From BLAST to PatternHunter: the Development of Alignment Tools and Genomic Research

Wei Wei
National University of Singapore, 21 Lower Kent Ridge Road, Singapore 119077
raquelle70679@163.com

Keywords: BLAST, PatternHunter, spaced seeds, alignment

Abstract. Developed in 1990, BLAST remained a most widely used heuristic in homology search. Its concept and algorithm of using seed matches to trigger alignments have been constantly studied, adopted and improved throughout the years. In this paper, we examines the development of BLAST-like tools, focusing on BLAST and PatternHunter, by giving an account of the fundamental concept as well as the refinement process of the algorithms, reviewing some of the limitations and difficulties, and discussing the status of current research.

1. Introduction

The discovery of DNA and genetics has spawned a groundbreaking revolution in modern biology. With increasing number of genomes being completely sequenced, tools that are able to efficiently analyze the sequences are in great demand. Taking into account the fact that sequence structures of genes and proteins are conserved in nature, searching for regions of similarity among sequences through the method of alignment became one of the most important and widely adopted methods in computational molecular biology [1].

In 1976, the Smith-Waterman algorithm was introduced to find the optimal alignment between two sequences by comparing the entire sequences with a dynamic approach [2]. Though the method has 100% sensitivity, it is way too time-consuming when the database is big. With the size of the Genbank growing exponentially, scientists have been making attempts to develop heuristic replacements which are able to produce optimal or near optimal results within a short period of time.

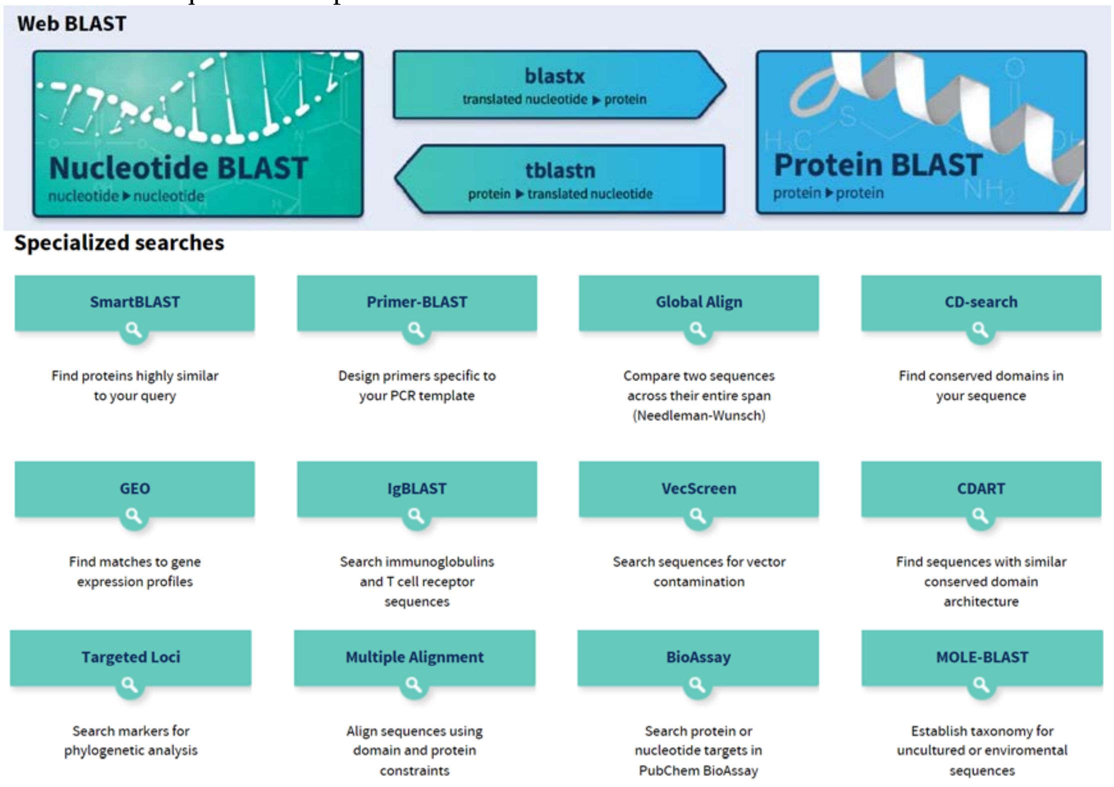
Over the years, various alignment tools have been developed and refined with the hope of achieving better sensitivity and efficiency. Some of such tools include FASTA (Lipman and Pearson, 1985), SIM (Huang and Miller, 1991), the Blast family (Altschul et al., 1990; Gish, 2001; Altschul et al., 1997, Zhang et al., 2000; Tatusova and Madden, 1999), PatternHunter (Ma et al., 2002) , and etc.[3][4].

