Querying Highly Similar Sequences

Carl Barton

King's College London, Dept. of Informatics, UK
Email: carl.barton@kcl.ac.uk

Mathieu Giraud

INRIA Lille Nord-Europe, France
Email: mathieu.giraud@lifl.fr

Costas S. Iliopoulos

King's College London, Dept. of Informatics, UK
Curtin University, Perth, Australia
Email: csi@kcl.ac.uk

Thierry Lecroq

University of Rouen, LITIS - EA4108, France
Email: thierry.lecroq@univ-rouen.fr

Laurent Mouchard

King's College London, Dept. of Informatics, UK
University of Rouen, LITIS - EA4108, France
Email: laurent.mouchard@univ-rouen.fr

Solon P. Pissis

Florida Museum of Natural History, University of Florida, USA
Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
Email: solon.pissis@h-its.org

Keywords: DNA sequencing, highly similar sequences, similarity searching, querying DNA sequences, next generation sequencing, NGS

Abstract: In this paper, we consider the Extreme Similarity Sequencing problem, where the focus is on finding occurrences of a pattern p in a set of sequences $S_0, S_1, \ldots, S_k$, where the sequences differ by a constant number of errors—around 10 in practice. We present an asymptotically fast $O(n + occ)$ algorithm, as well as a practical $O(\frac{nk}{w})$ algorithm for the extreme similarity sequencing problem, where n is the length of a sequence, occ is the number of candidate occurrences of reported by our technique, w is the size of the machine word, and the total number of errors is bounded by k, the number of sequences.

1 Introduction

DNA sequencing includes several methods and technologies that are used for determining the exact order of the nucleotide bases—adenine, guanine, cytosine, and thymine—in a DNA macromolecule.

Sequencing technology has come a long way since the time when traditional sequencing techniques required many laboratories around the world to cooperate for years in order to sequence the human genome for the first time. The traditional sequencing methods, developed in the mid 70's, had been the workhorse technology for DNA sequencing for almost thirty years. In 1977, Sanger and Coulson published two methodological papers on the rapid determination of DNA sequence (Sanger et al., 1977a; Sanger et al., 1977b), which would go on to revolutionise biology, as a whole, by providing a tool for analysing complete genes and, later, entire genomes. The method greatly improved earlier DNA sequencing techniques developed by Maxam and Gilbert (Maxam and Gilbert, 1977), published in the same year, and Sanger and Coulson's own "plus and minus" method, published two years earlier (Sanger and Coulson, 1975).

Recent advances in molecular biology have dramatically changed the way biological data analysis is performed (Margulies and Birney, 2008; Shendure et al., 2005). Next-Generation Sequencing technologies produce high-throughput data of highly controlled quality, one hundred times faster, and at a cost one hundred times smaller, than a decade ago. These technologies are very promising from the biological/medical viewpoints, opening new perspectives in terms of public health (Ng et al., 2010; Ostergaard et al., 2010; Simpson at al., 2011). The sequence of a human genome is made of approximately 3 billion nucleotides (A, T, C or G). Recently, a huge step forward has been made as scientists move away from using one generic consensus sequence representing a human genome to a large set of sequences of individual genomes. For instance, many algorithms and programmes have been published to deal with the task of efficiently mapping millions of short sequences to a reference sequence, namely Bowtie (Langmead et al., 2009) BWA (Heng and Richard, 2009), SOAP2 (Ruiqiang et al., 2009), REAL (Frousios et al., 2010). Studying such large sets of individual sequences helps in acquiring a better understanding of the correlation between phenotype (what a physician can observe at the human level) and genotype (what the molecular computational biologist can observe in the sequence at the cell level) and so called 'similarity searching' is an important problem in bioinformatics (Stevens et al., 2001).

Sets of sequences can be:

- Whole genome sequences: coming from Nuclear DNA, they are made of the four nucleotides and contain the sequences of all genes (that code proteins or other structural elements) and the sequences of all intergene bases.
- Exome sequences: portions of the whole genome sequences that contain only genes that will produce the proteins, subregions of these sequences will be transcribed into mRNAs that will eventually produce proteins.
- RNA sequences: cDNA sequences that have been obtained after retro transcribing existing mRNA sequences.

