Silesian University of Technology
Macro Faculty

Introduction to Bioinformatics

Laboratory

Sequence alignment

Piotr Wittchen

Specialization: Informatics
Contact e-mail: piotr.wittchen@gmail.com

Gliwice 2010

Introduction

The objective of the laboratory was to answer the question:
"Which of the modern elephants is more closely related to mammoths?"

Solution

Firstly, I found mtDNA sequences of the *Mammuthus primigenius (Mammoth), Elephas maximus (Indian Elephant)* and *Loxodonta africana (African Elephant)* in the NCBI Genbank. Next, I used dynamic programming approach of the Needleman-Wunsch algorithm in order to perform rest of the task. At the beginning, I extracted only 100 nucleotides from the each mtDNA sequence starting from the nucleotide no. 6000. When I compared sequences starting from the lower numbers of nucleotides, both sequences were exactly the same, so I had to move the range of comparison. After that, I created *DP Array* also known as *trace-back matrix* and filled it according to the algorithm. Afterwards, I performed trace-back procedure in order to locate gaps in the sequences, find the best alignment and score. I run whole program twice: for comparison between Mammoth and Indian Elephant and for comparison between Mammoth and African Elephant. Program was written in C++ language and compiled by g++ compiler under the Windows OS. Some output data (e.g. *DP Array*) can be extracted into the *.m Matlab file by the program. All procedures (filling DP Array, trace-back procedure and printing the alignment) are included in the Source code section in the further part of the report.

Global vs. local sequence alignment

I applied **global alignment**, because as we can read in the lecture notes *if two sequences have the same ancestor, we expect to have many symbol, especially entire subsequences, in common. Therefore most symbols in one seq. have corresponding homologous positions in the other seq. The alignment of entire seq. can be thought of as a finding the correspondence between their respective symbols.* All species examined in this laboratory have the same ancestor and very similar mtDNA sequence, so I had to apply global alignment.

Output data

First alignment:

Comaprison between:
Mammuthus primigenius (Mammoth) and Elephas maximus (Indian Elephant)

Sequence Alignment of the parts of the mtDNA

S1: >gi|164454826|dbj|AP008987.1| Mammuthus primigenius mitochondrial DNA, complete genome
S2: >gi|37496447|emb|AJ428946.1| Elephas maximus complete mitochondrial genome

```
GAGGA-GA--CC---TTCTATACCAACACCTATTCTGGTTTTTTGGACACCCTGAAGTCTATATTCTAATTCTCCCAGGATTTGGAATAGTTTCTCATAT
|   |  ||  |     | ||     ||   |  |    ||||    || |       | |||  |     || |      |     |   | ||  ||
GGAGGAGA-CC--TTCTATACCAACACCTATTCTGGTTTTTTGGACACCCTGAAGTCTATATTCTAATTCTCCCAGGATTTGGAATAGTTTCTCATAT--
```

Score of the alignment is: -23

Second alignment:

comaprison between:
Mammuthus primigenius (Mammoth) and Loxodonta Africana (African Elephant)

Sequence Alignment of the parts of the mtDNA

S1: >gi|164454826|dbj|AP008987.1| Mammuthus primigenius mitochondrial DNA, complete genome
S2: >gi|3021460|emb|AJ224821.1| Loxodonta africana complete mitochondrial genomic sequence

```
GAGGA-GA--C-----T-TAT--CAACACCTATTCTGGTTTTTTGGACACCCTGAAGTCTATATTCTAATTCTCCCAGGATTTGGAATAGTTTCTCATAT
|||||                  || |   | |  |  |||      |     |       | |  |  | |    |        | |   |  |    |
GAGGAGA--C-----T-TATCAACACCTATTCTGGTTTTTTGGACACCCTGAAGTCTATATTCTAATTCTCCCAGGATTTGGAATAGTTTCTCATA---T
```

