JMSCR Vol||05||Issue||03||Page 19388-19394||March 2017

www.jmscr.igmpublication.org
Impact Factor 5.84
Index Copernicus Value: 83.27
ISSN (e)-2347-176x ISSN (p) 2455-0450
crossref DOI: https://dx.doi.org/10.18535/jmscr/v5i3.168

IGM Publication
Journal Of Medical Science And Clinical Research
An Official Publication Of IGM Publication

Comparison between Hirschberg's algorithm and Needleman-Wunsch algorithm in finding optimal alignment in terms of search space and time complexity

Authors
Fahad Almsned, Arwa Alhamed
School of Systems Biology, George Mason University, Fairfax, VA 22030

Background

DNA sequences, which are the data representation in this work, are not neatly arranged sequences that store organisms genetic. Specifically, the information is encoded using four key chemicals, adenine, thymine, guanine and cytosine (abbreviated as A, T, G and C)[1]. This biological sequence data can be obtained from variety of public and private databases. With the growing amount of data, it became impractical to analyze DNA sequences manually, so faster algorithms and tools are needed. Sequence analysis is the process used to find information about a nucleotide or amino acid sequence using computational methods[2]. Sequence comparison, which is the fundamental procedure of analyzing sequence content, is regarded as one of the most fundamental problems of computational biology that usually solved with a technique known as sequence alignment. Sequence alignment can be defined as the problem of finding, which parts of the sequences are similar and which parts are different. Sequence alignment is a computationally challenging problem of paramount importance and is a fundamental operation performed in computational biology research. The goal is to produce the best alignment for a pair of DNA or protein sequences (represented as strings of characters). A good alignment has zero or more gaps inserted into the sequences to maximize the number of positions in the aligned strings that match. For example, consider aligning the sequences ATTGGC and AGGAC. By inserting gaps (-) in the appropriate place, the number of positions where the two sequences agree can be maximized: ATTGG-C and AGGAC[3]. The sequence alignment problem can be illustrated as, given a scoring function that measures the score of aligning characters at the same position from each sequence, calculate the total score of the alignment by adding the scores of all positions and find the maximum total score of every possible alignment.

One of the main problems in the comparison of sequences of biological data is an effort to determine their degree of similarity. Accordingly, many available algorithms and techniques in solving the problems of sequence alignment. There are many algorithms that maximize speed and do not concern with the accuracy of the result

alignment. In contrary, there are many algorithms that maximize accuracy and do not concern with the speed. Most current sequence comparison methods used in practice, such as, BLAST and FASTA are based on Heuristics, which are much faster, but do not provide optimal results. There are many algorithms written that use the approach of Dynamic Programming. However, Needleman-Wunsch algorithm was the first to introduce Dynamic Programming to compare biological sequences for finding the global alignment between two sequences[4]. Global sequence alignment is comparing the sequences entirely and it is most useful when the sequences in the query set are similar and of roughly equal size. The Needleman-Wunsch algorithm consists of three steps: first, initialization of the dot plot, score and the traceback matrices, second, calculation of scores and filling in the score and traceback matrices, third, deducing the alignment from the traceback matrix. When the steps were implemented there are three matrices produced, the dot plot matrix, the score matrix and the traceback matrix[4].

To improve the original algorithm performance, few modified forms have been proposed. These forms supposedly are faster and less computationally expensive without compromising the quality of the results. Hirschberg's algorithm, which is an improved version of Needleman-Wunsch algorithm, considers one of the improved algorithms to find optimal alignment[5]. While the Needleman-Wunsch algorithm works well for sequence alignment, its space complexity (O(mn)) limits the size of sequences it can align. Hirschberg'salgorithm uses a divide and conquers strategy to decrease the space requirement. Specifically, for two sequences (m and n) the first string is cut (m1 and m2) and the second string is cut in a corresponding place (into n1 and n2). The alignment is then solved recursively on m1 and n1, and m2 and n2. It is important to note that the two sub-strings (i.e. n1 and n2) need not have the same length.

In this work, we will evaluate and compare the performance of Hirschberg's algorithm and the original Needleman-Wunsch algorithm in finding optimal alignment in terms of search space and time complexity without compromising the accuracy and efficiency when analyzing large sequences. This work only focuses on similarity, as it is the preferred choice for biological applications.

