Algorithms for mapping short degenerate and weighted sequences to a reference genome

Pavlos Antoniou

University of Cyprus,
Department of Computer Science,
Nicosia, Cyprus
E-mail: panton@cs.ucy.ac.cy

Costas S. Iliopoulos

King's College London,
Department of Computer Science,
London, UK

Curtin University,
Digital Ecosystems and Business Intelligence Institute,
Perth, Australia
E-mail: csi@dcs.kcl.ac.uk

Laurent Mouchard

University of Rouen,
LITIS (EA 4108),
System and Information Processing,
76821 Mont Saint Aignan Cedex, France

King's College London,
Department of Computer Science,
London, UK
E-mail: Laurent.Mouchard@univ-rouen.fr

Solon P. Pissis*

King's College London,
Department of Computer Science,
London, UK
E-mail: solon.pissis@kcl.ac.uk
*Corresponding author

Abstract: Novel high-throughput (Deep) sequencing technologies have redefined the way genome sequencing is performed. They are able to produce millions of short sequences in a single experiment and with a much lower cost than previous methods. In this paper, we address the problem of efficiently mapping and classifying millions of short sequences to a reference genome, based on whether they occur exactly once in the genome or not, and by taking into consideration probability scores.

In particular, we design algorithms for *Massive Exact and Approximate Pattern Matching* of short degenerate and weighted sequences, derived from Deep sequencing technologies, to a reference genome.

Keywords: deep sequencing; high-throughput sequencing; string algorithms; degenerate sequences; weighted sequences.

Reference to this paper should be made as follows: Antoniou, P., Iliopoulos, C.S., Mouchard, L. and Pissis, S.P. (2009) 'Algorithms for mapping short degenerate and weighted sequences to a reference genome', *Int. J. Computational Biology and Drug Design*, Vol. 2, No. 4, pp.385–397.

Biographical notes: Pavlos Antoniou received his PhD in Computer Science at King's College London. He subsequently worked as a Visiting Bioinformatician in Cyprus Institute of Neurology and Genetics, where he designed software for microarray data analysis. He is currently a Visiting Lecturer in the Department of Computer Science at the University of Cyprus. His research interests include bioinformatics, string algorithms, and their applications to biological sequences and array-cgh data analysis.

Costas S. Iliopoulos received his PhD in Computer Science at the University of Warwick, UK. He is currently a Professor at King's College London in the Computer Science Department. He is the Head of the Algorithms Design Group. His research interests include string algorithms, computational biology and computational musicology.

Laurent Mouchard started working in Computational Biology in 1992. He was a member of the Informatics Research group at Celera Genomics in 2000, where he participated to the assembly of the Human Genome. Later he studied various combinatorial problems related to repeats in genomic sequences. For several years now, his main interests include genomic sequences analysis and more recently functional medical imaging.

Solon P. Pissis is a PhD student under Professor Costas S. Iliopoulos supervision in the Department of Computer Science, King's College London. He is a member of the Algorithms Design Group and his current research is focused on algorithms for Bioinformatics, string algorithms and parallel algorithms.

1 Introduction

The computational biology applications that have been developed for decades are strongly related to the technology that generates the data they consider. The algorithms, and the application parameters, are tuned in such a way that they abolished intrinsic limitations of the technology. As an example, the length of the data to be processed, or the quality/error rate that accompanies this data, are crucial elements that are considered for choosing the appropriate data structure for preprocessing, storing, analysing, and comparing sequences. Moreover, the way solutions are improved reflects both computer science and biotechnology advances.

Among the large number of equipment that produce data, the DNA sequencers play a central role. DNA sequencing is the generic term for all biochemical methods that determine the order of the nucleotide bases in a DNA sequence. It consists of obtaining (generally relatively short) fragments of a DNA sequence (typically less than a thousand bp – base pair). The Sanger sequencing method, presented by Sanger and Coulson (1975), Sanger et al. (1977), has been the workhorse technology for DNA sequencing for almost 30 years. It has been slowly replaced by technologies that used different coloured fluorescent dyes (Prober et al., 1987; Smith et al., 1986), and polyacrylamide gels. Later, the gels were replaced by capillaries, increasing the length of individually obtained fragments from 450 bp to 850 bp. Despite the many technological advances, obtaining the complete sequence of a genome was carried out in very large dedicated 'sequencing factories'. These factories require hundreds of automated sequencers using highly automated pipelines.

Along the years, sophisticated algorithms have been developed for assembling whole genomes, from a simple bacterial genome (Fleischmann et al., 1995), to the human genome (Celera Genomics, 2001). These algorithms were following the progress of the sequencing technologies, and were fully taking into account all the biases introduced by the equipment.

Very recent advances, based either on Sequencing-By-Synthesis (SBS) or on hybridisation and ligation, are producing millions of short reads overnight. Depending on the technology (454 Life Science, Solexa/Illumina or Polony Sequencing, to name a few), the size of the fragments can range from a dozen of base pairs to several hundreds.

