Multisequence Alignment as a new tool for Network Traffic Analysis

Krzysztof Fabjański[1], Adam Kozakiewicz[1], Anna Felkner[1], Piotr Kijewski[1] and Tomasz Kruk[1]

[1] NASK, Research and Academic Computer Network: nask@nask.pl

Abstract
This article presents a multiple sequence alignment as a method used for problems of motif[1] finding [13] in network traffic collection. Based on multisequence alignment we will present two bioinformatics approaches for finding longest common subsequence (LCS) [14] of network traffic signatures collection. The article starts from presenting the description of pairwise alignment algorithms, goes through the examples of its implementation and then comes to the part related to bioinformatics methods. At the end, some preliminary results concerning Center Star method will be presented.

1 Introduction

When biologists discover a new gene, its function is not always apparent. The usual approach is to compare its structure with genes, whose function has already been identified. Comparison (so-called *alignment*) of biological sequences is a basis for bioinformatics, a science focused on theories, algorithms, computational techniques and statistical methods, with the goal of solving problems of biological data analysis. Bioinformatics draws inspiration from may other branches of science and techniques it produces have often interesting application outside of biology – including automatic voice and handwriting recognition, but also in computing systems security. This paper focuses on methods of defining the similarity between biological sequences and show, how similar methods can be applied to the problem of recognition and characterization of computer network threats.

Many modern Intrusion Detection Systems[2] are combined with systems for automated generation of network signatures such as *Honeycomb* [15]. Tool which joins functions of IDS and automated signature creation system is known as an Early Warning System[3]. Main problem concerning classification and identification of network flows is related to extraction of common regions from network signatures sets [17] as it is shown in the figure 1. This process is related to finding a longest common subsequence (LCS) [13] of more then two strings. Longest common subsequence is a special subset of characters of

[1] Subsequent occurrence of a region in any other sequences.
[2] A piece of software that detects and logs any network anomalies.
[3] Main function of EWS is network traffic classification and potentially prediction and identification of new Internet threats, as an example of such system we can consider *ARAKIS* [1].

```
GET /        HTTP
||| |        ||||
GET /a/a.HTM HTTP
||| |        ||||
GET /        HTTP/1.1
```

Figure 1. Longest common subsequence extraction.

the string S_k arranged exactly in the same order how they occur in the rest of sequences form the set S. Formally, subsequence of a string S_k of a given length l can be written as the sequence of individual characters $S_k(1)S_k(2)S_k(3)\cdots S_k(l)$, where $S(i)$ denotes the single character on position i in string S_k. The longest common subsequence of a set of strings is the common subsequence of the greatest possible length. Although, the task of finding LCS of more then two strings can be solved using the technique of dynamic programming [13, 22], the time complexity [11, 7] and space complexity of this solution is unacceptable if the number of input strings is greater then tree. Therefore the dynamic programing is used only as an element of bioinformatics methods, especially in alignment of two sequences.

2 Sequence alignment

Sequence alignment is one of the most important problems in computational biology. It is used for finding similarities in sets of sequences of DNA, RNA or amino acids. Apart from finding highly conserved subregions[4] in a set of strings, sequence alignment allows us to reconstruct the evolutionary history of a taxa[5].

2.1 Alignment of two strings

An alignment of two strings [12] S_1 and S_2 is obtained by insertion of spaces (gaps) into S_1 sequence and S_2, and then placing one of the sequence above the other. To give an example of alignment of two strings, we will use following sequences [18]:

- TAGTCCTCA
- TCCAGCCCCAGGA

After alignment they have following structure:

```
T--AGTCCTC-A---
TCCAG-CC-CCAGGA
```

Alignment of sequences allows us to find highly conserved subregions or embedded patterns[6]. In order to extract common subsequence from the above alignment, we have to construct the consensus string [13]. It is performed by extraction of the consensus characters from each column of the alignment and then by concatenation of extracted characters. In the place of a consecutive spaces in consensus string we are placing pipe

[4] Repetitive similar subsequences that occur in all strings from the set.
[5] A classification or group of organisms.
[6] Repetitive common subsequences that occur in all strings from the set.

symbols "—". As the result of those operation we will obtain the longest common subsequence of given input strings:

```
T|AG|CC|C|A
```

```
Global: GTGTACICCAVAV
        G--TAC-CCA-AV

Local:  GTGTACICC-AVAV
        --GTAC-CCAAV--
```

Figure 2. Global and local alignment.

