Efficient alignment of pyrosequencing reads for re-sequencing applications

Francisco Fernandes, Paulo GS da Fonseca, Luis MS Russo, Arlindo L Oliveira and Ana T Freitas

Abstract

Background: Over the past few years, new massively parallel DNA sequencing technologies have emerged. These platforms generate massive amounts of data per run, greatly reducing the cost of DNA sequencing. However, these techniques also raise important computational difficulties mostly due to the huge volume of data produced, but also because of some of their specific characteristics such as read length and sequencing errors. Among the most critical problems is that of efficiently and accurately mapping reads to a reference genome in the context of re-sequencing projects.

Results: We present an efficient method for the local alignment of pyrosequencing reads produced by the GS FLX (454) system against a reference sequence. Our approach explores the characteristics of the data in these re-sequencing applications and uses state of the art indexing techniques combined with a flexible seed-based approach, leading to a fast and accurate algorithm which needs very little user parameterization. An evaluation performed using real and simulated data shows that our proposed method outperforms a number of mainstream tools on the quantity and quality of successful alignments, as well as on the execution time.

Conclusions: The proposed methodology was implemented in a software tool called TAPyR–Tool for the Alignment of Pyrosequencing Reads–which is publicly available from http://www.tapyr.net.

Background

Sequencing by capillary electrophoresis, known as the Sanger method [1], has been employed in many historically significant large-scale sequencing projects and is regarded as the gold standard in terms of both read length and sequencing accuracy [2]. Several Massively Parallel DNA Sequencing (MPDS) technologies have recently emerged, including the Roche/454 GS FLX System, the Illumina/Solexa Genome Analyser, and the AB SOLiD System, which are able to generate a few orders of magnitude more bases per instrument run, being considerably less expensive than the Sanger method [2,3]. These technologies are enabling researchers and practitioners to efficiently sequence genomes, leading to very significant advances in biology and medicine. However, the huge volume of data produced by MPDS technologies creates important computational challenges [4].

Moreover, the different platform-specific data characteristics require different algorithmic approaches. For instance, some applications may use the 454 Titanium platform to produce reads 400 bases long, some other studies may employ a SOLiD system set to produce short reads of 35 bases, and yet other projects may use the Illumina system to produce 2 × 75 bases paired-end reads. Given their large variety, it would be rather difficult for a single algorithm to handle all kinds of data optimally.

When sequencing a new organism, one is usually faced with the problem of assembling the sequence fragments (reads) together from scratch. However, when a sufficiently close sequence is already known, one may choose to use it as a reference and proceed by first mapping the reads to this reference and then determining the new sequence by extracting the consensus from the mapping results. The former strategy is called de novo sequencing, while the latter is known as re-sequencing.

Several tools have recently been developed for generating assemblies from short reads, e.g [5,6]. Similarly, several methods have been proposed to address the...
problem of efficiently mapping MPDS reads to a reference sequence, like [7-12], to cite a few. As referred before, the sheer volume of data generated by MPDS technologies (to the order of hundreds of gigabases per run), and the need to align reads to large reference genomes limit the applicability of standard techniques. Indeed, in a typical application, we may have to align hundreds of millions of reads to a reference genome that can be as large as few gigabases, a job that cannot be efficiently achieved through standard dynamic programming procedures.

One way to speed up the read alignment task is to resort to approximate indexing techniques. A first generation of aligners was based on hash tables of k-mers. Some of them, like SSAHA2 [13], build tables of k-mers of the target sequence, whilst others, like Newbler [14], index the reads, thus presumably requiring re-indexing for each new run. Recent developments in the field of compressed approximate indexes have led to a new family of alignment algorithms such as Segemehl [10], which uses an enhanced suffix array (see Implementation), and BWA-SW [11], which uses a FM-index (see Implementation) to accelerate Smith-Waterman alignments. Yet the number of aligners that support GS FLX pyrosequencing data is, as of today, relatively scarce compared to other technologies, most notably Illumina. Moreover, some of these tools find their origins in the days before the advent of the new sequencing technologies and only later were adapted to cope with new kinds of data [13], and some others target multiple kinds of data [10] being not optimized for pyrosequencing data. Given this state of affairs, we argue that there is still room for improvement in the realm of publicly available aligners specifically designed for high-throughput pyrosequencing data.