These Deep sequencing technologies have the potential to assemble a bacterial genome during a single experiment and at a moderate cost (Hernandez et al., 2008), and are aimed in sequencing DNA genomic sequences. One such technology, Pyrosequencing^{TM}, massively parallelises the sequencing via microchip sensors and nanofluids, and it produces reads that are approximately 200 bp long, and may not improve beyond 300 bp in the near future (Sundquist et al., 2007). In contrast, the technology developed by the Solexa/Illumina (Bennett, 2004), generates millions of very short mate-pair reads ranging from 25 bp to 50 bp long, although in the near future this number will be increased to 75 (Warren et al., 2007; Hernandez et al., 2008). The results of these new technologies mark the beginning of a new era of high-throughput short read sequencing that moves away form the traditional Sanger methods. The common denominator of these technologies is the fact that they are able to produce a massive amount of relatively short reads. Due to this massive amount of data generated by the above systems, efficient algorithms for mapping short sequences to a reference genome are in great demand.

Popular alignment programs like *BLAST* or *BLAT* are not successful because they focus on the alignment of fewer and longer sequences (Jiang and Wong, 2008). Therefore these programs are not suitable to map the newly generated short sequences to genomes. A new thread of applications addressing the short sequence mapping problem has been developed recently. These applications include *ELAND*, *SeqMap* and *SOAP*, and are based on the pigeonhole principle, which entails the use of indexing and hashing techniques.

ELAND is the mapping algorithm developed as part of the Illumina pipeline. It is optimised to map very short reads of 20–32 bp ignoring additional bases when the reads are longer, whilst allowing at most two mismatches between the read and the genomic sequence (Smith et al., 2008). *SOAP* (Li et al., 2008a), supports

multi-threaded parallel computing and allows up to two mismatches, or a gap of 1–3 bp without any other mismatch. *SeqMap* (Jiang and Wong, 2008), allows up to five mixed mismatches and inserted/deleted nucleotides in mapping.

RMAP (Smith et al., 2008) and *MAQ* (Li et al., 2008b) are ungapped mapping programs, which take read qualities, base position probabilities, and mate-pair information into account. Their strategies resemble the strategies developed by Ewing et al. (1998), that entails an assignment of a quality score to each position of the reads. The quality score encodes the probability that the base identified at a position is either correctly positioned or not.

The last two applications show the necessity for a measure of accuracy concerning the mapping methods. Accuracy can be quantified in terms of sensitivity and specificity. Possible causes of limitations in the accuracy of these experiments include sequencing errors, variation between sample genome and the reference genome, as well as ambiguities caused by repeats in the reference genome (Smith et al., 2008).

Therefore, the limitations of the equipment used, or the natural polymorphisms that can be observed between individual samples, can give rise to uncertain sequences. In uncertain sequences, in some positions more than one nucleotide can be incorrectly identified. There is referred to these sequences as *degenerate* or *indeterminate* sequences.

Degenerate string pattern matching has mainly been handled by bit mapping techniques (Baeza-Yates and Gonnet, 1992; Wu and Manber, 1992b). These techniques have been used to find matches for a degenerate pattern in a string (Holub et al., 2008), and the *agrep* utility (Wu and Manber, 1992a), has been virtually one of the few practical algorithms available for degenerate pattern matching.

Very often, each position of a sequence is accompanied by probabilities of each base occurring in the specific position. In the case of the high-throughput experiments, these quality scores, which accompany the raw sequence data, describe the confidence of bases in each read (Smith et al., 2008). The sequencing quality scores assign a probability to the four possible nucleotides for each sequenced base. Bases with low quality scores are more likely to be sequencing errors. These sequences, where the probability of every symbol's occurrence at every location is given, are called *weighted* sequences.

Weighted sequences are also used to represent relatively short sequences such as binding sites, as well as long sequences such as protein families profiles (Christodoulakis et al., 2006). Additionally, they have been used to represent complete chromosome sequences that were obtained using the traditional method of whole-genome shotgun strategy.

In this paper, we address the problem of efficiently mapping and classifying millions of short degenerate and weighted sequences to a reference genome, based on whether they occur exactly once in the genome or not, and by taking into consideration probability scores. We are considering a reference genome, which is therefore unambiguous, and will be represented as a solid text. The short sequences we would like to map on that reference genome might contain ambiguities, as indicated by their quality score. In particular, we define and solve the *Massive Exact and Approximate Pattern Matching* problem for mapping short degenerate and weighted sequences, derived from Deep sequencing technologies, to a reference genome.

Our approach preprocesses the genomic sequence, based on the short reads length, by using word-level parallelism and sorting. We do not hash the short reads, but instead we convert each read to a unique arithmetic value, using 2-bits-per-base encoding of the DNA alphabet, and then use, the pigeonhole principle, binary search, and simple word-level operations for the mapping.

The rest of the paper is organised as follows. In Section 2, the basic definitions that are used throughout the paper are presented. Section 3 formally defines the problems solved in this paper. Sections 4 and 5 present the proposed algorithms for the exact and the approximate case, respectively. In Section 6, some experimental results are presented. Finally, we briefly conclude with some future proposals in Section 7.