We can distinguish two types of alignments:
- global alignment,
- local alignment.

Local alignment allows us to find small parts (local) of two sequences where there are some similarities. Moreover, it makes no assumption about the whole length of the sequence whether it should be similar or not. On the other hand, global alignment allows us to align entire sequences. Each of those methods have some advantages and drawbacks. Global alignment can be computed using Needleman-Wunsch [6] algorithm. This algorithm gives us an optimal global alignment among explored all possible alignments and chooses the one with the best score. Techniques related to scoring will be discussed later in this paper. The main drawback of this algorithm is that the short and highly similar subsequences may be missed in the alignment. It is caused due to the fact that the rest of the sequence outweighs the small and highly similar regions. An alternative to the global alignment algorithm is the Smith-Waterman [6] algorithm. This algorithm allows us to determine the longest subsequence of the two sequences. Both algorithms are widely used in alignment of DNA sequences and either global and local alignments can be implemented using dynamic programing. Exemplary alignment using global and local techniques is presented in figure 2.

2.2 Multiple sequence alignment

A multiple sequence alignment (MSA) [12] is a generalization of the pairwise alignment. Insertion of gaps is performed into each string so that resulting strings have equal length. An exemplary optimal multiple alignment of four protein sequences [3]:
- AQPILLLV
- ALRLL
- AKILLL
- CPPVLILV

can look as it is shown below:

```
AWPILLLV
ALR-LL--
AK-ILLL-
CPPVLILV
```

Of course, there might exist more then one optimal alignment. Problem occurs when we want to compute multiple alignment of many sequences (hundreds of sequences). In the figure 3 [22] there is an exemplary computation process of an alignment of three sequences. In fact, the whole problem of multiple sequence alignment is NP-complete.

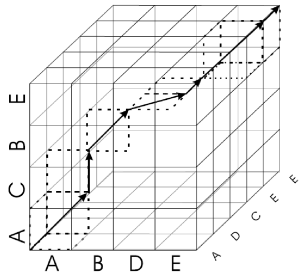

Figure 3. Dynamic programming for three sequences.

Thus, many heuristics, probabilistic and other approaches have been developed. We can divide those methods into eight groups[7] [22]:

- iterative algorithms (Realigner,PRRP,SAGA),
- progressive algorithms (ClustalW,TCoffe [20]),
- motif searching algorithms (Dialign, Blocks, eMotif),
- probabilistic methods (HMMs, Gibbs-Sampling),
- Divide-and-Conquer algorithms (DCA,OMA),
- exact algorithms (MSA,COSA,GSA).

In this paper we present two methods for common subsequence extraction from the given set of network signatures. First method uses an approximation algorithm and is known as Center Star method [2]. Second method implements the progressive approach and is known as ClustalW[8] method. This article covers detailed description of both methods. Both methods were slightly modified and adapted to our needs[9].

3 Sequence alignment and its implementation

Sequence alignment is a useful tool for network threat recognition in intrusion detection systems, automated threat signature generators and in malware analysis. To recognize a threat it is necessary to compare the new, observed behavior with previously identified or exemplary malicious behavior. The comparison is more useful, if it is not strict – this facilitates the detection of variants of attacks (behaviors) or attempts at masking the attack. Just like in bioinformatics problems, the main task is to find regularities, repeating similar sequences in large datasets.

[7] In parentheses there were given some exemplary algorithms which implements those techniques.
[8] ClustalW is a general purpose multiple sequence alignment program for DNA or proteins.
[9] Different methods for scoring and distance calculation were used.