The study of these various types of sequences is important for a clear understanding of the mechanisms that induce diseases. It is well known that minor and local modifications, such as Single Nucleotide Polymorphisms (as stored in dbSNP (NCBI, 2012)), may produce isoforms of a given protein that prevent it from been produced. This can cause several types of cancer (Engle, Simpson, and Landers, 2006)), Alzheimer's disease (Emahazion et al., 2001) and many others. It is therefore crucial to be able to clearly distinguish between "silent" mutations and mutations of interest.

Due to the proliferation of genomic data since the assembly of the human genome, two major problems have recently arose:

1. Producing data is now quicker than analysing data and interpreting results.
2. The amount of produced data already exceeds the computers capacity, in terms of storing all the information in main memory or the time that is needed for properly processing it.

The scientific bottlenecks have moved from being able to produce interesting data for important societal health studies to being able to store, process and interpret the massive amount of data produced in numerous research centres. From the computer science viewpoint, storing and querying efficiently one sequence, or a limited set of sequences can be done using specialised data structures. But our ultimate goal is to be able to store not only one or several sequences, but hundreds or thousands, that is possibly more than 3 trillions base pairs.

In this paper, we consider the Extreme Similarity Sequencing problem, in which we are given sequences $S_0, S_1, \ldots, S_k$, where S_i and S_j differ by only a few symbols—around 10 in practice. Our goal is to keep one reference sequence, say S_R, together with the differences between S_R and S_i (for all i) in such a way that we can efficiently answer queries on the entire set of sequences.

Background

There is existing literature on querying biological sequences, generally focusing on developing indexing schemes to allow compression of the sequence. Basic indexing techniques often construct one long string by concatenating all string together and use auxiliary data structures such as suffix trees and BWT (Adjeroh et al., 2002), to build an index of the entire set of sequences; however, when there are many types of sequences where these are not appropriate due to memory usage and

the types of sequences we are using. Using classical indexes in this way does not exploit the large similarities between the DNA sequences, meaning they often end up storing extra, unnecessary, information. More advanced data structures such as CSA (Lippert, 2005) and FM-index (Ferragina and Manzini, 2000) present an improvement over classical data structures and take into account the entropy of a sequence to produce a compressed index of the sequence using a technique known as block addressing. Recently there has been a focus on exploiting the inherent similarity present in multiple DNA sequences, in (Huang et al., 2010) they assume a number of common segments in the DNA sequence with errors in between. This allows them to exploit the common segments for fast string searching and low memory usage. Other techniques sacrifice guaranteeing a match to improve the time complexity, these tend to use filtering techniques which reduce the number of potential matches (similar to BLAST (Altschul et al., 1990) or FASTA (Pearson, 1990), such as searching using q-grams (Cao, Li, and Tung., 2005).

In our solution we consider a different model of the problem and do not place such restrictions on the structure of the string. Instead we assume that there are a constant number of errors between the sequences and try to exploit this fact to allow for fast searching of a large number of sequences with a low memory footprint. Our construction is a remarkably simple method of searching a large number of sequences which avoids complex indexing and assumptions about the structure of the string.

The paper is structured as follows: in Section 1, we describe the motivation of the problem; in Section 2, we describe the background results needed to support our constructions; and in Section 3, we give an informal outline of the algorithm followed by analysis of the algorithm and conclusions.

2 Preliminaries

An *alphabet* Σ is a finite non-empty set whose elements are called *symbols*. In this work, we consider the finite alphabet Σ for DNA sequences, where $\Sigma = \{A, C, G, T\}$. A *string* is a sequence of zero or more symbols from an alphabet Σ. The zero-symbol sequence is called the *empty string*, and is denoted by ε. The set of all the strings on the alphabet Σ is denoted by Σ^*. The set of all the strings on the alphabet Σ except the empty string ε is denoted by Σ^+. The length of a string x is denoted by $|x|$.

We denote by $x[i]$, for all $0 \leq i < |x|$, the symbol at index i of x. Each index i, for all $0 \leq i < |x|$, is a position in x when $x \neq \varepsilon$. It follows that the ith symbol of x is the symbol at position $i - 1$ in x, and that

$$x = x[0\,..\,|x| - 1]$$

A string x is a *factor* of a string y if there exist two strings u and v, such that $y = uxv$.