In this paper we present a new method for the alignment of pyrosequencing reads, like those produced by the 454 GS FLX platform. By focusing on this specific technology, our procedure manages to explore its data characteristics to achieve improved performance over other mainstream methods. Like many of those methods, ours also builds an index of the target (reference) sequence to accelerate the alignment. It then employs a multiple seed heuristic to anchor the best candidate alignments. Contrary to other seed-based alignment tools, our strategy adds more flexibility by dispensing with the need of determining the number and length of the seeds beforehand. Our heuristic relies on some assumptions that can be reasonably expected to hold true for re-sequencing projects based on pyrosequencing data, namely, that the optimal alignments are mostly composed of relatively large chunks of exact matches interspersed by small, possibly gapped, divergent regions. A banded dynamic programming is used to finish up the candidate multiple seed alignments considering user-specified error constraints. A detailed description of the algorithm and data structures is given in the “Implementation” section. In the “Results and Discussion” section, we present a comparison between our method and a set of tools of widespread use for the local alignment of pyrosequencing reads. We base our discussion on results obtained with both real and simulated data.

Implementation

Compressed indexes

The main data structure for sequence pattern matching is an index. Indexes reduce the time for matching a pattern because they restrict the search to the positions where it may occur instead of scanning the whole text. One of the most, if not the most, popular index structures is the suffix tree (ST), which is obtained by identifying common prefixes of the different suffixes of the represented text to nodes of a tree (see Figure 1(a) for an example). In such a structure, a pattern can be searched following edges with matching labels down from the root. Each leaf of the suffix tree represents a suffix of the text and, more generally, each node represents the subset of suffixes corresponding to the leaves of the subtree rooted at that node. The downside of indexes is that they need to be constructed a priori and have a bad reputation of using too much space. Despite providing for fast searching algorithms, suffix trees are particularly known for this bad characteristic. A popular alternative to suffix trees are suffix arrays (SA), that require asymptotically the same space, $O(n)$ computer words for a text of size $n$, but with a smaller proportionality factor. Suffix arrays are obtained by ordering the suffixes of a text lexicographically. A correspondence can be established between nodes of the suffix tree and contiguous intervals of the suffix array. An example of a suffix array is shown in Figure 1(b). Detailed descriptions of string matching and indexes, including the ones mentioned here, are widely available [15].

Recent research on indexes has focused on the fact that pointer representations require $O(n \log n)$ bits whereas the original text (the target genome, in our case) requires only $n \log \sigma$ bits, where $\sigma$ is the alphabet size, e.g. 4 for DNA and 20 for proteins. In an effort to reduce this gap, new indexes have been designed which became collectively known as compressed indexes [16] due to the fact that they rely heavily on data compression techniques. In spite of their reduced space, compressed indexes can be made to allow for an even broader range of operations than classical indexes, like generalized branching, that combines blocks of letters instead of just one letter at a time [17]. Our method uses an implementation of the FMIIndex [18] optimized
for the DNA alphabet. The FMIndex is a compressed index based on the Burrows-Wheeler transform (BWT) [19] requiring only \( O(n \log \sigma) \) bits of memory space. The BWT of a text \( t \) is obtained by appending an extra symbol \( $ \) to \( t \), and then sorting all cyclic permutations (rotations) of \( t \$ \) according to the lexicographic order, with \( $ \) being the lowest symbol. Thus a BWT of a text is essentially equivalent to its suffix array. In fact, they are related through the formula \( \text{BWT}[i] = t[\text{SA}[i] - 1] \), for \( i = 1, \ldots, |t| \). An example of a BWT is given in Figure 1(c).

The seed-based search approach
Due to the relatively large size of the GS FLX reads, it is not practical to use a plain index-based exact matching algorithm. Some sort of backtracking strategy could be used to allow for errors but, since the number of possible comparisons increases exponentially with their number, this becomes rapidly inefficient. Instead, we propose a seed-based search heuristic that explores the characteristics of the pyrosequencing data and of the bona fide alignments that are likely to arise in the context of resequencing applications. Since the error rates are usually low [20], and the prevalent type of pyrosequencing errors are small indels, with mismatches being much less common, we conjecture that the optimal alignments are expected to be formed of large chunks of exact matches interspersed by divergent gapped regions.