3.1 Anomaly detection

One of the first applications of sequence alignment to intrusion detection is mentioned in [8] – with explicit connection to bioinformatics. The authors focused on the problem of detection of masquerading attempts, using logs of the `acct` tool in Unix systems as input data. Masquerading detection involves a *user signature* – a sequence of commands collected from the user, compared with the current session of this user. The main assumption is that an intruder using another user's account will behave in a different way than the rightful owner of the account, and that this difference of issued commands should be detectable. The paper proposes a method of aligning sequences of command from the current session with the user signature using a modified version of the Smith-Waterman algorithm. The result of the alignment, using a proposed scoring function, is used to detect an intrusion. In the opinion of the authors a local alignment would not be sufficiently effective, as a lot of potentially interesting data would be ignored. Therefore, the authors have used a *semi-global* alignment. In this type of alignment only the suffixes or prefixes of compared sequences are aligned. The scoring function rewards matching commands, but does not penalize the existence of large, non-matching parts of the signature. As in every anomaly detection method, an arbitrary threshold must be chosen to separate a suspected intrusion from a normal, but slightly atypical session of the original user – this threshold was found empirically. The best experimental results show a 75.8% intrusion detection level with 7.7% of false positives. A full description of the algorithm, the scoring function, data preparation method and analysis of results can be found in [8].

Another method of intrusion detection, popular in the literature but rarely implemented is system call monitoring – the system calls of processes are monitored and compared to typical behavior of a given type of processes. Differences in behavior could indicate a successful attack on the application, resulting in execution of potentially malicious code. A recently proposed extension of this idea is based on *evolutionary distance* between sequences, defined as the sum of costs of substitutions, deletions and insertions. Instead of creating a model of the behavior of the monitored process, a "replica" of the process is created and executed in parallel. A difference in behavior of the two processes may be a symptom of an attack. Since an effect of the same attack on two identical processes on identical platforms must, by definition, be the same, diversification of replicas is necessary. Thus, good candidates for a replica are the same process running on a different platform (e.g. Windows instead of Linux), or even a different process with the same functionality (e.g. a different WWW server) on a different platform. The authors of the idea assume that even though the processes will use different system calls, the function of those calls will be similar. It is possible to correlate different system calls from different processes/platforms. A description of the method of computing the behavioral distance between processes and of the experimental results can be found in the paper [9]. In the next paper [10] the authors used hidden Markov models for this task.

3.2 Threat signature generation

Bioinformatics are much more often mentioned in the literature on network threats in the context of threat signature generation systems. This area of research has gained a lot of attention in the recent years. New, unknown threats appear very often. They are initially not recognized by the traditional intrusion detection systems based on threat

signatures, since a signature hasn't been created yet. In this case a very useful tool is an automatic system, capable of recognizing a new threat and generating its signature, preferably without human intervention.

The first system to automatically generate threat signatures was `honeycomb` [16], a plug-in for `honeyd`. While the system itself was not based on bioinformatics methods, it did use some algorithms for detection of repeating similar sequences. The system applied the *Longest Common Substring*[10] algorithm to find common sequences of bytes in different packets sent to the system. As the system was a part of a honeypot, all incoming packets were by definition suspected to be part of an attack. Unfortunately, the system did not scale well in real honeypot networks, it generated a lot of repeating signatures and was completely useless against polymorphic attacks. Additionally, lack of implemented signature management methods meant that with time it was difficult to tell, which signatures (and attacks) are indeed new.

`Honeycomb` had many descendants, using different methods to recognize repeating sequences in the data stream, using them as the basis for signature generation. However, more advanced bioinformatics algorithms weren't proposed until polymorphic attacks were targeted. In a polymorphic attack there are, by definition, few constant substrings (subsequences without gaps), common among all instances of the attack. Furthermore, the longest such substring, if found, is not necessarily the best sequence describing the attack.