Let x be a non-empty string and y be a string. We say that there exists an *occurrence* of x in y, or, more simply, that x *occurs in* y, when x is a factor of y.

Every occurrence of x can be characterised by a position in y. Thus we say that x occurs at the *starting position* i in y when $y[i\,..\,i + |x| - 1] = x$. It is sometimes more suitable to consider the *ending position* $i + |x| - 1$.

The Hamming distance δ_H is defined only for strings of the same length. For two strings x and y, $\delta_H(x, y)$ is the number of places in which the two strings differ. Let x be a nonempty string and y be a string, we say that x occurs in y with at most k-errors, if there exists a factor of y, say w, such that $\delta_H(x, w) \leq k$.

In the rest of this section we present a few existing results which are required for our constructions.

Lemma 1 *Given a text t and a pattern p we can find approximate occurrences of p with k errors in $O(n)$ time if k is bounded by a constant.*

Proof We know from (Landau and Vishkin, 1986) that we can find occurrences of p with k errors in $O(kn)$ time. So when k is bounded by a constant it is clear that the runtime is reduced to $O(n)$

Bit parallelism is a technique which exploits the ability to perform operations on bit vectors in constant time assuming the bit vector is $\leq w$ where w is the size of the machine word. This allows us to simulate a non-deterministic automaton in $O(\frac{nm}{w})$ time by updating multiple matchings for every word operation performed. The following Lemma is a consequence of this technique.

Lemma 2 *Given a text t of length n, a pattern p of length m and machine word of size w we can find occurrences of p in t in time $O(\frac{n}{w})$ if m is bounded by a constant.*

Proof We know from (Baeza-Yates and Gonnet, 1992) that we can perform string matching using bit word parallelism in time $O(\frac{mn}{w})$. It is clear that if we bound m by a constant this becomes $O(\frac{n}{w})$.

3 Main Ideas of the Algorithm

Here we present the main ideas of our construction, primarily that of minimising the cost of storing and querying the sequences.

3.1 Minimizing The Storage Space

Given the sequences $S_0, S_1, S_2, \ldots, S_k$ we will only keep S_0 and call it the reference sequence, denoted by S_R. Then, for all i, we keep the positions where S_R and S_i differ, storing them sequentially as a single string. We store the errors from S_1 followed by errors from S_2 and so on until S_k. In the sequence shown below the errors in the first division are those from S_1, the second from S_2 and so on. The vertical lines are added for clarity and are not part of the storage scheme.

146|07|13|23

We will also keep 2 additional lists $L(e_i) = i_m$ and $M(e_i) = \beta_i$ that given an error, e_i, will tell us in which sequence S_{i_m} the error occurred and the symbol that caused the error β_i. $L(e_i)$ can be stored very efficiently as a bit vector, with a 1 at the first error of a new sequence and a 0 otherwise. This gives us a sparse bit vector allowing us to efficiently perform rank/select operations (Golynski et al., 2011). It is clear that the minimum information we must store is one sequence, along with

all the errors. Our scheme stores one sequence along with a constant number of arrays all with length proportional to the number of errors in the sequences. This is an efficient method of storing many genomic sequences with a high degree of similarity.

3.2 Querying

Here we present an example of our querying algorithm for the extreme similarity sequencing problem. We are given the sequences S_R, S_1, S_2, S_3, S_4 and the pattern p, these are shown below with errors between S_R and S_i represented by bold and underlined symbols. We wish to find occurrences of p in the given sequences.

$$p = CTCG$$
$$S_R = ATCGAACG$$
$$S_1 = A\underline{\mathbf{C}}CG\underline{\mathbf{C}}A\underline{\mathbf{G}}G$$
$$S_2 = \underline{\mathbf{C}}TCGAAC\underline{\mathbf{A}}$$
$$S_3 = A\underline{\mathbf{C}}C\underline{\mathbf{A}}AACG$$
$$S_4 = AT\underline{\mathbf{AC}}AACG$$

Step 1

As mentioned previously, we will keep S_R and the list of differences. The differences are stored as a string of the positions of the errors, shown below and denoted by E.

$$E = 146|07|13|23$$

Step 2

Find p in S_R with a Hamming distance of at most 3. We wish to find those with a Hamming distance of at most 3 as that is the highest number of differences present in any sequence. It should be noted that the number of differences in one sequence is at most 10 regardless of the length of the sequence.