In this paper, we discuss some of the algorithms mentioned above: the BLAST family and PatternHunter. The reason these two particular programs are chosen is because they are similar in nature: both make use of the method of searching for short matches (called seeds) between sequences and extending them into optimal alignments, with the rationale that high-scoring local alignments are likely to be found between two sequences if there are reasonably long exact matches between them [5]. By reviewing the development of the BLAST-like tools including PatternHunter, we will examine how a simple idea facilitates the progress of genomic search.

2. BLAST

2.1 BLAST– original version

The Basic Local Alignment Search Tool (BLAST) is the most widely used program in bioinformatics. Since the launch of the original version in 1990, a family of BLAST programs has been developed. The differences among these programs mainly lie in the input type and the databases that the input is searched against. For example, BLASTN is useful in searching a nucleotide sequence against a nucleotide sequence database, while BLASTP searches an amino acid sequence against a protein sequence database. Besides these, there are also the PSI-BLAST, MEGABLAST and etc. tailored to suit the different needs of sequence analysis [1]. Figure 1 shows some of the major BLAST programs available online. In this paper, we limit our discussion to mainly nucleotide sequence comparison.



Web BLAST

Nucleotide BLAST
nucleotide ▶ nucleotide

blastx
translated nucleotide ▶ protein

tblastn
protein ▶ translated nucleotide

Protein BLAST
protein ▶ protein

Specialized searches

SmartBLAST
Find proteins highly similar to your query

Primer-BLAST
Design primers specific to your PCR template

Global Align
Compare two sequences across their entire span (Needleman-Wunsch)

CD-search
Find conserved domains in your sequence

GEO
Find matches to gene expression profiles

IgBLAST
Search immunoglobulins and T cell receptor sequences

VecScreen
Search sequences for vector contamination

CDART
Find sequences with similar conserved domain architecture

Targeted Loci
Search markers for phylogenetic analysis

Multiple Alignment
Align sequences using domain and protein constraints

BioAssay
Search protein or nucleotide targets in PubChem BioAssay

MOLE-BLAST
Establish taxonomy for uncultured or enviromental sequences

Fig.1 Major BLAST programs available

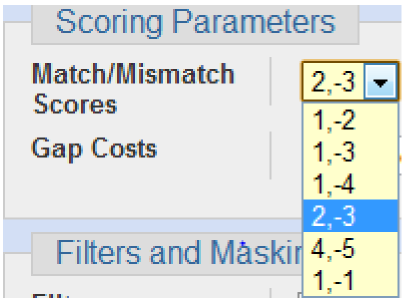
2.2 BLAST Algorithm

When aligning two sequences, there are three possible alignments in each single position: match, mismatch or gap (or sometimes called indel column which stands for insertion/deletion in terms of genomic mutation, which is the cost of gaps between two homologous sequences). When the two residues (nucleotides in the case of DNA sequences and amino acids in the case of protein sequences) in a position are the same, it is called a match. When the residues differ, it is called a mismatch. Gaps are inserted in between the residues to bring out matches further down the sequences. For example, aligning sequences ATTAGCC and ATAGCG, a possible alignment is as follows:

A T T A G C C
A T _ A G C G

In this case positions 1,2,4,5 and 6 correspond to matches. Position 7 is a mismatch while position 3 is a gap.

A scoring system is used to calculate the score of an alignment by assigning rewarding scores for matches and penalty scores for mismatches and gaps. In BLAST, a range of scoring options is provided for each type of program. Figures 2 and 3 show the options for scoring parameters for match/mismatch scores and gap costs respectively for BLASTN with the default settings highlighted in blue. Here, the gap cost is in the form of a penalty score for existence and another for extension. The existence cost (sometimes called gap opening cost) is applied to the starting position of a gap whenever a stretch of gaps appears in an alignment. For subsequent positions of consecutive gaps following the first position, the scores are calculated as the number of positions of gaps multiplied by the extension cost. If the existence cost is chosen to be 0, a gap will be penalized equally by the extension cost whenever it appears.

