**CHAPTER-1**

**1.1 Smith–Waterman algorithm**

The **Smith–Waterman algorithm** performs local [sequence alignment](http://en.wikipedia.org/wiki/Sequence_alignment); that is, for determining similar regions between two strings or [nucleotide](http://en.wikipedia.org/wiki/Nucleotide_sequences) or [protein sequences](http://en.wikipedia.org/wiki/Protein_sequence). Instead of looking at the [total sequence](http://en.wikipedia.org/w/index.php?title=Global_sequence_alignment&action=edit&redlink=1), the Smith–Waterman algorithm compares segments of all possible lengths and [optimizes](http://en.wikipedia.org/wiki/Mathematical_optimization) the similarity measure.

## 1.2 Background

The algorithm was first proposed by [Temple F. Smith](http://en.wikipedia.org/wiki/Temple_F._Smith) and [Michael S. Waterman](http://en.wikipedia.org/wiki/Michael_S._Waterman) in 1981. Like the [Needleman–Wunsch algorithm](http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm), of which it is a variation, Smith–Waterman is a [dynamic programming](http://en.wikipedia.org/wiki/Dynamic_programming) algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the [substitution matrix](http://en.wikipedia.org/wiki/Substitution_matrix) and the [gap-scoring](http://en.wikipedia.org/wiki/Gap_penalty) scheme). The main difference to the [Needleman–Wunsch algorithm](http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm) is that negative scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. [Backtracking](http://en.wikipedia.org/wiki/Backtracking) starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment. One does not actually implement the algorithm as described because improved alternatives are now available that have better [scaling](http://en.wikipedia.org/wiki/Scalability) (Gotoh, 1982)  and are more accurate (Altschul and Erickson, 1986).

## 1.3 Algorithm explanation

A [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) H is built as follows:


H(i,0) = 0,\; 0\le i\le m



H(0,j) = 0,\; 0\le j\le n



\text{ if } a_i = b_j
 then 
w(a_i, b_j) = w\text{(match)}
 
\text{ or if } a_i \neq b_j
 then 
w(a_i, b_j) = w\text{(mismatch)}


H(i,j) = \max \begin{Bmatrix}
0  \\
H(i-1,j-1) + \ w(a_i,b_j) & \text{Match/Mismatch} \\
H(i-1,j) + \ w(a_i,-) & \text{Deletion} \\
H(i,j-1) + \ w(-,b_j) & \text{Insertion}
\end{Bmatrix}
,\; 1\le i\le m, 1\le j\le n


Where:

* a, b = Strings over the [Alphabet](http://en.wikipedia.org/wiki/Alphabet) \Sigma
* m = \text{length}(a)
* n = \text{length}(b)
* H(i,j) – is the maximum Similarity-Score between a suffix of a[1...i] and a suffix of b[1...j]
* w(c,d),\; c, d\in\Sigma\cup\{'-'\}, '–' is the [gap-scoring](http://en.wikipedia.org/wiki/Gap_penalty) scheme

## 1.4 Example

* Sequence 1 = ACACACTA
* Sequence 2 = AGCACACA
* *w*(match) = +2
* w(a,-) = w(-,b) = w(\text{mismatch}) = -1

H =
\begin{pmatrix}
 &-&A&C&A&C&A&C&T&A \\
-&0&0&0&0&0&0&0&0&0 \\
A&0&2&1&2&1&2&1&0&2 \\
G&0&1&1&1&1&1&1&0&1 \\
C&0&0&3&2&3&2&3&2&1 \\
A&0&2&2&5&4&5&4&3&4 \\
C&0&1&4&4&7&6&7&6&5 \\
A&0&2&3&6&6&9&8&7&8 \\
C&0&1&4&5&8&8&11&10&9 \\
A&0&2&3&6&7&10&10&10&12
\end{pmatrix}


For the example, we get:

Sequence 1 = A-CACACTA

Sequence 2 = AGCACAC-A

## CHAPTER-2

## 2.1 Motivation

One motivation for local alignment is the difficulty of obtaining correct alignments in regions of low similarity between distantly related biological sequences, because mutations have added too much 'noise' over evolutionary time to allow for a meaningful comparison of those regions. Local alignment avoids such regions altogether and focuses on those with a positive score, i.e. those with an evolutionary conserved signal of similarity. A prerequisite for local alignment is a negative expectation score. The expectation score is defined as the average score that the scoring system ([substitution matrix](http://en.wikipedia.org/wiki/Substitution_matrix) and gap penalties) would yield for a random sequence.

Another motivation for using local alignments is that there is a reliable statistical model (developed by Karlin and Altschul) for optimal local alignments. The alignment of unrelated sequences tends to produce optimal local alignment scores which follow an extreme value distribution. This property allows programs to produce an [expectation value](http://en.wikipedia.org/wiki/Expectation_value) for the optimal local alignment of two sequences, which is a measure of how often two unrelated sequences would produce an optimal local alignment whose score is greater than or equal to the observed score. Very low expectation values indicate that the two sequences in question might be [homologous](http://en.wikipedia.org/wiki/Homology_(biology)), meaning they might share a common ancestor.

The Smith–Waterman algorithm is fairly demanding of time: To align two sequences of lengths *m* and *n*, [*O*](http://en.wikipedia.org/wiki/Big_O_notation)*(mn)* time is required. Smith–Waterman local similarity scores can be calculated in O(m) (linear) space if only the optimal alignment needs to be found, but naive algorithms to produce the alignment require O(mn) space. A linear space strategy to find the best local alignment has been described. [BLAST](http://en.wikipedia.org/wiki/BLAST) and [FASTA](http://en.wikipedia.org/wiki/FASTA) reduce the amount of time required by identifying conserved regions using rapid lookup strategies, at the cost of exactness.

An implementation of the Smith–Waterman Algorithm, SSEARCH, is available in the [FASTA](http://en.wikipedia.org/wiki/FASTA) sequence analysis package from . This implementation includes [Altivec](http://en.wikipedia.org/wiki/Altivec) accelerated code for [PowerPC](http://en.wikipedia.org/wiki/PowerPC) G4 and G5 processors that speeds up comparisons 10–20-fold, using a modification of the Wozniak, 1997 approach,  and an SSE2 vectorization developed by Farrarmaking optimal protein [sequence database](http://en.wikipedia.org/wiki/Sequence_database) searches quite practical.

## CHAPTER-3

## 3.1 Accelerated versions

### 3.1.1 FPGA

[Cray](http://en.wikipedia.org/wiki/Cray) demonstrated acceleration of the Smith–Waterman algorithm using a [reconfigurable computing](http://en.wikipedia.org/wiki/Reconfigurable_computing) platform based on [FPGA](http://en.wikipedia.org/wiki/Field-programmable_gate_array) chips, with results showing up to 28x speed-up over standard microprocessor-based solutions. Another FPGA-based version of the Smith–Waterman algorithm shows FPGA (Virtex-4) speedups up to 100X  over a 2.2 GHz Opteron processor. The [Time Logic](http://www.timelogic.com/) De Cypher and Code Quest systems also accelerate Smith–Waterman and Framesearch using PCIe FPGA cards.

A 2011 Master's thesis includes an analysis of FPGA-based Smith–Waterman acceleration.

### 3.1.2 GPU

[Lawrence Livermore National Laboratory](http://en.wikipedia.org/wiki/Lawrence_Livermore_National_Laboratory) and the US Department of Energy's [Joint Genome Institute](http://en.wikipedia.org/wiki/Joint_Genome_Institute) implemented an accelerated version of Smith–Waterman local sequence alignment searches using [graphics processing units](http://en.wikipedia.org/wiki/Graphics_processing_units) (GPUs) with preliminary results showing a 2x speed-up over software implementations. A similar method has already been implemented in the Biofacet software since 1997, with the same speed-up factor.