	A	T	C	G	A	A	C	G
1	0	✓	✓	✓				
2		×	×	×	×			
3			✓	3	4	5		
4				×	×	×	×	
5					4	5	✓	✓

✓ Match
× Mismatch
n Error at position n

We then check the candidate solutions, (those with less than 3 mismatches, in this case 1,3 and 5) against the list of differences, here called E, to find where they could occur. To match a candidate solution against E we construct a new sequence for each candiate solution which consists of the indicies of each mismatches in the candidate. We then take these new sequences and match them against E.

Using the example above the new sequences will be.

$$c_1 = 0$$

$$c_2 = 345$$

$$c_3 = 45$$

We now match these sequences against E and look for matches of a candidate sequence in E, in this case we see that the first candidate, c_1, has a match in E. We must now perform false match checking, by varifying that the next error in the sequence does not invalidate the match. It is clear that the length of the pattern is 4 so the error at 8 will have no effect on the validity of this match. We are now left to check that each symbol in the match is from the same sequence; this can be done using the $L(e_i)$ list. Finally we must varify that the error is the character we require, which can be achieved using the $M(e_i)$ list.

4 An Outline of the Algorithm

INPUT:
S_R a sequence of n symbols from an alphabet of A,C,G,T
A pattern p of length m over A,C,G,T
A list of positions $E = e_1 e_2 e_3 \dots e_\ell$
An array $L(e_i) = i_m$ for $1 \leq i \leq \ell$
An array $M(e_i) = \beta_i$ for $1 \leq i \leq \ell$

OUTPUT:
v, s such that p occurs in the s-th position of S_v

The Pseudocode for the algorithm can be found in Algorithm 1.

5 Correctness and Analysis

Theorem 1 *The algorithm correctly computes occurrences of p in $S_1 S_2 \dots S_k$*

Proof It is obvious that there are 2 cases where there can be a valid occurrence, those where p occurs exactly in S_R or those where p occurs as an approximate occurrence in S_R and exactly in some S_i.

Where p occurs exactly in S_R we simply report the occurrence and we are done.

Where p occurs approximately in S_R and exactly on S_i we must verify that the errors present in the approximate occurrence in S_R are also present in S_i, that there is no further error in S_i that would affect the match and that the errors are the same symbols as those in p.

It is clear that the algorithm satisfies these conditions in the second for loop and correctly reports exact matches in S_i.

Algorithm 1 ExtremeSimilarityAlgorithm

Input: S_R, p, $E[]$, $L(e_i)$, $M(e_i)$
  Let $\mu \leftarrow$ max number of errors between S_i and S_R
  Find all occurrence so p in S_R with at most m errors
  for each occurrence of p in S_R **do**
    let o_1 be the starting position in S_R of the occurrence of p
    let $d_1d_2d_3 \dots d_r$ be the position of the errors
    let $D \leftarrow d_1d_2d_3 \dots d_r$
    Find all occurrences of D in E
    for each occurrence of D in E **do**
      let $e_1e_2 \dots e_q$ be the positions of D in E
      if $L(e_i) \neq J$ where $J \leftarrow L(e_1)$ **then**
        reject
      end if
      if $M(e_i) = p_i$ **then**
        reject
      end if
    end for
    if $\neg(E[e_{q+1}] \leq o_1 + |p| \wedge L(e_{q+1}) = J)$ **then**
      reject
    end if
  end for
  return $L(d_i)$ of positions of p in S_R

5.1 Asymptotically Fast Algorithm

By Lemma 1 it will take $O(n)$ to find all occurrences of p in S_r with at most m errors as m is bounded by a constant factor.

The first loop executes at most n times, as that is the maximum possible matches, giving us another factor of $O(n)$.