Methods

The main aim of this study is to evaluate Hirschberg's algorithm in terms of search space and time complexity by comparing it to the original Needleman-Wunsch. The main variable is pairwise alignment execution time in milliseconds, which is used to evaluate the performance of both algorithms The lesser the time, the better the performance. Other variables are the match and gap scores, which are used to test if the algorithms are producing the same alignments; hence, they are theoretically comparable.

The variables are generated by aligning a fixed sequence (Gen Bank accession ID: EF445041) against other sequences. The sequence alignment is carried out in pairwise fashion using both algorithms; Needleman-Wunsch algorithm (NW), and Hirschberg's algorithm. The sequences will be obtained from National Center for Biotechnology Information (NCBI) Nucleotide Database. The sequences are randomly selected from a list containing Homo Sapiens genomic DNA sequences, which are 190 to 200 bp in length. For each algorithm, execution time, number of matches, and gaps will be reported and documented in separate excel sheet. Then, appropriate R statistical tests are done on both methods results. Python version 2.7.0 has been used to code both algorithms. One laptop has been used for both codes, to reduce hardware biases, with the following specifications: 3.1 GHz Intel Core i7 processor, 16 GB 1866 MHz DDR3 Memory, Intel Iris HD Graphics 6100 1024 MB GPU, macOS Sierra Operating System version

12.12.1. R version 3.2.3 is used to conduct the appropriate statistical tests.

Statistical Analysis

For each algorithm execution time in milliseconds, number of matches, and number of gaps are reported and documented in a separate excel sheet. Data from 50 sequences has been used to evaluate feasibility and generate the preliminary data. Descriptive statistics of the main variable, execution time in milliseconds, has been used to calculate the appropriate sample size, which turned to be 127 in each arm assuming that we are looking for at least 5 milliseconds difference between the 2 algorithms (80% power and 5% significant level). Sample size has been calculated using the following R code:

```
power.t.test( NULL , 5 , 14.15284 , .05 , .8)

##
##      Two-sample t test power calculation
##
##               n = 126.7398
##           delta = 5
##              sd = 14.15284
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

For each variable, the data has been visualized using histograms, boxplots, and Quantile-Quantile plots to evaluate normality of the distribution and presence of outliers. Then, means, medians, and standard deviations have been calculated. Welch Two Sample t-test has been used to compare execution time in milliseconds means difference between the two algorithms. Pearson‘s correlation has been used to test the strength of the linear relationship between the 2 other variables, match and gap scores, between the two algorithms.

Results

```
match=NW[1:127,2]
gap=NW[1:127,3]
time= NW[1:127,4]*1000

par(mfrow=c(3,3))

hist(match, col=4, main = "NW Match Score")
hist(gap, col=5, main = "NW Gap Score")
hist(time, col=3, main = "NW Execution Time in ms")

boxplot(match, col=4,main = "NW Match Score")
boxplot(gap, col=5, main = "NW Gap Score")
boxplot(time, col=3, main = "NW Execution Time in ms")

qqnorm(match,pch=20,cex.lab=1.5,main = "NW Match Score",
       ylab="Time", col=4)
qqline(match, col="red")

qqnorm(gap,pch=20,cex.lab=1.5,main = "NW Gap Score",
       ylab="Time", col="red")
qqline(gap, col="blue")