2 Preliminaries

A *string* is a sequence of zero or more symbols from an alphabet Σ. In this work, we are considering the finite alphabet Σ for DNA sequences, where $\Sigma = \{A, C, G, T\}$. The length of a string x is denoted by $|x|$. The ith symbol of a string x is denoted by $x[i]$. A string w is a *substring* of x if $x = uwv$, where $u, v \in \Sigma^*$. We denote by $x[i \dots j]$ the substring of x that starts at position i and ends at position j.

A *degenerate* string is a sequence $t = t[1 \dots n]$, where $t[i] \subseteq \Sigma$ for each i. When a position of the string is degenerate, and it can match more than one element from the alphabet Σ, we say that this position has *non-solid* symbol. If in a position only one element of the alphabet Σ is present, we refer to this symbol as *solid*.

A *weighted* string over alphabet Σ is a sequence $s = s[1 \dots n]$ of sets of couples. In particular, each $s[i]$ is a set $((q_1, \pi_i(q_1)), (q_2, \pi_i(q_2)), \dots, (q_{|\Sigma|}, \pi_i(q_{|\Sigma|})))$, where $\pi_i(q_j)$ is the occurrence probability of character q_j at position i. A symbol q_j occurs at position i of a weighted sequence $s = s[1 \dots n]$ if and only if the probability of occurrence of symbol q_j at position i is greater than zero, i.e., $\pi_i(q_j) > 0$. For every position $1 \leq i \leq n$, $\sum_{j=1}^{|\Sigma|} \pi_i(q_j) = 1$. For example, $\binom{A\ 0.8}{C\ 0.2}$ is a non-solid symbol, implying that base A occurs with probability 80% and C with probability 20%.

For two strings x and y, such that $|x| = |y|$, the Hamming distance $\delta_H(x, y)$ is the number of places in which the two strings differ, i.e., have different characters. Formally

$$\delta_H(x, y) = \sum_{i=1}^{|x|} 1_{x[i] \neq y[i]} \tag{1}$$

where $1_{x[i] \neq y[i]} = \begin{cases} 1, & \text{if } x[i] \neq y[i] \\ 0, & \text{otherwise} \end{cases}$.

3 Problems definition

We denote the generated short reads as the set $p_0, p_1, \dots, p_{r-1}$, where $r > 10^7$, and we call them *patterns*. The length of each pattern is currently, typically between 25 bp and 50 bp long. Without loss of generality, we denote the length of the

patterns as ℓ. We assume that the data is derived from high quality sequencing methods and therefore we will consider patterns with at most $\mu = 3$ non-solid symbols. We are given a genomic solid sequence $t = t[1 \dots n]$, where $n > 10^8$, and a positive threshold $k \geq 0$.

We define the *Massive Exact and Approximate Pattern Matching* problem for mapping degenerate and weighted sequences to a reference genome, as follows.

Problem 1: *Find whether the degenerate pattern* $p_i = p_i[1 \dots \ell]$, *for all* $0 \leq i < r$, *with at most* μ *non-solid symbols, occurs with at most* k*-mismatches in* $t = t[1 \dots n]$.

Problem 2: *Find whether the weighted pattern* $p_i = p_i[1 \dots \ell]$, *for all* $0 \leq i < r$, *occurs with at most* k*-mismatches in* $t = t[1 \dots n]$, *with probability at least* γ, *if* $\prod_{j=1}^{\ell} \pi_j(q_j) \geq \gamma$.

We mainly focus on the following classes of both problems:

Class 1: p_i occurs in t exactly once.

Class 2: p_i occurs with at most 1-mismatch in t.

Class 3: p_i occurs with at most 2-mismatches in t.

The uniqueness of a mapped read guarantees an adequate placement on the sequence, and provides anchors that will be used for placing mate-pair reads, as well as other connected reads, as explored by Li et al. (2008b).

Class 2 and Class 3 correspond to cases that the pattern either contains a sequencing error (quality score associated with read is indicating it), or a small difference between a mutant and the reference genome, as explored by Sultan et al. (2008) and Wang et al. (2008).

4 Massive exact pattern matching

In this section, we solve the problem of Class 1. The focus is to find occurrences of pattern p_i, for all $0 \leq i < r$, in text $t = t[1 \dots n]$. In particular, we are interested in whether p_i occurs in t exactly once. In order for the procedure to be efficient we will make use of word-level parallelism by transforming each substring of length ℓ of t into a *signature*. We get the signature $\sigma(x)$ of a string x, by transforming it to its binary equivalent using 2-bits-per-base encoding of the DNA alphabet, and storing its decimal value into a computer word.

4.1 Algorithm I

Our aim is to preprocess text t and create two lists Λ and Λ'. Both lists hold elements of type $(i, \sigma(z_i))$, where i represents the starting position of substring $z_i = t[i \dots i + \ell - 1]$. List Λ holds each duplicate substring of length ℓ of t, whereas list Λ' holds each unique substring of length ℓ of t.

An outline of Algorithm I is as follows.