For many years identification of a polymorphic attack using signatures expressed as subsequences of the attack was thought impossible. Signature-based intrusion detection systems were expected to disappear soon. However, in paper [19] it was shown, that every polymorphic attack must contain constant, repeating values, allowing the attack to successfully exploit a given vulnerability. Some constants are also required to use a given protocol to communicate with the attacked application. Description of such an attack is, therefore, possible, although difficult – a good description of the attack is neither a single common substring, nor the longest common subsequence, which might contain too many random individual characters. A local alignment is necessary to find a common region in all variants of a polymorphic attack. This approach was suggested in the `polygraph` system. It is a signature generation system, using information from another system to identify suspect flows. Using a set of such flows a signature is created as a set of short separate character sequences. For example, a signature for the Apache-Knacker exploit was as follows (expressed as a regular expression):

```
GET .* HTTP/1.1\r\n.*:.* \r\nHost:.* \r\n.*:.*\r\nHost:.*\xFF\xBF.*\r\n
```

To find common characters for the flows the authors used a modified version of the Smith-Waterman algorithm. The modification included rewarding continuous alignment, since such signatures are less likely to cause false positives. Groups of characters were rewarded, while gaps were penalized, where gaps are not only the maximal length of subsequences matched with spaces, but also the maximal length of subsequences of non-matching characters. The penalties were selected so that character sequences were more

[10] A *different* term than Longest Common Subsequence

likely to be aligned if their grouping is typical for a given protocol. In application this would mean that different scoring functions for different protocols should be developed.

In the experiment, the system was tested on three real exploits – two for `httpd` servers (the Apache-Knacker exploit and the ATPhttpd exploit) and one for BIND server (the BIND-TSIG exploit). Clet, a well known tool for polymorphic attack generation was used. It was found, that Clet had many weaknesses – in each variant of the exploit many constant sequences were found. Since the goal of the experiment was to test the system with the assumption of nearly perfect polymorphism, the code of the exploit was manually changed using random values, leaving the sequences necessary for its functioning intact. The signature generator based on the modified Smith-Waterman algorithm produced the correct signature in all tests, giving 0.0008% false positives for Apache-Knacker, and 0% for the BIND-TSIG exploit – verified against a test pool of "proper" traffic. Results for the ATPhttpd exploit were not published. Only 3 samples of the exploit were necessary to reach such a high level of precision. The generated signatures can be used in many modern intrusion detection systems like `snort`.

Another approach to polymorphic worm detection was used in [24]. Like in the previous case, common regions were searched for – using *Gibbs sampling* and creating signatures based on the frequencies of individual characters. Gibbs sampling is also used in bioinformatics to find *motifs* – unchanged by evolution regions in protein sequences.

4 Center Star method

As it was mentioned before, multiple sequence alignment is an NP-complete problem. Presented method, Center Star, is an approximation way of multisequence alignment. Thus, expected result can be, but does not have to be, an optimal multiple sequence alignment. The Center Star method consists of three main steps. Before the three main steps of the Center Star algorithm will be presented, there will be given the definition of a distance between two sequences S_1 and S_2. In order to calculate the distance of two sequences we have to compute the value of alignment of two sequences S_1 and S_2[11]. Usually, if we want to calculate the score of an alignment we have to define the scoring matrix[12] for each individual character from the alphabet. In this case $\sum' = \{A, G, C, T, \_\}$ [18].

d	A	G	C	T	_
A	1	0	0	0	0
G		1	0	0	0
C			1	0	0
T				1	0
_					0

Table 1. Scoring matrix for an alphabet $\sum' = \{A, G, C, T, \_\}$.

Concerning two sequences S_1 and S_2 and the same sequences but with inserted spaces S_1' and S_2' (alignment representation from section 2.1) of the length l we can easily

[11] We will use the example shown in the section 2.1.

[12] Sometimes we call it scoring scheme, scoring function or objective function.

calculate the value of alignment of S_1 and S_2 as a $D(S_1, S_2) = \sum_{i=1}^{l} d(S_1'(i), S_2'(i))$. In order to calculate the distance between two sequences S_1 and S_2 we have to calculate the difference $dist(S_1, S_2) = l - D(S_1, S_2)$. In bioinformatics we can meet different scoring schemes as well as different methods for distance calculation[13].

Three steps of the Center Star method are as following:

1. Given a set of sequences S, find the Center sequence $S_c \in S$ whose sum of pairwise distances [21] to all other sequences is minimal $min(\sum_{S_j \in S} dist(S_c, S_j)$. Note that during this step we are computing all pairwise alignments[14] in the set of sequences S.
2. Having the Center sequence S_c chosen, create the multiple sequence alignment by adding sequences one by one. Always align optimally with the current Center sequence S_c.
3. Having the multiple sequence aligned, extract the consensus sequence from it and replace all gaps with pipe symbols in order to mark separate subsequences occurrence, exactly as it was shown in section 2.1.