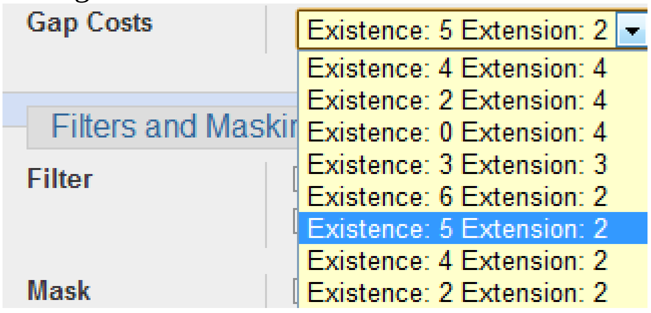

Scoring Parameters

Match/Mismatch Scores: 2,-3

Gap Costs

Filters and Maskir

Fig.2 Match/mismatch scores for BLASTN


Gap Costs: Existence: 5 Extension: 2

Filters and Maskir

Filter

Mask

Existence: 4 Extension: 4
Existence: 2 Extension: 4
Existence: 0 Extension: 4
Existence: 3 Extension: 3
Existence: 6 Extension: 2
Existence: 5 Extension: 2
Existence: 4 Extension: 2
Existence: 2 Extension: 2

Fig.3 Gap costs or BLASTN

The set of scoring parameters differ for different types of BLAST programs. A score for an alignment is calculated by summing all the scores for match/mismatch and gap penalties (negative scores). The alignment with the highest alignment score will be chosen as the optimal. Obviously, the choice of scoring parameters has an impact on the final optimal alignment(s) found.

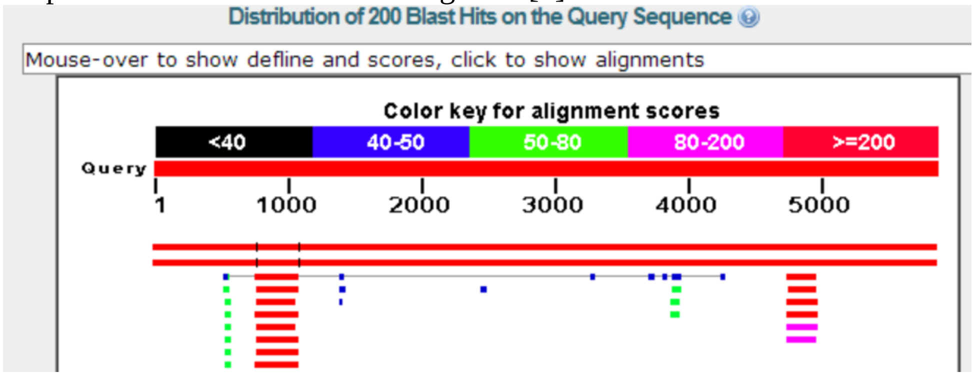
With the aid of the scoring system, a BLAST algorithm works in two steps: the filtration step and the extension step.