Step 1: We partition the text with the help of a window (sliding window mechanism), which size is ℓ, into a set of substrings $z_1, z_2, \ldots, z_{n-\ell+1}$, where $z_i = t[i \ldots i+\ell-1]$, for all $1 \leq i \leq n-\ell+1$. We compute the signature $\sigma(z_i)$ of each substring z_i, pack it into the couple $(i, \sigma(z_i))$, and add the couple to a list $\mathcal{L}$. Notice that, as soon as we compute $\sigma(z_1)$, then each $\sigma(z_i)$, for all $2 \leq i \leq n-\ell+1$, can be obtained in constant time (using a 'shift'-type operation).

Step 2: We sort the list $\mathcal{L}$ based on the signature field $\sigma(z_i)$, using a well-known sorting algorithm e.g., *Quicksort* (Cormen et al., 2001).

Step 3: We run sequentially through the sorted list $\mathcal{L}$ and check whether the signatures in $\mathcal{L}[i]$ and $\mathcal{L}[i+1]$ are equal or not, for all $1 \leq i \leq n-\ell+1$. If they are equal, then the encoded substrings stored in $\mathcal{L}[i]$ and $\mathcal{L}[i+1]$ are duplicates, and we have to check whether the signatures in $\mathcal{L}[i]$ and $\mathcal{L}[i-1]$ are equal or not. If they are equal, then the encoded substring in $\mathcal{L}[i]$ is a duplicate, which has already been added to the list Λ. If they are not equal, we add $\mathcal{L}[i]$ to Λ. If $\mathcal{L}[i]$ and $\mathcal{L}[i+1]$ are not equal, we check whether the signatures in $\mathcal{L}[i]$ and $\mathcal{L}[i-1]$ are equal or not. If they are equal, then the encoded substring in $\mathcal{L}[i]$ is a duplicate, which has already been already added to Λ. If they are not equal, then the encoded substring in $\mathcal{L}[i]$ is a unique substring and it is added to the list Λ'.

Main Step: Assume that we have a degenerate or weighted query $p_i[1 \ldots \ell]$, for all $1 \leq i \leq r$. Let

$$p_i = p_i[1] \times p_i[2] \times \cdots \times p_i[\ell] \tag{2}$$

the Cartesian product. For each $p_{i_s} \in p_i$, $1 \leq s \leq |\Sigma|^{\mu}$, we compute signature $\sigma(p_{i_s})$. In addition to that, if p_i is a weighted pattern (Problem 2), we must check that $\prod_{j=1}^{\ell} \pi_j(q_j) \geq \gamma$ is satisfied.

We can check whether p_i occurs in t, exactly once, by using binary search, as follows.

1 If there exists only one p_{i_s}, such that $p_{i_s} \in p_i \; AND \; \sigma(p_{i_s}) \in \Lambda'$, then p_i is a unique pattern, and the algorithm returns its starting position in t.

2 If there exists one or more p_{i_s}, such that $p_{i_s} \in p_i \; AND \; \sigma(p_{i_s}) \in \Lambda$, or more than one p_{i_s}, such that $p_{i_s} \in p_i \; AND \; \sigma(p_{i_s}) \in \Lambda'$, then p_i occurs in t more than once.

3 If $\sigma(p_{i_s}) \notin \Lambda$ and $\sigma(p_{i_s}) \notin \Lambda'$, for all $p_{i_s} \in p_i$, then p_i does not occur in t.

Theorem 1: *Given the text $t = t[1 \ldots n]$, the set of patterns $p_0, p_1, \ldots, p_{r-1}$, the length ℓ of the patterns, and the word size w of the machine, Algorithm I solves Class 1 of Problems* 1 *and* 2, *in $\mathcal{O}(\lceil \ell/w \rceil n \log n + \lceil \ell/w \rceil r |\Sigma|^{\mu} \ell \log n)$ units of time.*

Proof: In Step 1, we compute the signatures in $\mathcal{O}(\lceil \ell/w \rceil n)$ time. In Step 2, the time required for sorting the list $\mathcal{L}$ is $\mathcal{O}(\lceil \ell/w \rceil n \log n)$. In Step 3, creating the two lists Λ and Λ' requires $\mathcal{O}(\lceil \ell/w \rceil n)$ time. The main step runs in $\mathcal{O}(\lceil \ell/w \rceil r |\Sigma|^{\mu} \ell \log n)$ time. Hence, asymptotically, the overall time is $\mathcal{O}(\lceil \ell/w \rceil n \log n + \lceil \ell/w \rceil r |\Sigma|^{\mu} \ell \log n)$, which is $\mathcal{O}(n \log n + r \ell \log n)$, in practice. □

Notice that in the case that $2\ell > w$, where w is equal to 32 or 64 in practice, our algorithm can easily be modified to store the signatures in $\lceil 2\ell/w \rceil$ computer words.

5 Massive approximate pattern matching

In this section, we solve the problem of Class 2 and Class 3. The focus is to find occurrences of p_i, for all $0 \leq i < r$, in text $t = t[1 \ldots n]$ with at most k-mismatches. In particular, we are interested whether p_i occurs with at most 1-mismatch in t, for the problem of Class 2, or with at most 2-mismatches, for the problem of Class 3.