qqnorm(time,pch=20,cex.lab=1.5,main="NW Execution Time in ms",
       ylab="Time", col="red")
qqline(time, col="blue")
```

For Needleman-Wunsch algorithm, gap and match score data are normally distributed with slimier histogram patterns, but execution time data is slightly skewed to the left. No gaps in all histograms. In all box plots, the median is almost in the middle of the first quartile (Q1) and the third quartile (Q3), which support normality of then data. Except for execution time. Also, we have few to no outliers in all datasets. The Q-Q plots show that normality is probably a reasonably good approximation except for execution time where we have a light left tail when the time fall under 200 ms. We will assume normal distribution of all data sets (Figure1).

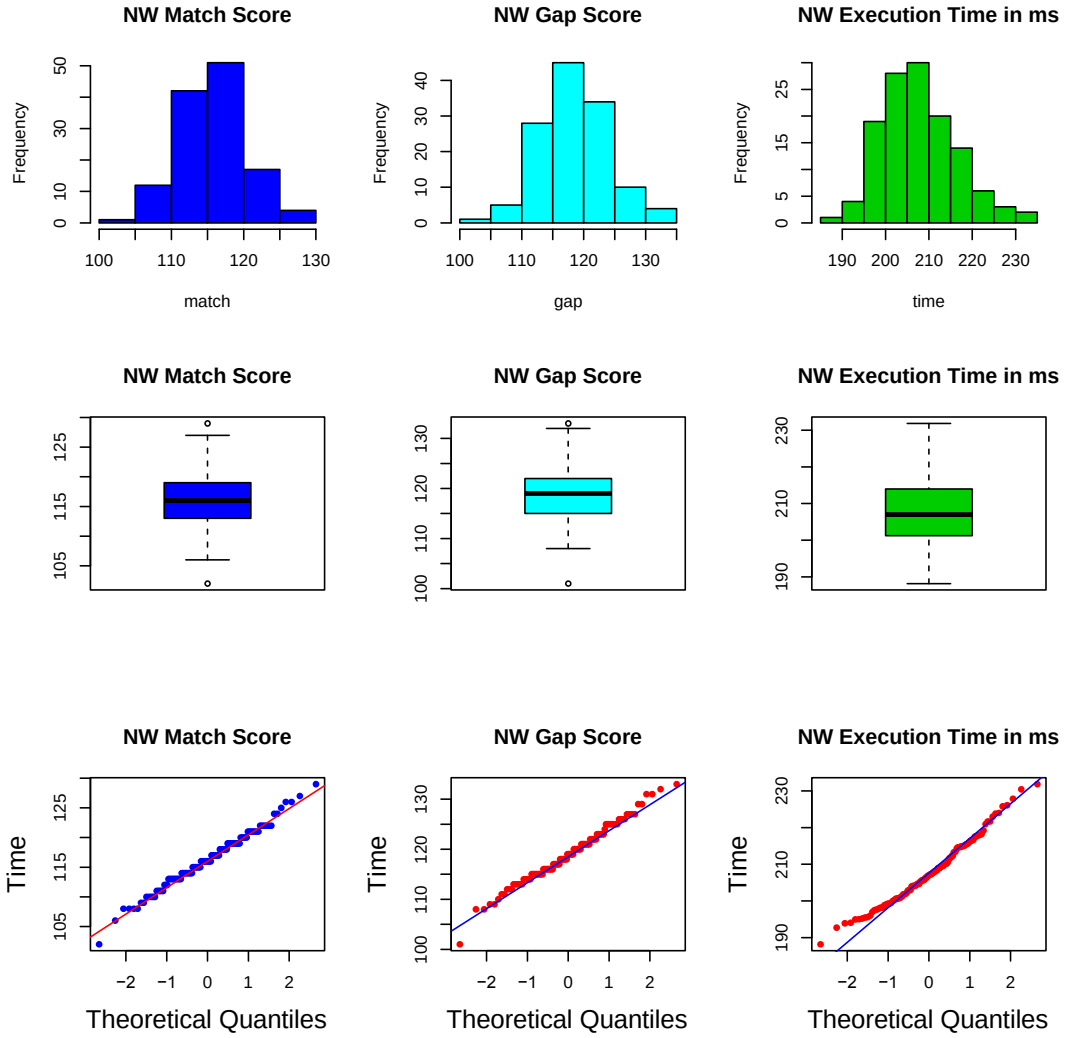

Figure 1: Descriptive statistics for Needleman-Wunsch's algorithm; Gap Score, Match score, and Execution time in ms.

```
mean(match);median(match);sd(match)

## [1] 116.2283
## [1] 116
## [1] 4.486645

mean(gap);median(gap);sd(gap)

## [1] 118.9528
## [1] 119
## [1] 5.40555

mean(time);median(time);sd(time)

## [1] 207.7391
## [1] 206.9991
## [1] 8.552902
```

The mean, median, and SD of match score are (116.2283, 116, and 4.486645 respectively), the mean, median, and SD of gap score are (118.9528, 119, and 5.40555 respectively), and the mean, median, and SD of execution time score are (207.7391, 206.9991, and 8.552902 respectively). In all 3 variables, there is a slight difference between the mean and the median, which support normality of the data.