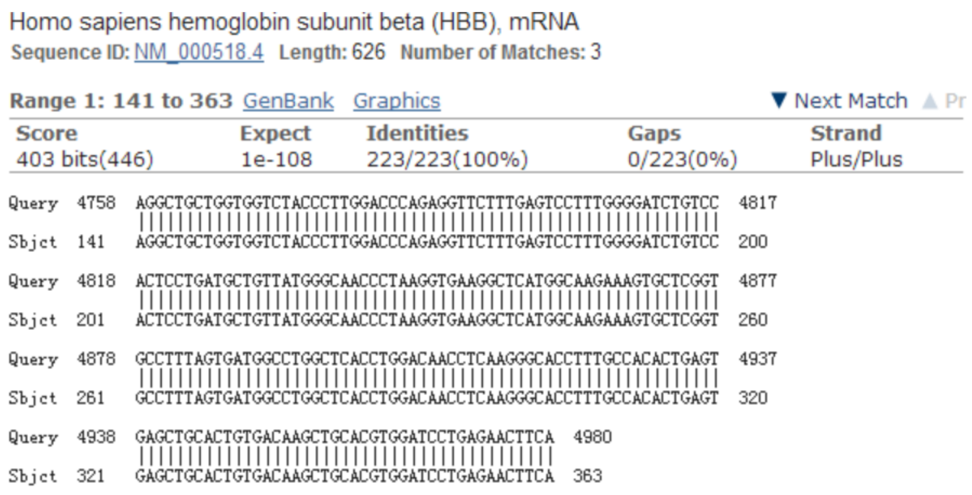
(1) Filtration

In this step, the database sequences (or target sequences) are scanned for consecutive matching pairs of length w (known as seeds) between the query and target sequences. The choice of w greatly affects the search result. When w is large, the sensitivity is low while the speed is fast. When w is small, it is the opposite. In the original version of BLAST, also known as ungapped BLAST, w is set by default to be 11 in BLASTN and 3 in BLASTP to balance between sensitivity and speed [1].

(2) Hit extension

In this step, each seed hit found in the previous step is extended in both ends to obtain a high-scoring segment pair (HSP) whose alignment score is higher than a threshold S. The extension stops once the alignment score drops more than X below the maximum sore that has attained up to that position. BLAST outputs a list o HSPs together with E-value that measures how frequent such HSPs would occur by chance. An example of the output is shown in Figure 4. The extension step is observed to take up more than 90% of the running time [1].


Distribution of 200 Blast Hits on the Query Sequence
Mouse-over to show defline and scores, click to show alignments
Color key for alignment scores
<40
40-50
50-80
80-200
>=200
Query
1
1000
2000
3000
4000
5000



```
Homo sapiens hemoglobin subunit beta (HBB), mRNA
Sequence ID: NM_000518.4  Length: 626  Number of Matches: 3

Range 1: 141 to 363  GenBank  Graphics                          ▼ Next Match  ▲ Pr
 Score              Expect     Identities           Gaps              Strand
 403 bits(446)      1e-108     223/223(100%)        0/223(0%)         Plus/Plus

Query  4758   AGGCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCC  4817
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  141    AGGCTGCTGGTGGTCTACCCTTGGACCCAGAGGTTCTTTGAGTCCTTTGGGGATCTGTCC  200

Query  4818   ACTCCTGATGCTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGT  4877
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  201    ACTCCTGATGCTGTTATGGGCAACCCTAAGGTGAAGGCTCATGGCAAGAAAGTGCTCGGT  260

Query  4878   GCCTTTAGTGATGGCCTGGCTCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGT  4937
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  261    GCCTTTAGTGATGGCCTGGCTCACCTGGACAACCTCAAGGGCACCTTTGCCACACTGAGT  320

Query  4938   GAGCTGCACTGTGACAAGCTGCACGTGGATCCTGAGAACTTCA  4980
              |||||||||||||||||||||||||||||||||||||||||||
Sbjct  321    GAGCTGCACTGTGACAAGCTGCACGTGGATCCTGAGAACTTCA  363
```

Fig.4 Parts of the output produced when searching G.gorilla beta-globin gene against human genomic + transcript using BLASTN.

The key challenge faced by this type of algorithms is the dilemma between sensitivity and specificity which is correlated to the size of seeds: large seeds tend to miss out alignments with significant similarity but do not contain the exact seed matches, while small seeds create too many random hits which slow down the computation [3]. To solve the problem, many improvements have been made on the basis of the algorithm explained above.

2.3 Gapped BLAST

This is a refined version of the original version of BLAST. The central idea adopted by this improved algorithm is the 'two-hit' method in which extension is triggered only when two non-overlapping hits within a distance A are detected. This algorithm is based upon the observation that an HSP of interest is usually much longer than a single word pair and often contains multiple hits within a relatively short distance of one another.

This key alteration improves the speed by an approximation of three times. The improvement comes from the fact that with the requirement of two hits to trigger an extension, fewer extensions are triggered, hence cutting down the running time on extension step. At the same time, the sensitivity of the algorithm is maintained by reducing the size of each individual seed [6].

3. PatternHunter

PatternHunter runs in a similar fashion as that of BLAST. The major difference lies in the choice of seeds: instead of using w consecutive letters as seeds, PatternHunter exploits nonconsecutive letters in the filtration step. This innovation was based on the observation made by Ma et al in 2002 that allowing spaces within a seed improves the sensitivity and speed tremendously [3].