5.1 Algorithm II

Here, the idea of using the pigeonhole principle to split each signature into ν fragments is adopted. By requiring some of the ν fragments (instead of all of them) to be perfectly matched, the non-candidates can be filtered out very quickly (Jiang and Wong, 2008). For example, to admit two mismatches (Class 2), a read can be splitted into four fragments. The two mismatches can exist in at most two of the fragments (at the same time). Then, if we try all six combinations of the two fragments as the seed, we can catch all hits with two mismatches (Li et al., 2008a).

Lemma 1: *Given the number of fragments ν of a string $x = \{x^1, x^2, \ldots, x^\nu\}$, and the number of allowed mismatches k, $k \leq \nu$, any of the k mismatches cannot exist, at the same time, in at least $\nu - k$ fragments of x.*

Proof: Immediate from the pigeonhole principle. □

Our aim is to preprocess t and construct ν sorted lists $\mathcal{L}_j$, for all $1 \leq j \leq \nu$. Each list $\mathcal{L}_j$ holds elements of type $e(z_i^j) = (i, \sigma(z_i^j), prev_i^j, next_i^j)$, where i represents the starting position of substring $z_i = t[i \ldots i + \ell - 1]$, $prev_i^j$ points to the element of the previous fragment in $\mathcal{L}_{j-1}$, for all $2 \leq j \leq \nu$, and $next_i^j$ points to the element of the next fragment in $\mathcal{L}_{j+1}$, for all $1 \leq j \leq \nu - 1$. Let $e(z_i) = \{e(z_i^1), \ldots, e(z_i^\nu)\}$ and $c_g(\sigma(x)) = \{\sigma(x^{q_1}), \ldots, \sigma(x^{q_{\nu-k}})\}$, for all $1 \leq g \leq \binom{\nu-k}{\nu}$, the $\binom{\nu-k}{\nu}$ combinations of $\sigma(x) = \{\sigma(x^1), \ldots, \sigma(x^\nu)\}$. We define the following operations:

- *find-all* $(c_g(\sigma(x)), \mathcal{L})$: It returns $A_g = \{e(z_{u_1}), \ldots, e(z_{u_v})\}$, such that for each $\sigma(x^{q_j}) \in c_g(\sigma(x))$, for all $1 \leq j \leq \nu - k$, $\sigma(x^{q_j}) \in e(z_{u_i}^{q_j})$, for all $1 \leq i \leq v$. It uses binary search to first locate $\sigma(x^{q_1})$ in list $\mathcal{L}_{q_1}$ in $\mathcal{O}(\log n)$ time, and then return $\sigma(x^{q_j}) \in c_g(\sigma(x))$, for all $2 \leq j \leq \nu - k$, with the use of pointers, in $\mathcal{O}(v\nu)$.
- *find-rest* $(A_g, c_g(\sigma(x)), \sigma(x))$: It returns $T_g = \{e(z_{u_1}), \ldots, e(z_{u_v})\}$, such that for each $\sigma(x^{q_j}) \in \{\sigma(x) - c_g(\sigma(x))\}$, for all $1 \leq j \leq k$, $\sigma(x^{q_j}) \in e(z_{u_i}^{q_j})$, for all $1 \leq i \leq v$. Given A_g, it returns T_g in $\mathcal{O}(vk)$.
- *bitop* $(\sigma(x), \sigma(y))$: It returns $\delta_H(x, y)$ in constant time, given that $|x| = |y| = \alpha$ and $2\alpha \leq w$, where w is the size of the computer word.

An outline of Algorithm II is as follows.

Step 1: We partition the text into a set of substrings $z_1, z_2, \ldots, z_{n-\ell+1}$, where $z_i = t[i \ldots i+\ell-1]$, for all $1 \leq i \leq n-\ell+1$. We compute $\sigma(z_i)$, split it into ν fragments $\sigma(z_i) = \{\sigma(z_i^1), \sigma(z_i^2), \ldots, \sigma(z_i^\nu)\}$, and add $(i, \sigma(z_i^j), i, i)$, to a list $\mathcal{L}_j$, for all $1 \leq j \leq \nu$. As soon as we compute $\sigma(z_1)$, then each $\sigma(z_i)$, for all $2 \leq i \leq n-\ell+1$, can be retrieved in constant time (using 'shift'-type of operation).

Step 2: We sort the lists $\mathcal{L}_j$, for all $1 \leq j \leq \nu$, based on the signature field $\sigma(z_i^j)$, ensuring that, in a case that we swap elements, we preserve that $prev_i^{j+1}$ and $next_i^{j-1}$ point to the swapped element.

Main Step: Assume that we have a degenerate or weighted query $p_i[1 \ldots \ell]$, for all $1 \leq i \leq r$. For each $p_{i_s} \in p_i$, $1 \leq s \leq |\Sigma|^\mu$, we compute signature $\sigma(p_{i_s})$. In addition to that, if p_i is a weighted pattern (Problem 2), we must check that $\prod_{j=1}^{\ell} \pi_j(q_j) \geq \gamma$ is satisfied. We split $\sigma(p_{i_s})$ into ν fragments $\sigma(p_{i_s}) = \{\sigma(p_{i_s}^1), \sigma(p_{i_s}^2), \ldots, \sigma(p_{i_s}^\nu)\}$.