Concerning the second step of this method, the insertion of gaps is performed according to specified rules. Positions of the gaps that were inserted during early alignments are not changed as new sequences are added. On the other hand, adding gaps to pre-aligned sequences is performed if needed. In the figure 4 consecutive steps of Center Star method are presented [2].

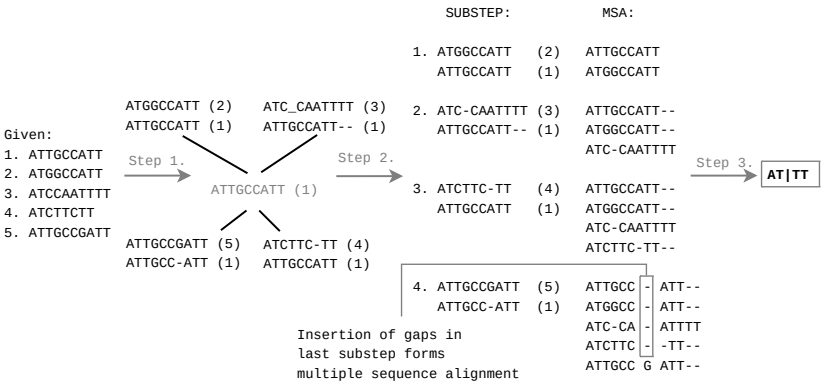

Figure 4. Presentation of an exemplary multiple sequence alignment using Center Star method.

5 ClustalW method

ClustalW method represents the progressive approach in finding multiple sequence alignment. This method consists of four main steps:

1. First step is exactly the same as for Center Star method. Using Needleman-Wunsch algorithm for finding an exact pairwise alignment (see section 2.1) of two sequences

[13]The other way of calculating the distance is to divide the number of mismatches on non-gapped positions by the number of non-gapped pairs.

[14]Pairwise alignment can be computed using Needleman-Wunsch algorithm.

we are computing the distance between all sequences from the set S. In this step we are forming the distance matrix[15] between all pairs of sequences from the set S.

2. Using Neighbor-Joining algorithm [23, 5, 25, 3] we designate the neighbor-joining tree[16]. Having a guide tree computed we can calculate the pairwise alignments in the order designated by this tree. Each alignment (sequence-sequence, sequence-profile[17], profile-profile) involves dynamic programming by the sum of pairs score method.

3. The next step of this method is similar to the Center Star algorithm. Combining the alignments starts from the most closely related group to the most distantly related groups by going from the tip of similarity tree to the root of the tree. During combining process, a rule "Once a gap, always a gap"[18] is preserved.

4. The last step is related to extraction of the consensus string and substitution of gaps with pipe symbols from the computed multiple sequence alignment.

As an example of this ClustalW method we can refer to the figure 5.

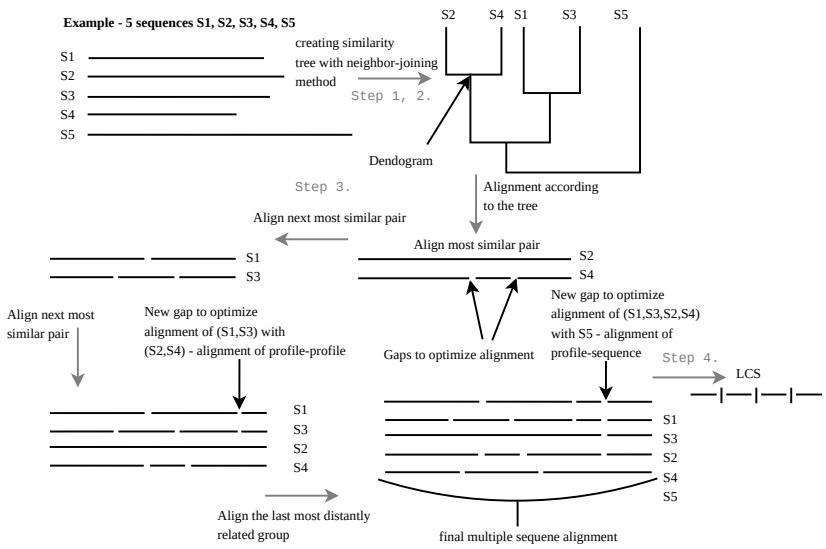

Figure 5. Presentation of an exemplary multiple sequence alignment using the ClustalW method.