```
match=Hirschberg[1:127,2]
gap=Hirschberg[1:127,3]
time= Hirschberg[1:127,4]*1000

par(mfrow=c(3,3))
hist(match, col=4,main="H Match Score")
hist(gap, col=5,main="H Gap Score")
hist(time, col=3, main = "H Execution Time in ms")

boxplot(match, col=4,main="H Match Score")
boxplot(gap, col=5,main="H Gap Score")
boxplot(time, col=3, main = "H Execution Time in ms")

qqnorm(match,pch=20,cex.lab=1.5,main="H Match Score",
       ylab="Time", col=4)
qqline(match, col="red")

qqnorm(gap,pch=20,cex.lab=1.5,main="H Gap Score",
       ylab="Time", col="red")
qqline(gap, col="blue")

qqnorm(time,pch=20,cex.lab=1.5, main = "H Execution Time in ms",
       ylab="Time", col="red")
qqline(time, col="blue")
```

For Hirschberg's algorithm, match score data is normally distributed according to histogram patterns, but execution time and gap score data are slightly skewed to the left. No gaps in all

histograms, except for gap score. In all box plots, the median is almost in the middle of the First quartile (Q1) and the third quartile (Q3), which support normality of then data., except for match score. Also, we have few outliers in all datasets.

The Q-Q plot for match score shows that normality is probably a reasonably good approximation. The Q-Q plot for execution time has a left tail when the time fall under 430 ms and a right tail above 465 ms and The Q-Q plot for gap score has a right tail a right tail when score goes above 50. We will assume normal distribution of all data sets (Figure 2).

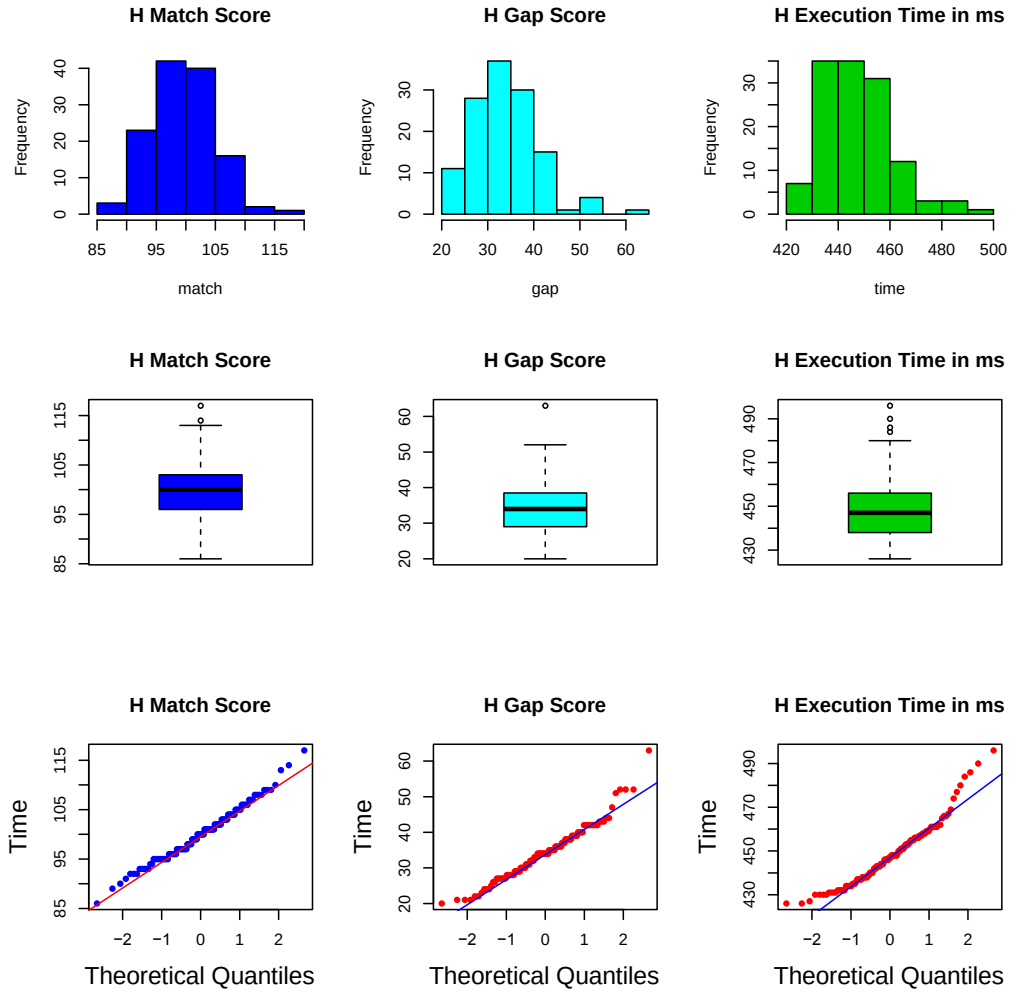

Figure 2: Descriptive statistics for Hirschberg's algorithm; Gap Score, Match score, and Execution time in ms.