Moreover, since the read lengths are of a few hundred bases, we can expect the exact match regions to be large enough so we can use segments of them, called seeds, as a backbone to pin down the position of the alignment on the reference sequence, or at least reduce the amount of candidate positions to a manageable number of possibilities that can be tested individually. In this case, the optimal alignments can be obtained by expanding the candidate multiple seed matches into alignments of the whole read by filling up the remaining regions and selecting those with best overall scores.

Our strategy for choosing the seeds consists in approximately partitioning the read into maximal exact match blocks in a greedy fashion. More precisely, let \( r = r_1 \ldots r_m \) be the read. The procedure starts at the first position of the read and uses the index to find the largest prefix of the read with exact occurrences in the reference sequence, say \( r[1 \ldots l] = r_1 \ldots r_l \). In practice we obtain the equivalent of an interval of the BWT which contains the positions of the reference sequence \( g \) at which \( r[1 \ldots l] \) occurs. Obviously, by maximality, \( r[1 \ldots l + 1] \) does not occur in \( g \). This happens because none of the occurrences of \( r[1 \ldots l] \) is followed by \( r_{l+1} \) or, put another way, because there is a mismatch between \( r_{l+1} \) and the letter following each occurrence of \( r[1 \ldots l] \) (or because it occurs at the very end of \( g \)). If \( r_{l+1} \neq r_l \), then we set \( r[1 \ldots l] \) as the first seed and proceed as above to find the next seed starting from position \( l + 2 \). If,
however, we have \( r_{i+1} = r_p \) this means that the difference occurred in the middle of a homopolymer (contiguous subsequence of identical bases), most likely due to an insertion sequencing error. In this case, we set \( r[1 ... l] \) as the first seed as before, but advance the cursor to the start of the next homopolymer, i.e. to the smallest \( i' > l \) s.t. \( r_{i'} = r_l \). We repeat this process until the end of the read is reached.

Once we have the set of seeds and their individual positions in the reference sequence, we need to identify subsets of occurrences of distinct seeds that are in accordance with their original order and spacing in the read, which can then serve as a support for the final alignments. More formally, let \( g \) be the reference sequence, and \( r \) be a read. Let also \( s_1, \ldots, s_k \) be \( k \) substrings of \( r \) such that \( r = s_1 a_1 s_2 \ldots a_k s_k \) where, for \( i = 1, \ldots, k \), \( s_i \) denote the seeds and \( a_i \) denote the substrings in between them. For each \( i = 1, \ldots, k \), let \( o_i \geq 0 \) be the number of exact occurrences of \( s_i \) in \( g \). We then have \( o = \prod_{i=1}^{k} o_i \) ways to choose a set containing one occurrence for each of the \( s_i \) substrings. Let \( p = (p_1, \ldots, p_k) \) be one of such \( o \) tuples of distinct seed occurrences. If, for \( i = 1, \ldots, k-1 \), we have \( p_i \leq p_{i+1} \) and \( |a_i| \geq |a_{i+1}| \) for some given \( e_i \geq 0 \), then we say that these occurrences are coherent. Hence, a coherent set of seed occurrences is composed of positions which respect the relative order of the corresponding seeds in the read and such that, for any two consecutive seeds, the distance between their occurrences lies within a restricted interval around the actual distance between those seeds in the read. For the sake of flexibility, we do not restrict ourselves to coherent sets containing occurrences of all the seeds. We also take partial sets containing occurrences of only some of those seeds as good candidates for further expansion.

The set of seeds \( (s_1, \ldots, s_k) \) and their occurrence positions in \( g \) can be obtained with the index in linear time. However, the number of combinations of occurrences of different seeds can be rather large, especially if some of them are short. This makes it impractical to test all possibilities for coherence. Nonetheless, most of these combinations will typically be non-coherent, and if we care to previously sort the set of occurrences of each seed, we can efficiently search for coherent combinations using, again, a greedy strategy, simply by scanning the seed matches in \( g \) from left to right, partitioning them into maximal non-overlapping sets of consecutive coherent matches. Although this might seem a rough approach at first glance, in fact this strategy has shown to be adequate because of the relatively large size of the seeds and small separation between them, which makes it difficult for occurrences of two consecutive seeds to be interspersed with an occurrence of a third one.