[15] Sometimes called similarity matrix.

[16] Unrooted tree, when root is placed at the midpoint of the longest chain of consecutive edges we will obtain a guide tree or similarity tree.

[17] Profile for multiple alignment specifies for each column the frequency that each character appears in the column [13].

[18] That is, once a gap is entered into the alignment, it will always remain in that alignment, and all sequences added subsequently will also receive that gap.

6 Evaluation

Both presented methods give as a result the multiple sequence alignment. Both methods do not guarantee that the multiple sequence alignment obtained during calculation will be an optimal solution. Center Star method gives at most twice the score of the optimal multiple alignment of the set of sequences S [13, 2]. Complexity of both methods is approximately the same. For both methods first step of the algorithms requires $\binom{j}{2}O(n^2)$ time, where j is the number of sequences in the set S. Step 2 and 3 of the Center Star method requires $O(j^2 l)$ time, assuming that l is an upperbound on the length of the sequences in the MSA. Time complexity of the steps 2, 3 and 4 of the ClustalW method mainly depends on the way of construction of the guide tree. Time complexity of Neighbor-Joining algorithm is $O(n^2)$ [4]. The ClustalW method, apart from giving the multiple sequence alignment, provides also the clusterization mechanism. The guide tree computed during 2 step of the ClustalW algorithm provides us the mechanism for hierarchical clusterization. Concerning the figure 5 and the similarity tree, we can treat it as a dendogram where internal nodes (profiles) represent the sequences from the set S. Going higher and higher in the hierarchy of the dendogram we are grouping sequences into bigger and bigger collections.

6.1 Preliminary results of the Center Star method

The Center Star method was fully implemented in C++. In order to implement the distance matrix the Standard Template Library's (v3.3) vector of vector class was used. All tests where performed on real dataset taken form the Arakis database. A distance between two sequences was calculated using formula described in section 4 (number of mismatches on non-gapped positions was divided by the number of non-gapped pairs). All results are presented in table 2. Tests where executed on the Intel(R) Xeon(TM) CPU 3.00GHz with 2075808 kB of the total memory. Compiler used for compilation was g++ (v4.1).

Number/Length	*DM time*	*LCS time*	*Total time*
72/600	262.83	0.01	262.84
6/2500	17.53	0.00	17.53
304/300	1150.36	0.04	1150.40
18/300	6.07	0.00	6.07
50/450	64.58	0.00	64.58

Table 2. Preliminary results of the Center Star method.

First column of the table presents input dataset, number of signatures and their average length. Second column gives results concerning time of the process of distance matrix creation (DM time). The third column shows run-time of the last phase of the Central Star method. It gives us time of common subsequence extraction from the multiple sequence alignment. The last column presents total run-time of the Center Star method. All results are given in seconds. As it is shown, run-time complexity of the Center Star method heavily depends on the process of distance matrix creation. The greater the number of input signature, the greater run-time of the algorithm. The same is with average length of the input signatures. The greater the length of the signatures,

the greater time required to compute consecutive pairwise alignment and in consequences greater the run-time of the distance matrix creation process. Quality of the extracted common subsequence was fair. Extracted common subsequence was compared with the super signature (pseudo longest common subsequence) from the Arakis database. In some cases, the quality of the extracted common subsequence obtained with Center Star method was even better then quality of the super signature.