```
mean(match);median(match);sd(match)

## [1] 100.0394
## [1] 100
## [1] 5.269562

mean(gap);median(gap);sd(gap)

## [1] 34.37008
## [1] 34
## [1] 7.118948

mean(time);median(time);sd(time)

## [1] 448.0866
## [1] 447
## [1] 13.48722
```

The mean, median, and SD of match score are (100.0394, 100, and 5.269562 respectively), the mean, median, and SD of gap score are (34.37008, 34, and 7.118948 respectively), and the mean, median, and SD of execution time score are (448.0866, 447, and 13.48722 respectively). In all 3 variables, there is a slight difference between the mean and the median, which support normality of the data.

Since we are conducting a parametric test(t-test),we must first start with the test for equality of variances (F-test) for execution time in both algorithms. The null hypothesis H0: the execution time in milliseconds variances is equal in both

algorithms. The alternative hypothesis H1: the execution time in milliseconds variances is not equal in both algorithms.

```
Ftest = var.test(Hirschberg[1:127,4]*1000,NW[1:127,4]*1000,
                 alternative= "two.sided");Ftest

##
##  F test to compare two variances
##
## data:  Hirschberg[1:127, 4] * 1000 and NW[1:127, 4] * 1000
## F = 2.4867, num df = 126, denom df = 126, p-value = 5.211e-07
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.750925 3.531575
## sample estimates:
## ratio of variances

##          2.486669
```

The F test value is 2.4867 and the p-value is less than 5.211e-07, which is highly statistically significant. We have enough statistical evidence to reject the null hypothesis and accept the alternative hypothesis. Inequality of the variances will be assumed when conducting t-test for execution time.

The null hypothesis H0: the execution time in milliseconds means are equal in both algorithms. The alternative hypothesis H1: the execution time in milliseconds means are not equal in both algorithms.

```
t.test(Hirschberg[1:127,4]*1000,NW[1:127,4]*1000, var.equal = FALSE,
       alternative= "two.sided")

##
##  Welch Two Sample t-test
##
## data:  Hirschberg[1:127, 4] * 1000 and NW[1:127, 4] * 1000
## t = 169.6, df = 213.23, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  237.5541 243.1409
## sample estimates:
## mean of x mean of y
##  448.0866  207.7391
```

The Welch Two Sample t-test value is 169.6 and the p-value is less than 2.2e- 16, which is highly statistically significant. We have enough statistical evidence to reject the null hypothesis and accept the alternative hypothesis. Needleman- Wunsch algorithm execution time mean, which is 207.7391 ms, is less than Hirschberg's algorithm execution time mean, which is 448.0866 ms. Needleman- Wunsch algorithm has faster execution time than Hirschberg's algorithm.

Match and gap scores are used to test if the algorithms are producing the same alignments; hence, they are theoretically comparable.

```
cor.test(Hirschberg[1:127,2],NW[1:127,2])

##
##  Pearson's product-moment correlation
##
## data:  Hirschberg[1:127, 2] and NW[1:127, 2]
## t = 12.519, df = 125, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6570012 0.8142587
## sample estimates:
##       cor
## 0.7458468
```

```
cor.test(Hirschberg[1:127,3],NW[1:127,3])