Several [GPU](http://en.wikipedia.org/wiki/GPU) implementations of the algorithm in [NVIDIA](http://en.wikipedia.org/wiki/NVIDIA)'s [CUDA](http://en.wikipedia.org/wiki/CUDA) C platform are also available. When compared to the best known CPU implementation (using SIMD instructions on the x86 architecture), by Farrar, the performance tests of this solution using a single [NVidia GeForce 8800 GTX](http://en.wikipedia.org/wiki/GeForce_8_Series) card show a slight increase in performance for smaller sequences, but a slight decrease in performance for larger ones. However the same tests running on dual [NVidia GeForce 8800 GTX](http://en.wikipedia.org/wiki/GeForce_8_Series) cards are almost twice as fast as the Farrar implementation for all sequence sizes tested.

A newer GPU CUDA implementation of SW is now available that is faster than previous versions and also removes limitations on query lengths. See [CUDASW++](http://www.nvidia.com/object/swplusplus_on_tesla.html).

Eleven different SW implementations on CUDA have been reported, three of which report speedups of 30X.

### 3.1.3 SIMD

In 2000, a fast implementation of the Smith–Waterman algorithm using the SIMD technology available in [Intel](http://en.wikipedia.org/wiki/Intel) [Pentium](http://en.wikipedia.org/wiki/Pentium_(brand)) [MMX](http://en.wikipedia.org/wiki/MMX_(instruction_set)) processors and similar technology was described in a publication by Rognes and Seeberg. In contrast to the Wozniak (1997) approach, the new implementation was based on vectors parallel with the query sequence, not diagonal vectors. The company [Sencel Bioinformatics](http://www.sencel.com/) has applied for a patent covering this approach. Sencel is developing the software further and provides executables for academic use free of charge.

A [SSE2](http://en.wikipedia.org/wiki/SSE2) vectorization of the algorithm (Farrar, 2007) is now available providing an 8-16-fold speedup on Intel/AMD processors with SSE2 extensions. When running on Intel processor using the [Core micro architecture](http://en.wikipedia.org/wiki/Core_(microarchitecture)) the SSE2 implementation achieves a 20-fold increase. Farrar's SSE2 implementation is available as the SSEARCH program in the [FASTA](http://en.wikipedia.org/wiki/FASTA) sequence comparison package. The SSEARCH is included in the [European Bioinformatics Institute](http://en.wikipedia.org/wiki/European_Bioinformatics_Institute)'s suite of [similarity searching programs](http://www.ebi.ac.uk/Tools/SSS/).

Danish bioinformatics company [CLC bio](http://en.wikipedia.org/wiki/CLC_bio) has achieved speed-ups of close to 200 over standard software implementations with SSE2 on an Intel 2.17 GHz Core 2 Duo CPU, according to a [publicly available white paper](http://www.clccell.com/download.html).

Accelerated version of the Smith–Waterman algorithm, on [Intel](http://en.wikipedia.org/wiki/Intel) and [AMD](http://en.wikipedia.org/wiki/AMD) based Linux servers, is supported by the [GenCore 6](http://www.biocceleration.com/GenCore6-General.html) package, offered by  [Biocceleration](http://www.biocceleration.com/). Performance benchmarks of this software package show up to 10 fold speed acceleration relative to standard software implementation on the same processor.