For most of the second loop we can make use of an Aho Corasick automaton (Aho and Corasick, 1975) built for the candidate sequences of $c_1 \dots c_i$. This can be built in time proportional to the sum of the lengths of the patterns, as each pattern is of constant length it is proportional to the number of candidate sequences, which is α. It is now simple to find all occurrences of all c_j in E as we simply run E through the automaton reporting matches as we go. As E is bounded by the number of errors in total, finding all occurrences of the candidates will take $O(k + occ)$ in total, where occ is the number of occurrences reported. All executions of this step take a total time of $O(n + occ)$, but we have another factor of $O(\alpha)$ from the one off initialisation. Checking if an occurrence occurs entirely within one sequence can be handled entirely by carefully constructing the string E. Instead of just sticking all the differentiating sequences together, we can add an extra character, \$, between each differentiating sequence. Now an occurrence is only reported if it is entirely within one sequence and we can easily track which sequence we're currently checking. We must now check that the errors are the symbols from the pattern using the $M(e_i)$ array, this will take constant time per occurrence as we will check ≤ 10 positions so $O(occ)$ in total; finally we must check that the next error will not

effect the match, this is one operation and can be done in constant time. So all together we have a run time of $O(n + occ)$.

Theorem 2 *The algorithm correctly computes the occurrences of p in $S_1S_2 \dots S_k$ in $O(n + occ)$ time.*

5.2 Practical Algorithm

Although asymptotically efficient, the above algorithm may not be practical due to the overheads and memory requirements of building the automaton and from the auxilliary data structures used in the approximate matching algorithm. To account for this we present a practical $O(\frac{nk}{w})$ algorithm with the same scheme as the previous algorithm. In this algorithm we use bit operations to perform 'find all occurrences of D in E' in $O(\frac{k}{w})$ time by Lemma 2 using the SHIFT ADD(Baeza-Yates and Gonnet, 1992) algorithm. It should be noted that when there are only a few sequences the list of errors will fit in one machine word and this becomes $O(1)$ meaning the algorithm runs in linear time.

We also make use of a more memory efficient version of the Landau-Vishkin Algorithm(Castro Miranda and Ayala-Rincón, 2005) for approximate matching which uses a suffix array to reduce memory requirements.

Other than this the time taken is the same as the previous algorithm giving a total runtime of $O(\frac{nk}{w})$.

Theorem 3 *The algorithm correctly computes occurrences of p in $S_1S_2 \dots S_k$ in $O(\frac{nk}{w})$ time.*

Conclusion

In this paper, we describe both asymptotically efficient and practical algorithms for the EXTREME SIMILARITY SEQUENCING problem. We are currently in the process of implementing and performing experiments to determine the performance of these algorithms on real biological data. We expect that this data will show us that the practical algorithm works extremely well when the number of errors can fit within a small number of machine words and easily outperforms the naive solution.

Future work includes completing the implementation and testing, as well as working on the more complex open problems presented below with an aim to extend these techniques to problems which also take into account different types of distance measures such as Levenstein distance. In addition to this we plan on making use of more compressed datastructures to further reduce the memory requirements of our technique.

Open Problems

In this paper, we introduce algorithms for the EXTREME SIMILARITY SEQUENCING problem, but a number of related open problems remain.

High Similarity Sequencing: We are given the sequences S_1, $S_2, \ldots, S_k$ where $S_j = j_1 I_1 j_2 I_2 \ldots j_m I_m$ in other words each sequence is formed from a sequence of identical blocks $I_1, I_2, \ldots, I_k$ with differences between them $j_1, j_2, \ldots, j_m$

The goal is to keep one reference sequence together with the differences in order to answer queries. The reference sequence together with the errors would be stored in specially designed data sequences. There is some existing work in this area (Huang et al., 2010)

Complex Similarity Sequencing: We are given the sequences $S_1, S_2, \ldots, S_k$ where $S_j = j_1 I_1 j_2 I_2 \ldots j_m I_m$ where $S_1 = j_1 I_1 j_2 I_2 \ldots j_m I_m$ and s_j is a permutation of the blocks $I_1, I_2, \ldots, I_k$ with errors between them $j_1, j_2, \ldots, j_m$

The goal is to keep one reference sequence together with the differences and permutations in order to answer queries. The reference sequence, permutation and errors would be stored in specially designed data sequences.