##
##  Pearson's product-moment correlation
##
## data:  Hirschberg[1:127, 3] and NW[1:127, 3]
## t = 0.24268, df = 125, p-value = 0.8087
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1530927  0.1951775
## sample estimates:
##        cor
## 0.02170071
```

Match score values are highly correlated between the algorithms. Correlation coefficient r=0.7458468, p-value less than 2.2e-16, which is highly statistically significant. Gap score values are very weakly correlated between the algorithms. Correlation coefficient r=0.02170071, p-value =0.8087, which is not statistically significant. (Figure 3) is showing scatter plots for the variables in both algorithms.

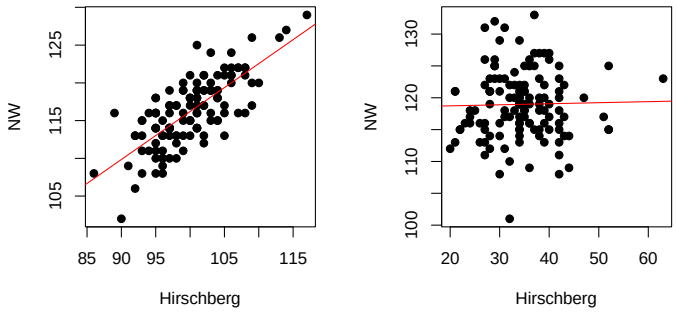

Figure 3: Scatter plot of Gap and Match scores for both algorithms

Discussion

The main objective of this study is to evaluate Hirschberg's algorithm in terms of search space and time complexity by comparing it to the original Needleman- Wunsch using pairwise alignment execution time in millisecond. Although some variables data were slightly skewed, as indicated by histograms and Q-Q plots, normality was assumed for all variables. Both F-test and Welch Two Sample t-test were highly

statistically significant to reject the null hypothesis of equal means of the execution time in milliseconds; consequently, the alternative hypothesis is accepted. The original Needleman-Wunsch algorithm is faster than Hirschberg's algorithm.

Match and gap scores were used to test if the algorithms are producing the same alignments; hence, they are theoretically comparable. Match score values are highly positively correlated (r=0.7458468) and highly statistically significant (p-value less than 2.2e-16) proving that both algorithms output is similar. However, gap score values are very weakly correlated between the algorithms (r=0.02170071) and the test is not no statistically significant (p-value =0.8087), which means that both algorithms are producing different alignments; hence, we cannot compare them in terms of performance.

The poor correlation in match score is possibly related to effect of outliers in the data, so non-parametric test, like Spearman's rank-order correlation, may solve for this problem.

```
cor.test(Hirschberg[1:127,3],NW[1:127,3], method = "spearman")

##
##  Spearman's rank correlation rho
##
## data:  Hirschberg[1:127, 3] and NW[1:127, 3]
## S = 333930, p-value = 0.8077
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##      rho
## 0.021816
```

Spearman's rank-order correlation shows close results to Pearson's correlation. Gap score values are very weakly correlated between the algorithms (r=0.021816) and the test is not no statistically significant (p-value = 0.8077). This is enough prove that presence of outliers is not the direct cause of the weak correlation.

Another possible cause for the weak correlation is a hidden relation between gap scores in certain sequences and execution time, which confounds the final results. To solve for this, all execution time variable outliers have been excluded and only data posing normal distribution are left. Match score correlation then is conducted on the sample with normal distribution only.

```
NNW=filter(NW, Execution.Time*1000 >200)
NH=filter(Hirschberg, Execution.Time*1000 >430,
          Execution.Time*1000 <465)

cor.test(NH[1:103,3],NNW[1:103,3])

##
##  Pearson's product-moment correlation
##
## data:  NH[1:103, 3] and NNW[1:103, 3]
## t = -0.60071, df = 101, p-value = 0.5494
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2503007  0.1354222
## sample estimates:
##         cor
## -0.05966619
```

Gap score values are still very weakly correlated between the algorithms (r=- 0.05966619) and the test is not no statistically significant (p-value = 0.5494). Although our analysis show that Needleman-Wunsch algorithm has faster execution time than Hirschberg's algorithm, algorithms are not comparable since they are having a different gap score; hence, a different mechanism of action.

References

1. S. K. Moore. 2000. Understanding the Human Genome. IEEE Press. 11: 34-35.
2. Todd, J. Vision and Aoife McLysaght. 2003. Computational Tools and Resources in-Plant Genome Informatics.
3. K. Charter, J. Schaeffer, and D. Szafron. 2000. Sequence Alignment using Fast LSA. International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences. 239-245.
4. S. B. Needleman and Christian, D. Wunsch. 1970. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Sequences. Journal of Molecular Biology. 48: 443-453.
5. Hirschberg, D. S. (1975). "A linear space algorithm for computing maxi- mal common subsequences". Communications of the ACM. 18 (6): 341343.