Currently the only company in bioinformatics to offer both SSE and FPGA solutions accelerating Smith–Waterman, [CLC bio](http://en.wikipedia.org/wiki/CLC_bio) has achieved speed-ups of more than 110 over standard software implementations with [CLC Bioinformatics Cube](http://www.clccube.com/)

The fastest implementation of the algorithm on CPUs with [SSSE3](http://en.wikipedia.org/wiki/SSSE3) can be found the SWIPE software (Rognes, 2011), which is available under the [GNU Affero General Public License](http://en.wikipedia.org/wiki/GNU_Affero_General_Public_License). In parallel, this software compares residues from sixteen different database sequences to one query residue. Using a 375 residue query sequence a speed of 106 billion cell updates per second (GCUPS) was achieved on a dual Intel [Xeon](http://en.wikipedia.org/wiki/Xeon) X5650 six-core processor system, which is over six times more rapid than software based on Farrar's 'striped' approach. It is faster than [BLAST](http://en.wikipedia.org/wiki/BLAST) when using the BLOSUM50 matrix.

### 3.1.4 Cell Broadband Engine

In 2008, Farrardescribed a port of the Striped Smith–Watermanto the [Cell Broadband Engine](http://en.wikipedia.org/wiki/Cell_Broadband_Engine) and reported speeds of 32 and 12 GCUPS on an [IBM QS20 blade](http://en.wikipedia.org/wiki/IBM_BladeCenter#QS20) and a Sony [PlayStation 3](http://en.wikipedia.org/wiki/PlayStation_3), respectively.

**CHAPTER-4**

**4.1 Conclusions And Future Development**

Short-read alignment is thought to be the computing bottleneck of the analysis of new sequencing data. Fortunately, the active development of alignment algorithms opens this bottleneck even with the rapidly increasing throughput of sequencing machines. In a couple of years, however, long reads will dominate again and programs developed for short reads will not be applicable; long-read alignment and de novo assembly will become crucial.

In addition, while major sequencing centers have sufficient localized computing resources to analyze data at present, such resources are not available to small research groups, which hamper the application of new sequencing technologies. Even between major centers, data sharing in a large collaborative project such as the 1000 genomes project ([http://1000genomes.org](http://1000genomes.org/)) poses challenges. One possible solution to these problems might be cloud computing, with data uploaded and analyzed in a shared cloud. Several researchers have explored this approach, but establishing a cloud computing framework requires the efforts of the entire community. Furthermore, data transfer bottlenecks and leased storage have yet to be proved cost-effective for cloud computing.

Another trend of development is the simultaneous alignment against multiple genomes. Li et al. [2] have found the presence of extensive novel sequences absent from the human reference genome, which may lead to the loss of information when reads are aligned to a single genome. In the light of large-scale resequence projects such as the 1000 genomes project, the Drosophila population genomics project ([http://dpgp.org](http://dpgp.org/)) and the 1001 genomes project ([http://1001genomes.org](http://1001genomes.org/)), alignment against multiple genomes will become increasingly important. Several groups [3, 4] have pioneered in this direction; the proposal of unifying multi-genome alignment and de novo assembly with an assembly graph (Birney and Durbin, personal communication) is attractive, but how to apply the methods given genome-wide human data is yet to be solved in practice.

**CHAPTER-5**

**5.1 Refrence**

1. Langmead B, Schatz MC, Lin J, et al. Searching for SNPs with cloud computing. Genome Biol.2009;10(11):R134. [[PMC free article](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3091327/)] [[PubMed](http://www.ncbi.nlm.nih.gov/pubmed/19930550)]

2. Li R, Li Y, Zheng H, et al. Building the sequence map of the human pan-genome. Nat Biotechnol.2010;28:57–63. [[PubMed](http://www.ncbi.nlm.nih.gov/pubmed/19997067)]

3. Schneeberger K, Hagmann J, Ossowski S, et al. Simultaneous alignment of short reads against multiple genomes. Genome Biol. 2009;10:R98. [[PMC free article](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2768987/)] [[PubMed](http://www.ncbi.nlm.nih.gov/pubmed/19761611)]

4. Mäkinen V, Navarro G, Sirén J, et al. Storage and retrieval of individual genomes. In: Batzoglou S, editor. RECOMB, Lecture Notes in Computer Science, Vol. 5541. Berlin: Springer; 2009. pp. 121–37.

5. <https://www.google.com>

6. http://www.python.com