References and Notes

Adjeroh, D., Zhang, Y., Mukherjee, A., M. Powell and Bell, T. (2002) 'DNA sequence compression using the Burrows Wheeler Transform', in *Proc. IEEE Bioinformatics Conference*, Stanford University, CA, 2002 pp.303

Altschul, S., Gish, W., Miller, W., Myers, E. and D. Lipman. 1990 'A basic local alignment search tool', *Journal of Molecular Biology*, Vol. 215 No. 3 pp.403 - 410

Aho, A. V., and Corasick, M. J. (1975) Fast pattern matching: An aid to bibliographic search. *Communications of the ACM*, Vol. 18 No. 6. pp.333340

Baeza-Yates, R and Gonnet, G. H. (1992) 'A new approach to text searching', *Communications of the ACM*, Vol. 35 No. 10. pp.74 - 82

Cao, X., Li, C. S. and Tung, K. H. A., (2005) 'Indexing DNA sequences using q-grams', in DASFAA:*Proceedings of the 10th International Conference on Database Systems for Advanced Applications* Beijing, China, pp.4 - 16

Emahazion, T., Feuk, L., Jobs, M., Sawyer, S. L., Fredman, D., St Clair, D., Prince, J. A., and Brookes, A., J. (2001) 'SNP association studies in Alzheimers disease highlight problems for complex disease analysis', *Trends in Genetics*, Vol. 17 No. 7 pp.407 - 413

Engle L. J., Simpson C. L. and Landers J. E. (2006) 'Using high-throughput SNP technologies to study cancer', *Oncogene*, Vol. 25 No. 11 pp.1594 - 1601

Farach, M. (1997) 'Optimal Suffix Tree Construction with Large Alphabets' *Proceeding FOCS '97 Proceedings of the 38th Annual Symposium on Foundations of Computer Science* pp.137

Ferragina, P. and Manzini, G. (2000) 'Opportunistic data structures with applications', in FOCS:*Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA

Frousios, K., and Iliopoulos, C. S., Mouchard, L., Pissis, S. P., and Tischler, G. (2010) 'REAL: an efficient REad ALigner for next generation sequencing reads' in BCB 2011: *Proceedings of the first ACM International Conference on Bioinformatics and Computational Biology*, ACM, Chicago, United States of America, pp.154 - 159

Golynski, A., Orlandi, A., Raman, R. and Rao, S. S. (2011) 'Optimal Indexes for Sparse Bit Vectors' *CoRR* [online]. Avaliable at http://arxiv.org/abs/1108.2157 (Accessed 15 February 2012)

Huang, S,. Lam, T. W., Sung, W. K., Tam, S. L. and Yiu, S. M. (2010) 'Indexing similar DNA sequences' in AAIM'10: *Proceedings of the 6th international conference on Algorithmic aspects in information and management*, Weihai, China, pp.180 - 190

Landau, G. M. and Vishkin, U. (1986) 'Introducing efficient parallelism into approximate string matching and a new serial algorithm' in STOC '86: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, Berkeley, California, United States pp.220 - 230

Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009) 'Ultrafast and memory-efficient alignment of short DNA sequences to the human genome', *Genome biology*, Vol. 10 No. 3 pp.R25+

Lippert, R. A. (2005) 'Space-efficient whole genome comparisons with Burrows-Wheeler', *Journal of computational biology*, Vol. 12, No. 4. pp. 407-415

Li, H. and Durbin, R. (2009) 'Fast and accurate short read alignment with Burrows-Wheeler transform', *Bioinformatics*, Vol. 25 No. 14 pp.1753 - 1760

Li, R., Yu, C., Li, Y., Lam, T. W., Yiu, S. M., Kristiansen, K. and Wang, J. (2009) 'SOAP2: an improved ultrafast tool for short read alignment', *Bioinformatics*, Vol. 25 No. 16 pp.1966 - 1967

Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., Berka, J., Braverman, M. S., Chen, Y., Chen, Z., Dewell, S. B., Du, L., Fierro, J. M., and Gomes, X. V., Godwin, B. C., He, W., Helgesen, S., Ho, C. H., Irzyk, G. P., Jando, S. C., Alenquer, M. L. I., Jarvie, T. P., Jirage, K. B., Kim, J., Knight, J. R., Lanza, J. R., Leamon, J. H., Lefkowitz, S. M., Lei, M., Li, J., Lohman, K. L., Lu, H., Makhijani, V. B., McDade, K. E., McKenna, M. P., Myers, E. W., Nickerson, E., Nobile, J. R., Plant, R., Puc, B. P., Ronan, M. T., Roth, G. T., Sarkis, G. J., Simons, J. F., Simpson, J. W., Srinivasan, M., Tartaro, K. R., Tomasz, A., Vogt, K. A., Volkmer, G. A., Wang, S. H., Wang, Y., Weiner, M. P., Yu, P., Begley, R. F., and Rothberg, J. M. (2005) 'Genome sequencing in microfabricated high-density picolitre reactors', *Nature*, Vol. 437 No. 7057 pp.376 - 380

Maxam, A. M. and Gilbert, W. (1977) 'A new method for sequencing DNA', *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 74 No. 2 pp.560 - 564

Miranda, R. C. and Ayala-Rincn, M. (2005) 'A Modification of the Landau-Vishkin Algorithm Computing Longest Common Extensions via Suffix Arrays' in *Advances in Bioinformatics and Computational Biology*, Springer Berlin / Heidelberg

National Center for Biotechnology Information (2012) 'Database of SNPs' [online], Avaliable at http://www.ncbi.nlm.nih.gov/projects/SNP/index.html (Accessed 15 February)

Ng, S. B., Buckingham, K. J., Lee, C., Bigham, A. W., Tabor, H. K., Dent, K. M., Huff, C. D., Shannon, P. T., Jabs, E. W., Nickerson, D. A., Shendure, J. and Bamshad, M. J. (2010) 'Exome sequencing identifies the cause of a mendelian disorder', *Nature Genetics*, Vol. 42 No. 1 pp.30 - 35

Ostergaard, P., Simpson, M. A., Brice, G., Mansour, S., Connell, F. C., Onoufriadis, A., Child, A. H., Hwang, J., Kalidas, K., Mortimer, P. S., Trembath, R. and Jeffery, S. (2010) 'Rapid identification of mutations in GJC2 in primary lymphoedema using whole exome sequencing combined with linkage analysis with delineation of the phenotype', *Journal of Medical Genetics*, Vol. 48 No. 4 pp.251 - 255

Pearson, W. P. (1990) 'Rapid and sensitive sequence comparison with FASTP and FASTA', *Methods in enzymology*, Vol. 183 pp.63 - 68

Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, C. A., Hutchison, C. A., Slocombe, P. M. and Smith, M. (1977a) 'Nucleotide sequence of bacteriophage phi X174 DNA', *Nature*, Vol. 265 No. 5596 pp.687 - 695

Sanger, F. and Coulson, A. R. (1975) 'A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase', *Journal of Molecular Biology*, Vol. 94 No. 3 pp.441 - 448

Sanger, F. and Coulson, A. R. (1977b) 'DNA sequencing with chain-terminating inhibitors', *Proceedings of the National Academy of Science of the United States of America*, Vol. 74 No. 12 pp.5463 - 5467

Shendure, J., Porreca, G. J., Reppas, N. B., Lin, X., McCutcheon, J. P., Rosenbaum, A. M., Wang, M. D., Zhang, K., Mitra, R. D. and Church, G. M. (2005) 'Accurate Multiplex Polony Sequencing of an Evolved Bacterial Genome', *Science*, Vol. 309 No. 5741 pp.1728 - 1732

Simpson, M. A., Irving, M. D., Asilmaz, E., Gray, M. J., Dafou, D., Elmslie, F. V., Mansour, S., Holder, S. E., Brain, C. E., Burton, B. K., Kim, K. H., Pauli, R. M., Aftimos, S., Stewart, H., Kim, C. A., Holder-Espinasse, M., Robertson, S. P., Drake, W. M. and Trembath, R. C. (2011) 'Mutations in NOTCH2 cause Hajdu-Cheney syndrome, a disorder of severe and progressive bone loss', *Nature Genetics*, Vol. 43 No. 4 pp.303 - 305

Stevens, R., Goble, C., Baker, P. and Brass, A. (2001) 'A classification of tasks in bioinformatics', *Bioinformatics*, Vol. 17 pp.180 - 1888