Once the potential read occurrences indicated by coherent multiple-seed matches are found, the algorithm runs a banded Needleman-Wunsch dynamic programming procedure with Gotoh’s modifications to align the non-seed segments of the read to their counterparts in the genome. That is, if we have a coherent set of occurrences of the seeds \( s_{j_1}, \ldots, s_{j_q} \), s.t. the read decomposes into \( r = b_0 s_{j_1} b_{j_2} s_{j_2} b_{j_q} s_{j_q} b_{q'} \) and the reference sequence into \( g = c_0 s_{j_1} c_{j_2} s_{j_2} c_{q} s_{j_q} c_{q'} \) then we align each pair \( (b_i, c_i) \), for \( i = 0, \ldots, q \). Of course, for both ends, \( (b_0, c_0) \) and \( (b_{q'}, c_{q'}) \), we perform semi-global alignments. The largest candidate coherent multiple seed matches are extended this way and accepted as a read occurrence if either the overall alignment identity stays above a given threshold percentage \( t \) or, alternatively, if the sum of the errors in-between the seeds and at the extremes of the read do not exceed a pre-established number \( e \). The algorithm can be chosen to report all the accepted occurrences or only the one(s) with the least errors. The strategy described above is illustrated in Figure 2.

**Synthetic data generation**

In order to evaluate the algorithms in a controlled setting, we generated artificial data sets with a procedure inspired by empirical studies on GS FLX data [20,21], and designed to yield reads with characteristics similar to real data. In our procedure, \( n \) random contiguous subsequences are extracted from a given ‘source’ sequence \( g \). The lengths of these initial subsequences are drawn from a normal distribution with mean \( \mu \) and standard deviation \( \sigma \), also provided as input. Next, these subsequences are modified to simulate sequencing errors as follows. In the GS FLX high-throughput pyrosequencing procedure, the template molecules are sequenced one maximal homopolymer at a time (formally, a homopolymer can consist of a single base), as opposed to one base at a time in the traditional Sanger method. Hence, the most common type of error in pyrosequencing consist in the misinterpretation of the intensity of the signal that determines the length of the homopolymer being read, leading to an insertion or deletion of identical consecutive bases in the read, relative to the actual template sequence. Miscalled base errors (substitutions) also occur but they are comparatively much less frequent. Sequence quality is known to be non-uniform along the read, being lower at the extremes, particularly towards the 3’ end. Also, errors tend to affect long homopolymers more than short ones. However, for the sake of simplification, we consider that errors are uniformly distributed along the read and that the prevalence and size of indels are not affected by the length of the homopolymers. More
Figure 2 Sketch and example of the TAPyR procedure. (a) Sketch of the seed strategy employed by TAPyR. In this schema, three seeds are chosen, and seven matches of these seeds are found on the reference genome, three for the first seed, two for the second, and two for the third. These occurrences are ordered in the genome and scanned from left to right. Multiple seed matches are formed by extending current partial matches with the next occurrence if the coherence criteria are met. Otherwise, the current multiple match is stored as a potential candidate and a new one is started. In this example, we finish with five potential candidates for extension indicated by the dashed boxes. The largest candidate(s), i.e. the multiple seed occurrence that span most bases, are chosen for extension. In this case, that should be \((u_2^1, u_3^1)\) . (b) A more concrete example, in which we have a sequence of length 15, which was originally read from position 101 of the genome, with one insertion at position 5 and one substitution at position 9. The algorithm starts searching from the beginning of the read in the index, but cannot continue beyond the fourth character. At this point, we have the first seed \(s_1 = aaaa\), which occurs at position 101 in the genome. The next character of the read is skipped, and the search continues from position 6, which is the beginning of the second seed. Seed \(s_2 = cccc\) happens to have an accidental occurrence at the position 201, which is not related to the actual read position in the genome. Again, we skip the next (mismatched) character of the read and restart at position 11. This time the search reaches the end of the read, and yields the last seed \(s_3 = gggg\), occurring at position 110. These three occurrences are now sorted according to their position in the genome, and it turns out that the occurrences of \(s_1\) and \(s_3\) form a coherent multiple seed occurrence of combined length 9. The other candidate would be composed of the occurrence \(s_2\) alone which is not chosen for expansion since it is smaller. The space between the two seeds is then filled using dynamic programming, and the correct mapped position (101) is returned along with the final alignment.
precisely, the procedure takes in three parameters $p_{\text{sub}}$, $p_{\text{ins}}$ and $p_{\text{del}}$ which correspond to the probabilities of having a substitution, an insertion or a deletion in any given homopolymer, regardless of its length and position in the read. Moreover, these events are considered to be mutually exclusive, that is, we assume that for any homopolymer being sequenced, there can either be a substitution error with probability $p_{\text{sub}}$, an insertion with probability $p_{\text{ins}}$ a deletion with probability $p_{\text{del}}$ or it can be correctly sequenced with probability $1 - (p_{\text{sub}} + p_{\text{ins}} + p_{\text{del}})$. Whenever a mismatch takes place, the mis-called base is randomly chosen according to substitution probabilities indicated in a matrix $m$, given as input. Each row/column of the element $m[a, b]$ indicates the probability for $a$ to be mis-called as (replaced by) $b$. As for indels, the lengths of the gaps are drawn from Zipfian distributions, which are discrete power-law distributions with mass function $f(k; \gamma, \omega) = k^{-\gamma} / \sum_j k^{-\gamma} \propto k^{-\gamma}$, for $k = 1, \ldots, \omega$. In our case, $\omega$ is a positive integer parameter that corresponds to a maximum allowed gap size, and $\gamma > 0$ controls the shape of the distribution: the greater its value the higher the prevalence of small gaps. We use specific exponent parameters, $\gamma_{\text{ins}}$ and $\gamma_{\text{del}}$, for insertion and deletion operations, respectively.