A spaced seed used in PatternHunter can be represented by a string of 1's and 0's in which the 1's denote the positions that are required to match while 0's denote the positions where exact matches are not required (also called the 'don't care' positions). The number of 1's in a spaced seed is called weight. For instance, PatternHunter uses the optimal model of 111010010100110111 which has length 18 and weight 11. It was calculated by Ma et al that a spaced model 110100110010101111 has a significantly higher sensitivity when compared to consecutive models of weight 11 and 10. The reduced computing time is a result of reduced probability of random hits [3].

Introduced in 2003 by Li et al, PatternHunter II is the revised version of PatternHunter with the improvement of using multiple seeds [11]. The advantage of using multiple seeds is that it provides a better tradeoff between sensitivity and specificity which is negatively correlated to the false positive rate. According to Sun et al, adding seeds to a seeded alignment algorithm increases false positive rate at least linearly while reducing the seed weight produces an exponential increase [17]. Thus a wisely chosen set of multiple seeds will achieve sensitivity comparative to that of using a single seed but with much lower false positive rate. A proper tradeoff between the number and weight of the seeds allows the algorithm to achieve a sensitivity approaching that of the Smith-Waterman algorithm with a default BLASTN speed [11, 3].

The greatest concern of PatternHunter is probably the complexity in finding a set of optimal multiple seeds as well as assessing its sensitivity. In fact, even searching for a single optimal spaced seed is not easy. Given a string consisting of m 1’s and n 0’s, the total number of possible seeds with weight m is given by (m+n-2)!/(m-1)!(n-1)! (with the restriction that the string has to start and end with 1). Among them, some of the spaced seeds are even worse than consecutive seeds of the same weight (for instance, seeds in the form of (10k)m1, k,m>1 is worse than consecutive seeds of the same weight) [13]. One essential property is that a good spaced seed has very little internal periodicity [16]. Yet the strict and definite statements regarding the relative power of different seeds are yet to be established [8, 13].

Also, certain limitations of PatternHunter have been pointed out with solutions proposed. Chung et al pointed out that the seed adopted by PatternHunter (111010010100110111) was only optimal at the range of similarity levels from 61%-73% [4, 14] and that a new measure of sensitivity that covers a range of similarity levels of homologous regions is required. Their proposed model using a method called hit integration has found the seed 111001011001010111 to be more optimal than that of PatternHunter with about 2% higher sensitivity. Chen et al introduced the concept of ‘half gapped seed’, providing a solution to the difficulty in fine-tuning the tradeoff between sensitivity and efficiency in normal spaced seeds [12]. Illie et al suggested a new approach called overlap complexity in the effort to simplify the computation of sensitivity, stating that the sensitivity of a seed is inversely proportional to the number of overlapping hits [8].

4. Discussion

The concept of spaced seed which triggered the implementation of PatternHunter has attracted much interest even till today. Much research has been done on finding optimal spaced seeds and many interesting results have been produced since then. For instance, though it was proven both finding optimal seeds and computing hit probability of multiple seeds are NP-hard [13], Li et al came out with a greedy algorithm to find good spaced seeds and good results have been achieved using the algorithm [11]. Buhler et al pointed out that spaced seeds designed for aligning coding regions usually contain repeating ‘110’ patterns that ignore every third position [9]. The rationale may not be too difficult to comprehend: coding regions of DNA sequences tend to have synonymous substitutions that are frequently observed at the third positions of codons [10]. It is also interesting to note that the optimal spaced seed used in PatternHunter contains 3 out of its 6-codon span in the fifth reading frame: 11101001010011 0111, which perhaps might explain its effectiveness to some extent.

Though there are still many questions remain unanswered, spaced seed models have been adopted by other sequence analysis programs and new gene search programs with improved sensitivity and efficiency have been developed as a result. phRaider, a tool used in identifying transposable elements, incorporates the PatternHunter spaced seed model and hence achieves a 10-time speedup on human chromosomal sequences while improving sensitivity [7]. Exonhunter makes use of PatternHunter with coding region detection seed to enhance search results [15].

5. Conclusion

We have briefly looked through the development process of BLAST and PatternHunter. Interestingly, there are similarities even in their development paths in which a simple idea is constantly refined and inspired. PatternHunter was a great improvement on BLAST with the help of a simple but essential idea, yet PatternHunter II was built on the basis of the original PatternHunter in the fashion similar to how Gapped BLAST was built on the original version of BLAST. Though each step seemed trivial, cumulatively it has brought forth great advancement. Apparently, there is still a huge room for improvement for alignment tools such as BLAST and PatternHunter besides the unsolved mathematical questions mentioned above. So far, a majority of researches have focused on finding better ways to filter the sequences to improve the quality of search. However, the extension process still takes up most of the computing time, not to mention the memory constrain during scanning and indexing. In fact, in the event when the subject sequences are too large, PatternHunter II divides them into several smaller segments and search each of them in turn. As such, measures have to be taken to minimize the risk of losing alignments on the boundaries [11]. Yet perhaps in the future we will be able to avoid these problems altogether, with the advance in information technology and computer science.