We compute $c_g(\sigma(p_{i_s}))$, for all $1 \leq g \leq \binom{\nu}{\nu-k}$, and perform:

1 $A_g = \textit{find-all}\ (c_g(\sigma(p_{i_s})), L)$

2 $T_g = \textit{find-rest}\ (A_g, c_g(\sigma(p_{i_s})), \sigma(p_{i_s}))$.

If there exists $e(z_{u_j}) \in T_g$, for all $1 \leq j \leq v$, $1 \leq g \leq \binom{\nu}{k}$, such that $\sum_{\lambda=q_1}^{q_k} \textit{bitop}(\sigma(p_{i_s}^\lambda), \sigma(z_{u_j}^\lambda)) \leq k$, for all $\sigma(z_{u_j}^\lambda) \in e(z_{u_j}^\lambda)$, then p_i occurs in t with at most k-mismatches, and the algorithm returns its starting position in t. If there does not exist $e(z_{u_j}) \in T_g$, for all $1 \leq j \leq v$, $1 \leq g \leq \binom{\nu}{k}$, such that $\sum_{\lambda=q_1}^{q_k} \textit{bitop}(\sigma(p_{i_s}^\lambda), \sigma(z_{u_j}^\lambda)) \leq k$, for all $\sigma(z_{u_j}^\lambda) \in e(z_{u_j}^\lambda)$, then p_i does not occur in t.

Theorem 2: *Given the text $t = t[1 \ldots n]$, the set of patterns $p_0, p_1, \ldots, p_{r-1}$, the length of the patterns ℓ, the number of fragments ν, and the number of mismatches k, Algorithm II solves Class 2 and Class 3 of Problems 1 and 2, in $\mathcal{O}(\nu n \log n + r|\Sigma|^\mu \ell \binom{\nu}{\nu-k} \log n)$ units of time.*

Proof: Step 1 can be done in $\mathcal{O}(n)$ time. In Step 2, the time required for sorting the list $\mathcal{L}_j$, for all $1 \leq j \leq \nu$, is $\mathcal{O}(\nu n \log n)$. The main step runs in $\mathcal{O}(r|\Sigma|^\mu \ell \binom{\nu}{\nu-k} \log n)$ time. Hence, asymptotically, the overall time is $\mathcal{O}(\nu n \log n + r|\Sigma|^\mu \ell \binom{\nu}{\nu-k} \log n)$, which is $\mathcal{O}(\nu n \log n + r\ell \binom{\nu}{\nu-k} \log n)$, in practice. □

6 Experimental results

In order to evaluate the correctness and efficiency of Algorithms I and II, we have mapped 1,872,404 Illumina 25 bp degenerate reads, taken from RNA-Seq experiments (Mortazavi et al., 2008), to genomic sequences of various lengths of the mouse chromosome X, as well as to the whole chromosome sequence (166,650,296 bp), allowing at most 2-mismatches. Notice that, these sequences were obtained from a set of $31,116,663$ Illumina 25 bp reads, such that for each degenerate read, $1 < \mu \leq 3$, i.e., we have discarded all the reads with less than 1 and more than 3 'N's. Without loss of generality concerning the complexity of the

algorithms, we assign the same probability of each base occurring in each non-solid position. The experimental results of Algorithms I and II are illustrated in Tables 1 and 2, respectively.

Table 1 Mapping and classifying 1,872,404 Illumina-Solexa 25 bp degenerate reads to the mouse chromosome X, allowing no mismatches

Genomic sequence (bp)	*Running time* (s)	*Unique*	*Duplicate*	*Mapped reads (total)*
40,000,000	18	1,352	6,145	7,497
80,000,000	27	2,422	9,234	11,656
166,650,296	45	3,132	13,637	16,769

Table 2 Mapping 1,872,404 Illumina-Solexa 25 bp degenerate reads to the mouse chromosome X, allowing at most 2-mismatches

Genomic sequence (bp)	*Running time* (s)	*Mapped reads*
40,000,000	211	226,946
80,000,000	411	368,460
166,650,296	927	554,532

The presented experimental results are very promising concerning both, the efficiency of the proposed algorithms, and the sensitivity of our approach in terms of mapping. Algorithm I maps 16,769 out of 1,872,404 Illumina-Solexa 25 bp degenerate reads to the mouse chromosome X in 45 s, allowing no mismatches. Naturally, we expect the number of mapped reads to increase when allowing for a small number of mismatches. Algorithm II maps 554,532 out of 1,872,404 Illumina-Solexa 25 bp degenerate reads to the mouse chromosome X in 927 s, allowing at most 2-mismatches.