7 Conclusion

To sum up, ClustalW and Center Star algorithms have some advantages and disadvantages. One of the biggest drawback of both algorithms is that the run-time complexity is very high. On the other hand, the whole task is an NP-complete problem, so we cannot expect better run-time complexity. ClustalW as well as Center Star method can be modified in order to decrease the run-time complexity. In Center Star method instead of finding all pairwise alignment, we can take a randomly selected sequence from the set S and compute all pairwise alignments of it with the rest of sequences. As a result we would omit the process of choosing the Center sequence, which involves computation of all pairwise alignments in the set of network signatures. This improvement leads to the better time complexity, but on the other hand it will result in worse common subsequence extraction. In our case better time complexity is more important than worse common subsequence extraction. Extraction of the common subsequence during pre-processing phase should be performed in online mode. On the other hand clusterization of already created signatures (process of common subsequence extraction) might be performed in offline mode.

Bibliography

[1] Arakis. www.arakis.pl.

[2] Bioinformatics multiple sequence alignment. homepages.inf.ed.ac.uk/fgeerts/course/msa.pdf.

[3] Multiple alignment: heuristics. www.bscbioinformatics.com/Stu/Dbq/clustalW.pdf.

[4] Neighbor joining. http://www.cs.tau.ac.il/~rshamir/algmb/98/scribe/html/lec09/node23.html.

[5] The neighbor-joining method. http://www.icp.ucl.ac.be/~opperd/private/neighbor.html.

[6] Sequence alignment. http://helix.biology.mcmaster.ca/721/outline2/node37.html.

[7] Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *String Processing Information Retrieval, 7th International Symposium, SPIRE'00, La Coruna, Spain, 27-29 September 2000, Proceedings*, pages 39–48, Washington, DC, 2000. IEEE Computer Society.

[8] Scott Coull, Joel Branch, Boleslaw Szymanski, and Eric Breimer. Intrusion detection: A bioinformatics approach. In *Proceedings of the 19th Annual Computer Security Applications Conference*, page 24, Washington, DC, USA, 2003. IEEE Computer Society.

[9] Debin Gao, Michael K. Reiter, and Dawn Song. Behavioral distance for intrusion detection. In *In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID 2005)*, 2005.

[10] Debin Gao, Michael K. Reiter, and Dawn Song. Behavioral distance measurement using hidden markov models. In *In Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, 2006.

[11] Ronald I. Greenberg. Bounds on the number of longest common subsequences, 2003.

[12] D. Gusfield. Efficient method for multiple sequence alignment with guaranteed error bounds. Report CSE-91-4, Computer Science Division, University of California, Davis, 1991.

[13] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[14] Dan Hirschberg. A linear space algorithm for computing common subsequences. *Communication of the ACM*, 18:341–343, 1975.

[15] Christian Kreibich. Honeycomb. Automated signature creation using honeypots - http://www.icir.org/christian/honeycomb/index.html.

[16] Christian Kreibich and Jon Crowcroft. Honeycomb - creating intrusion detection signatures using honeypots. In *Proceedings of the Second Workshop on Hot Topics in Networks (Hotnets II). Cambridge Massachusetts: ACM SIGCOMM*, Boston, November 2003.

[17] Christian Kreibich and Jon Crowcroft. Efficient sequence alignment of network traffic. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 307–312, New York, NY, USA, 2006. ACM Press.

[18] Paolo Pin Matteo Barigozzi. Multiple string alignment. 2003.

[19] James Newsome, Brad Karp, and Dawn Song. Polygraph - automatically generating signatures for polymorphic worms. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 226–241, Washington, DC, USA, 2005. IEEE Computer Society.

[20] C. Notredame, D.G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–17, 2000.

[21] S. W. Perrey, J. Stoye, V. Moulton, and A. W. M. Dress. On simultaneous versus iterative multiple sequence alignment. Materialien/Preprints 111, Universität Bielefeld, Forschungsschwerpunkt Mathematisierung – Strukturbildungsprozesse, 1997.

[22] Knut Reinert. Introduction to multiple sequence alignment. *Algorithmische Bioinformatik WS 03*, pages 1–30, 2005.

[23] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–25, 1987.

[24] Yong Tang and Shigang Chen. Defending aagainst internet worms: A signature-based approach. In *Proceedings of the 24th Annual Conference IEEE INFOCOM 2005*, March 2005.

[25] Zhiping Weng. Protein and dna sequence analisys be561, 2005. Boston University.