**Results and Discussion**

We evaluated TAPyR against other mainstream mapping tools which are also able to deal with high-throughput pyrosequencing reads, namely BWA-SW [11], SSAHA2 [13], Segemehl [10], GASSST [12], and Newbler [14]. Our analyses were performed with real and simulated data sets, with the objective of assessing the efficiency and accuracy of the aforementioned tools in the context of re-sequencing projects.

**Results on real data**

The biological data sets we used, summarized in Table 1, encompass a reasonable variety of organism types, including two bacteria (Streptococcus pneumoniae and Escherichia coli), one protozoan (Plasmodium falciparum), one nematode (Caenorhabditis elegans), one insect (Drosophila pseudoobscura), and one human chromosome. They also cover re-sequencing applications with reads from individuals of the same species (human), different and mutated strains of the same species (bacteria and worm), and different (sub-)species (fly).

In this experiment, we wanted to analyze the ability of the algorithms to produce high coverage mappings, which directly relates to the proportion of reads that can be successfully mapped. High coverage is essential to the successful completion of a re-sequencing project, with about 20-25× coverage being required for optimal results with the GS FLX technology [22]. Attaining such high levels depends naturally on the amount of available data, but equally on the capacity of the alignment tool to map the reads correctly, especially in the presence of

| Table 1 Real data sets |
|------------------------|
| Reference genome Source | Reference genome size | SRA accession and species | Total reads | Average read length |
|-------------------------|-----------------------|---------------------------|-------------|-------------------|
| S. pneumoniae ATCC 70069 [GenBank: FM211187] | ≈2.2 Mbp | SRR001327 S. pneumoniae CDC1873-00 | 646,724 | 253 |
| E. coli 0127:H6 E2348/69 [GenBank: FM180568.1] | ≈4.96 Mbp | SRR000868 E. coli K-12 | 588,397 | 263 |
| P. falciparum 3D7 PlasmoDB rel 7.0 | ≈23.3 Mbp | SRR006911 P. falciparum 3D7 | 203,196 | 223 |
| C. elegans WormDB rel. WS210 | ≈103 Mbp | SRR022943 C. elegans Lynch MA41 mutation-accumulation line derived from N2 | 3,214,353 | 103 |
| D. pseudoobscura FlyBase rel. 2.14 | ≈150 Mbp | SRR003807 D. pseudoobscura Flagstaff 1993 | 834,659 | 239 |
| H. sapiens Chr. 15 ENSEMBL ver. GRCh37 | ≈100 Mbp | SRR001420 Human individual NA15510 | 3,204 | 212 |

Biological data sets used for the evaluation of the algorithms. The read data sets were downloaded from the Sequence Read Archive (SRA) public repository.
inevitable sequencing errors and natural variation. The other aspect we wanted to assess was the efficiency of the algorithms in terms of computation time. Efficiency is a critical aspect for any algorithm in modern high-throughput data processing pipelines, given the rapid increase in the volume of data being produced.