The proposed algorithms were implemented in ANSI C language. The implementation is available at a website (http://www.dcs.kcl.ac.uk/pg/pississo/), which has been set up for maintaining the source code and the documentation. The programs accept FASTA format for the reference and the short reads. The experiments were conducted on a machine with 3 GHz Intel Xeon CPU and 16 GB memory, running 64-bit Linux operating system.

7 Discussion

In this paper, we have presented novel and efficient algorithms to tackle the data emerging from the new Deep sequencing technologies. The new technologies produce a huge number of very short sequences, and these sequences need to be classified, tagged and recognised as parts of a reference genome. The proposed algorithms can manipulate this data for Massive Exact and Approximate Pattern Matching of degenerate and weighted sequences to a reference genome.

We have presented Algorithm I for the exact matching. It runs in $\mathcal{O}(n \log n + r\ell \log n)$ time, where n is the length of the genomic sequence, r is the number of

patterns, and ℓ is the length of the patterns. In addition to that, we have presented an algorithm for the approximate matching. Algorithm II runs in $\mathcal{O}(\nu n \log n + r\ell\binom{\nu}{\nu-k} \log n)$ time, where ν is the number of equal length fragments into which each pattern is split, and k is the number of allowed mismatches. The presented experimental results are very promising, in terms of efficiency and sensitivity on mapping.

Algorithm I was presented by Antoniou et al. (2009b), as part of the conference version of this paper. A variation of Algorithm I, for Massive Exact Pattern Matching of solid sequences to a reference genome, was implemented on a real dataset, and presented by Antoniou et al. (2009a). A variation of Algorithm II, for Massive Approximate Pattern Matching of solid sequences to a reference genome, was implemented on a real dataset, and presented by Antoniou et al. (2010).

Our immediate target is to build a software tool, which will be based on the presented algorithms, and will be used by the biologists for mapping short sequences to a reference genome. Due to the massive amount of data generated by the Deep sequencing technologies, parallelising the presented algorithms, with the use of the message-passing parallelism model, is potentially a further target worth investigating.

Acknowledgements

The authors would like to thank the Editors of this Journal.

References

Antoniou, P., Daykin, J.W., Iliopoulos, C.S., Kourie, D., Mouchard, L. and Pissis, S.P. (2009a) 'Mapping uniquely occurring short sequences derived from high throughput technologies to a reference genome', *Proceedings of the 9th International Conference on Information Technology and Applications in Biomedicine (ITAB'09)*, (in press).

Antoniou, P., Iliopoulos, C.S., Mouchard, L. and Pissis, S.P. (2009b) 'Practical and efficient algorithms for degenerate and weighted sequences derived from high throughput sequencing technologies', *Proceedings of the International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing (IJCBS'09)*, Shanghai, China, pp.164–170.

Antoniou, P., Iliopoulos, C.S., Mouchard, L. and Pissis, S.P. (2010) 'A fast and efficient algorithm for mapping short sequences to a reference genome', *Advances in Computational Biology*, Advances in Experimental Medicine and Biology, Springer (in press).

Baeza-Yates, R. and Gonnet, G. (1992) 'A new approach to text searching', *Commun. ACM*, Vol. 35, pp.74–82.

Bennett, S. (2004) 'Solexa ltd', *Pharmacogenomics*, Vol. 5, No. 4, pp.433–438.

Christodoulakis, M., Iliopoulos, C.S., Mouchard, L., Perdikuri, K., Tsakalidis, A. and Tsichlas, K. (2006) 'Computation of repetitions and regularities on biological weighted sequences', *Journal of Computational Biology*, Vol. 13, No. 6, pp.1214–1231.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2001) *Introduction to Algorithms*, 2nd ed., The MIT Press.

Dömölki, B. (1964) 'An algorithm for syntactic analysis', *Computational Linguistics*, Vol. 8, pp.29–46.

Ewing, B., Hillier, L., Wendl, M.C. and Green, P. (1998) 'Base-calling of automated sequencer traces using phred. I. Accuracy assessment', *Genome Research*, Vol. 8, No. 3, pp.175–185.

Fleischmann, R.D., Adams, M.D., White, O., Clayton, R.A., Kirkness, E.F., Kerlavage, A.R., Bult, C.J., Tomb, J.F., Dougherty, B.A. and Merrick, J.M. (1995) 'Whole-genome random sequencing and assembly of *Haemophilus influenzae*', *Science*, Vol. 269, pp.496–512.

Celera Genomics (2001) 'The sequence of the human genome', *Science*, Vol. 291, pp.1304–1351.

Hernandez, D., Francois, P., Farinelli, L., Osteras, M. and Schrenzel, J. (2008) 'De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer', *Genome Research.*, Vol. 18, No. 5, pp.1518–1525.

Holub, J., Smyth, W.F. and Wang, S. (2008) 'Fast pattern-matching on indeterminate strings', *J. Discrete Algorithms*, Vol. 6, No. 1, pp.37–50.

Jiang, H. and Wong, W.H. (2008) 'Seqmap: mapping massive amount of oligonucleotides to the genome', *Bioinformatic*, Vol. 24, No. 20, p.2395, 2396.