It is also noteworthy that alignment-free sequence comparison methods have been gaining popularity in recent years as they are believed to circumvent some of the underlying constrains and difficulties that apply solely to methods using alignments [18, 19].

It is therefore completely possible that the methods of sequence alignment will eventually become part of the history as they are replaced by better algorithms and techniques. Nevertheless, the contributions that they have made in the field of bioinformatics and genomic studies will become a stepping stone for human beings to unveil the mysteries of life and remain as an inspiration for the generations to come.

References

[1] Heath, Lenwood S., and Naren Ramakrishnan, eds. *Problem Solving Handbook in Computational Biology and Bioinformatics*. No. 784. Springer Science & Business Media, 2010.

[2] Waterman, Michael S., Temple F. Smith, and William A. Beyer. "Some biological sequence metrics." *Advances in Mathematics* 20.3 (1976): 367-387.

[3] Ma, Bin, John Tromp, and Ming Li. "PatternHunter: faster and more sensitive homology search." *Bioinformatics* 18.3 (2002): 440-445.

[4] Choi, Kwok Pui, Fanfan Zeng, and Louxin Zhang. "Good spaced seeds for homology search." *Bioinformatics and Bioengineering, 2004. BIBE 2004. Proceedings. Fourth IEEE Symposium on*. IEEE, 2004.

[5] Choi, Kwok Pui, and Louxin Zhang. "Sensitivity analysis and efficient method for identifying optimal spaced seeds." *Journal of Computer and System Sciences* 68.1 (2004): 22-40.

[6] Altschul, Stephen F., et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." *Nucleic acids research*25.17 (1997): 3389-3402.

[7] Schaeffer, Carly E., et al. "phRAIDER: Pattern-Hunter based Rapid Ab Initio Detection of Elementary Repeats." *Bioinformatics* 32.12 (2016): i209-i215.

[8] Ilie, Lucian, and Silvana Ilie. "Multiple spaced seeds for homology search."*Bioinformatics* 23.22 (2007): 2969-2977.

[9] Buhler, Jeremy, Uri Keich, and Yanni Sun. "Designing seeds for similarity search in genomic DNA." *Proceedings of the seventh annual international conference on Research in computational molecular biology*. ACM, 2003.

[10] Misawa, Kazuharu, and Reiko F. Kikuno. "GeneWaltz–A new method for reducing the false positives of gene finding." *BioData mining* 3.1 (2010): 1.

[11] Li, Ming, et al. "PatternHunter II: Highly sensitive and fast homology search."*Journal of bioinformatics and computational biology* 2.03 (2004): 417-439.

[12] Chen, Wei, and Wing-kin Sung. "On half gapped seed." *Genome Informatics*14 (2003): 176-185.

[13] Li, Ming, Bin Ma, and Louxin Zhang. "Superiority and complexity of the spaced seeds." *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006.

[14] Chung, Won-Hyoung, and Seong-Bae Park. "Hit integration for identifying optimal spaced seeds." *BMC bioinformatics* 11.1 (2010): 1.

[15] Brejová, Broňa, et al. "ExonHunter: a comprehensive approach to gene finding." *Bioinformatics* 21.suppl 1 (2005): i57-i65.

[16] Brown, Daniel G. "A survey of seeding for sequence alignment."*Bioinformatics algorithms: techniques and applications* (2008): 126-152.

[17] Sun, Yanni, and Jeremy Buhler. "Designing multiple simultaneous seeds for DNA similarity search." *Journal of Computational Biology* 12.6 (2005): 847-861.
[18] Leimeister, Chris-Andre, et al. "Fast alignment-free sequence comparison using spaced-word frequencies." *Bioinformatics* (2014): btu177.
[19] Weitschek, Emanuel, et al. "Next generation sequencing reads comparison with an alignment-free distance." *BMC research notes* 7.1 (2014): 1.