The results of our tests are shown in Table 2. In that comparison, we included two lines corresponding to TAPyR being set to report alignments with at least 50% (TAPyR 50) and 85% (TAPyR 85) identity, designated by TAPyR 50 and TAPyR 85, respectively. The other tools were run with their default options for 454 data, except for the following modifications. SSAHA2 and Segemehl were set to report only the best alignment for each read. For GASSST, we set the minimum identity to 85% (option -p 85) to match Segemehl. Newbler was set not to generate large files (option -nobig) and to load the index into the main memory (option -m). Reported times refer to the total number of CPU-seconds that the process used directly, as given by the Linux command time -f "%U\%.%U\%.%U\%%.%U\%.%U\%%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%.%U\%...
The results of the tests are shown in Table 3. We notice that all algorithms give quite accurate results in all the tested conditions. In any case, TAPyR behaved among the best in terms of accuracy, mapping virtually all the tested conditions. Moreover, as in the previous experiments, our method has confirmed to be fastest by a comfortable margin.

**Memory usage**

We also measured the memory requirements of the evaluated tools in the tests with real data discussed above. The figures presented in Table 4 show the sizes of the index files on disk, when they exist, and the peak usage of main memory for the different data sets. As it can be seen, BWA displayed the smallest requirements in absolute terms, followed closely by TAPyR. The other tools, especially those based on k-mer tables, demand substantially more memory. As expected, TAPyR’s index files scale linearly with the size of the indexed genomes (by a multiplicative factor of ≈ 1.6). Apart from the index, which is loaded into main memory, TAPyR uses only a small additional amount of space (mainly for the dynamic programming part), so that the total amount of required RAM also scales linearly with the indexed genome (by a factor of ≈ 2.5). These modest and predictable requirements make TAPyR suitable for large genomes with moderately-sized machines.

**Conclusions**

The combination of state of the art indexing techniques and a seed-based search approach led to the development of a new read mapping method for high-throughput pyrosequencing data. By using an effective heuristic which explores the characteristics of this particular kind of data in the context of typical re-sequencing applications, our method manages to achieve convincing performance in terms of speed and in terms of the number and precision of aligned reads, as demonstrated by our tests with real and simulated data. In fact, our proposed solution has displayed class-leading CPU-time performance and excellent use of input reads in comparison to other mainstream tools. An added-value of our procedure comes from the fact that it requires almost no external parameterization. As a matter of fact, the main user options are end-of-the-chain cutoff parameters that concern the quality of the reported alignments in terms of minimal identity or maximal number of errors, having no consequence on the accuracy of the heuristic and only marginal impact on the overall execution time. Memory requirements are also on par with the best in this category of tools, being not only small in absolute terms but, more importantly, linearly proportional to the size of the input reference sequence by a small factor. Based on these results, we propose that TAPyR constitutes an advantageous alternative for re-sequencing projects based on pyrosequencing data.

**Availability and requirements**

Project name: TAPyR—Tool for the Alignment of Pyrosequencing Reads

- Project home page: http://www.tapyr.net
- Operating system(s): multiple (requires a C compiler only)
- Programming language: C
- Other requirements: none (see Results section for an idea of memory usage)
- License: GNU GPL
- Restrictions to use by non-academics: none additional
Acknowledgements
We thank Ana Tereza Vasconcelos and her staff from LNCC, Brazil, for sharing their expertise with the GS FLX data and for granting us access to their platform.

Funding
This work was supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds, by the FCT PhD grant SFRH/BD/45586/2008 (FF), and by the PTDC program (project PTDC/EBI-EIA/112283/2009).

Author details
1 Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento (INESC-ID), R. Alves Redol 9, 1000-029 Lisboa, Portugal.
2 Instituto Superior Técnico-Universidade Técnica de Lisboa (IST/UTL), Av. Rovisco Pais, 1049-001 Lisboa, Portugal.