Li, R., Li, Y., Kristiansen, K. and Wang, J. (2008a) 'SOAP: short oligonucleotide alignment program', *Bioinformatics*, Vol. 24, No. 5, p.713, 714.

Li, H., Ruan, J. and Durbin, R. (2008b) 'Mapping short DNA sequencing reads and calling variants using mapping quality scores', *Genome Research*, Vol. 18, No. 11, pp.1851–1858.

Mortazavi, A., Williams, B.A.A., McCue, K., Schaeffer, L. and Wold, B. (2008) 'Mapping and quantifying mammalian transcriptomes by *RNA-Seq*', *Nature Methods*, Vol. 5, No. 7, pp.621–628.

Prober, J.M., Trainor, G.L., Dam, R.J., Hobbs, F.W., Robertson, C.W., Zagursky, R.J., Cocuzza, A.J., Jensen, M.A. and Baumeister, K. (1987) 'A system for rapid DNA sequencing with fluorescent chain-terminating dideoxynucleotides', *Science*, Vol. 238, No. 4825, pp.336–341.

Sanger, F. and Coulson, A.R. (1975) 'A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase', *J. Mol. Biol.*, Vol. 94, pp.441–448.

Sanger, F., Nicklen, S. and Coulson, A.R. (1977) 'DNA sequencing with chain-terminating inhibitors', *Proc. Natl. Acad. Sci., USA*, Vol. 74, pp.5463–5467.

Smith, A., Xuan, Z. and Zhang, M. (2008) 'Using quality scores and longer reads improves accuracy of solexa read mapping', *BMC Bioinformatics*, Vol. 9, No. 1, p.128.

Smith, L.M., Sanders, J.Z., Kaiser, R.J., Hughes, P., Dodd, C., Connell, C.R., Heiner, C., Kent, S.B. and Hood, L.E. (1986) 'Fluorescence detection in automated DNA sequence analysis', *Nature*, Vol. 321, No. 6071, pp.674–679.

Sultan, M., Schulz, M.H., Richard, H., Magen, A., Klingenhoff, A., Scherf, M., Seifert, M., Borodina, T., Soldatov, A., Parkhomchuk, D., Schmidt, D., O'Keeffe, S., Haas, S., Vingron, M., Lehrach, H. and Yaspo, M.L. (2008) 'A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome', *Science*, Vol. 321, No. 5891, pp.956–960.

Sundquist, A., Ronaghi, M., Tang, H., Pevzner, P. and Batzoglou, S. (2007) 'Whole-genome sequencing and assembly with high-throughput short-read technologies', *PLoS ONE*, Vol. 2, No. 5, p.e484.

Wang, Z., Gerstein, M. and Snyder, M. (2008) 'RNA-seq: a revolutionary tool for transcriptomics', *Nature Reviews. Genetics*, Vol. 10, No. 1, pp.57–63.

Warren, R.L., Sutton, G.G., Jones, S.J. and Holt, R.A. (2007) 'Assembling millions of short DNA sequences using SSAKE', *Bioinformatics*, Vol. 24, No. 3, p.500, 501.

Wu, S. and Manber, U. (1992a) 'Agrep – a fast approximate pattern-matching tool', *Proceedings USENIX Winter 1992 Technical Conference*, San Francisco, CA, pp.153–162.

Wu, S and Manber, U. (1992b) 'Fast text searching: allowing errors', *Commun. ACM*, Vol. 35, No. 10, pp.83–91.

Bibliography

Hall, N. (2007) 'Advanced sequencing technologies and their wider impact in microbiology', *J. Exp. Biol.*, Vol. 210, No. 9, pp.1518–1525.

Iliopoulos, C.S., Mouchard, L. and Pinzon, Y. (2001) 'The max-shift algorithm for approximate string matching', *Proceedings of the 5th Workshop on Algorithm Engineering (WAE'01)*, August, Denmark, pp.13–25.

Karp, R.M. and Rabin, M.O. (1987) 'Efficient randomized pattern-matching algorithms', *IBM J. Res. Dev.*, Vol. 32, No. 2, pp.249–260.

Margulies, E.H. and Birney, E. (2008) 'Approaches to comparative sequence analysis: towards a functional view of vertebrate genomes', *Nat. Rev. Genet.*, Vol. 9, No. 4, pp.303–313.

Morozova, O. and Marra, M.A. (2008) 'Applications of next-generation sequencing technologies in functional genomics', *Genomics*, Vol. 95, No. 5, pp.255–264.

Myers, E.W. (1999) 'A fast bit-vector algorithm for approximate string matching based on dynamic progamming', *Journal of the ACM*, Vol. 46, pp.1–13.

Schuster, S.C. (2008) 'Next-generation sequencing transforms today's biology', *Nat. Methods*, Vol. 5, pp.16–18.

Shaner, M.C., Blair, I.M. and Schneider, T.D. (2008) 'Sequence logos: a powerful, yet simple, tool', *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, Vol. 1, Architecture and Biotechnology Computing, pp.813–821.

Wold, B. and Myers, R. (2008) 'Sequence consensus methods for functional genomics', *Nature Methods*, Vol. 5, No. 1, pp.19–21.