-06, 2010.

[8] Holtgrewe, M., Emde, A.-K., Weese, D., Reinert, K., A Novel And Well-Defined Benchmarking Method For Second Generation Read Mapping *BMC Bioinformatics*, **12**, 210, 2011.

[9] Kierzynka, M., GPU-accelerated graph construction for the whole genome assembly, *Phd. thesis, Poznan University of Technology, Poznan, Poland*, 2014.

[10] Kuksa, P., Pavlovic, V., Efficient alignment-free DNA barcode analytics, *BMC bioinformatics*, **10**, 14, 2009, 1–18.

[11] Langmead, B., Salzberg, S.L., Fast gapped-read alignment with Bowtie 2, *Nat Methods*, **9**, 4, 2013, 357–359.

[12] Langmead, B., Trapnell, C., Pop, M., Salzberg, S.L., Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, *Genome Biology*, **10**, 3, 2009, 1–10.

[13] Liu, Y., Schröder, J., Schmidt, B., Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data, *Bioinformatics*, **29**, 3, 2013, 308–315.

[14] Needleman, S.B., Wunsch, C.D., A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J Mol Biol*, **48**, 3, 1970, 443–453.

[15] Polychronopoulos, D., Weitschek, E., Dimitrieva, S., Bucher, P., Felici, G., Almirantis, Y., Classification of selectively constrained dna elements using feature vectors and rule-based classifiers, *Genomics*, **104**, 2, 2014, 79–86.

[16] Reinert, G., Chew, D., Sun, F., and Waterman, M.S., Alignment-free sequence comparison (I): statistics and power, *Journal of Computational Biology*, **16**, 12, 2009, 1615–1634.

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

[18] Wan, L., Reinert, G., Sun, F., Waterman, M.S., Alignment-free sequence comparison (II): theoretical power of comparison statistics, *Journal of Computational Biology*, **17**, 11, 2010, 1467–1490.

[19] Weese, D., Emde, A-K., Rausch, T., Döring, A., Reinert, K., RazerS – fast read mapping with sensitivity control *Genome Research*, **19**, 2009, 1646–1654.

[20] Weese, D., Holtgrewe, M., Reinert, K., RazerS 3: faster, fully sensitive read mapping, *Bioinformatics*, **28**, 20, 2012, 2592–2599.

[21] Weitschek, E., Cunial, F., Felici, G., LAF: Logic Alignment Free and its application to bacterial genomes classification, *BioData mining*, **8**, 1, 2015.

[22] Weitschek, E., Santoni, D., Fiscon, G., De Cola, M.C., Bertolazzi, P., Felici, G., Next generation sequencing reads comparison with an alignment-free distance, *BMC research notes*, **7**, 1, 2014, 1–13.

Received 11.01.2016, Accepted 12.05.2016