Authors’ contributions
AF and AO conceived the project. All authors have participated in the design and refinement of the method and in the analysis of the results. FF wrote most of the code, helped by LR and PF. FF and PF collected the data and performed the experiments. The authors collectively drafted the manuscript. PF, AF, and FF wrote most of the text. All authors revised and approved the final version.

Received: 16 December 2010 Accepted: 16 May 2011

References
1. Sanger F, Nicklen S, Coulson AR: DNA sequencing with chain-terminating inhibitors. 1977. Biotechnology 1992, 24:104-8.
2. Bonetta L: Genome sequencing in the fast lane. Nat Methods 2006, 3:141-147.
3. Schuster SC: Next-generation sequencing transforms today’s biology. Nat Methods 2008, 5:16-8.
4. Pop M, Salzberg SL: Bioinformatics challenges of new sequencing technology. Trends Genet 2008, 24(3):142-9.
5. Chaisson MJ, Pevzner PA: Short read fragment assembly of bacterial genomes. Genome Res 2008, 18(2):324-30.
6. Zerbino DR, Birney E: Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 2008, 18(5):821-9.
7. Langmead B, Trapnell C, Pop M, Salzberg SL: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 2009, 10(3):R25.
8. Li H, Durbin R: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 2009, 25(14):1754-60.
9. Li R, Yu C, Li Y, Lam TYW, Yu SM, Kristiansen K, Wang J: SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics 2009, 25(15):1966-7.
10. Hoffmann S, Otto C, Kurtz S, Sharma CM, Khaitovich P, Vogel J, Stadler PF, Huckermüller J: Fast mapping of short sequences with mismatches, insertions and deletions using index structures. PLoS Comput Biol 2009, 5(9):e1000502.
11. Li H, Durbin R: Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics 2010, 26(5):589-95.
12. Rizz G, Lavenier D: GASSST: global alignment short sequence search tool. Bioinformatics 2010, 26(20):2534-40.
13. Ning Z, Cox AJ, Mullikin JC: SSaha: a fast search method for large DNA databases. Genome Res 2001, 11(10):1725-9.
14. Droese M, Hill B: The Genome Sequencer FLX System-longer reads, more applications, straight forward bioinformatics and more complete data sets. J Biotechnol 2008, 136(1-2):23-10.
15. Gusfield D: Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, NY, 1997.
16. Navarro G, Mäkinen V: Compressed full-text indexes. ACM Comp Surv 2007, 39(2):1-69.
17. Russo L, Navarro G, Oliveira A: Dynamic Fully-Compressed Suffix Trees. Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008). LNCS 2008, 191-203.
18. Ferragina P, Manzini G, Mäkinen V, Navarro G: Compressed representations of sequences and full-text indexes. Transactions on Algorithms (TALG 2007, 3(2).
19. Burrows M, Wheeler DJ: A Block-Sorting Lossless Data Compression Algorithm. Technical report 124, Palo Alto, CA, Digital Equipment Corporation 1994 [http://citeseer.ist.psu.edu/76182].
20. Huse SM, Huber JA, Morrison HG, Sogin ML, Welch DM: Accuracy and quality of massively parallel DNA pyrosequencing. Genome Biol 2007, 8(7):R143.
21. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen YJ, Chen Z, Dewell SB, Du L, Fierro JM, Gomes XV, Godwin BC, He W, Helgesen S, Ho CH, Ho CH, Iryzk GP, Jando SC, Alenquer MU, Jarve TP, Jarige KB, Kim JB, Knight JR, Lanza JR, Leamon JH, Lefkowitz SM, Lei M, Li J, Lehman KL, Lu H, Makhijani VB, McDade KE, McKenna MP, Myers EW, Nickerson E, Noble JR, Plant R, Puc BP, Ronan MT, Roth GT, Sarkis GJ, Simons JF, Simpson JW, Srinivasan M, Tartar KR, Tomasz A, Vogt KA, Volkmer GA, Wang SH, Wang Y, Weiner MP, Yu P, Begley PF, Rothberg JM: Genome sequencing in microfabricated high-density picolitre reactors. Nature 2005, 437(7057):37-80.
22. Aury JM, Crasquin C, Barbe V, Roger O, Mangenot S, Samson G, Poulain J, Anthouard V, Scarpelli C, Artiguenave F, Wincker PF: High quality draft sequences for prokaryotic genomes using a mix of new sequencing technologies. BMC Genomics 2008, 9:603.