Score of the alignment is: -29

Source code

Note: Source code of the program is also attached to this report in the *sequenceAlignment.cpp* file.

```
/**
 * Introduction to Bioinformatics - Sequence alignment
 * @author: Piotr Wittchen <piotr.wittchen@gmail.com>
 * Macrofaculty, specialization: Informatics
 */

#include <iostream>
#include <fstream>

/**
 * initial value determining what sequences we compare
 * comaprison = 1 for comaprison between Mammuthus primigenius (Mammoth) and Elephas maximus (Indian Elephant)
 * comaprison = 2 for comaprison between Mammuthus primigenius (Mammoth) and Loxodonta africana (African Elephant)
 */
int comparison = 1;

/**
 * global variables for storing sequences their names and DPArray
 */
char * sequenceName1;
char * sequenceName2;
char * sequenceData1;
char * sequenceData2;
int ** DPArray;

/**
 * allocating memory for sequences and DPArray
 */
void allocateMemory()
{
    sequenceName1 = new char[100];
    sequenceName2 = new char[100];
    sequenceData1 = new char[100];
    sequenceData2 = new char[100];

    DPArray = new int*[100];
```

```
    for(int DPi = 0; DPi < 100; DPi++)
        DPArray[DPi] = new int[100];
}

/**
 * deallocating memory
 */
void deallocateMemory()
{
    delete[] sequenceName1;
    delete[] sequenceName2;
    delete[] sequenceData1;
    delete[] sequenceData2;
    for(int DPi = 0; DPi < 100; DPi++)
        delete[] DPArray[DPi];

    delete[] DPArray;
}

/**
 * reading 100 nucleotides starting from the 6000th position
 * in the single mtDna sequence inside the FASTA file
 */
void readMtDNA(char * FASTAfile, int sequenceNumber, int startingPosition = 6000)
{
    std::ifstream file;
    file.open(FASTAfile);

    char fileGet;
    bool sequenceBegin = false;
    int i = 0, j = 0, k = 0;

    while(file.good())
    {
        fileGet = char(file.get());

        if(fileGet == 10) // new line
            sequenceBegin = true;

        if(!sequenceBegin)
        {
            if(sequenceNumber == 1)
```

```
                sequenceName1[k++] = fileGet;
            else
                sequenceName2[k++] = fileGet;
        }

        if(sequenceBegin && fileGet != 10 && i < 100)
        {
            if(j == 0)
            {
                if(sequenceNumber == 1)
                    sequenceName1[k] = '\0';
                else
                    sequenceName2[k] = '\0';
            }

            // reading sequence from the nucleotide on the custom starting position (6000th)
            // in order to perform comparison,
            // because in the earlier parts of mtDNA sequences are exactly the same
            if(j > startingPosition)
            {
                if(sequenceNumber == 1)
                    sequenceData1[i++] = fileGet;
                else
                    sequenceData2[i++] = fileGet;
            }
            j++;
        }
    }
    file.close();
}

/**
 * filling DPArray with a given initial penalty
 */
void fillDPArray(int penalty = -6)
{
    for(int i = 0; i < 100; i++)
    {
        for(int j = 0; j < 100; j++)
        {
            if(i == 0)
                DPArray[i][j] = j * penalty;
```

```
            else if(j == 0)
                DPArray[i][j] = i * penalty;
            else
            {
                int * choosingArray = new int[3];
                choosingArray[0] = DPArray[i][j - 1] + penalty;
                choosingArray[1] = DPArray[i - 1][j] + penalty;

                if(sequenceData1[i] == sequenceData2[j])
                    choosingArray[2] = DPArray[i - 1][j - 1] + 1;
                else
                    choosingArray[2] = DPArray[i - 1][j - 1];

                int max = choosingArray[0];

                for(int k = 0; k < 3; k++)
                {
                    if(choosingArray[k] > max)
                        max = choosingArray[k];
                }

                delete[] choosingArray;
                DPArray[i][j] = max;
            }
        }
    }
}

/**
 * saving DPArray into Matlab *.m file
 */
void saveDPArray()
{
    std::ofstream file;
    file.open("DPArray.m");
    file << "DPArray = [";
    for(int m = 0; m < 100; m++)
    {
        for(int n = 0; n < 100; n++)
            file << DPArray[m][n] << ",";

        file << ";" << std::endl;
```

```
    }
    file << "];";
    file.close();
}

/**
 * traceback procedure
 * beginning from DPArray[99][99]
 * ending at DPArray[0][0]
 */
void traceback()
{
    int i = 99, j = 99;
    while( !(i <= 0 && j <= 0) )
    {
        if(sequenceData1[i] == sequenceData2[j])
        {
            i--;
            j--;
        }
        else
        {
            if(DPArray[i - 1][j] > DPArray[i][j - 1])
            {
                sequenceData1[i] = '-';
                i--;
            }
            else
            {
                sequenceData2[j] = '-';
                j--;
            }
        }
    }
}

void printSequences()
{
    for(int i = 0; i < 100; i++)
        std::cout << sequenceData1[i];

    std::cout << std::endl;
```

```
    for(int j = 0; j < 100; j++)
    {
        if( (sequenceData1[j] == sequenceData2[j]) && sequenceData1[j] != '-' )
            std::cout << "|";
        else
            std::cout << " ";
    }

    std::cout << std::endl;

    for(int k = 0; k < 100; k++)
        std::cout << sequenceData2[k];

    std::cout << std::endl << std::endl;;
}

void printScore()
{
    int score = 0;

    for(int i = 0; i < 100; i++)
    {
        if( (sequenceData1[i] == sequenceData2[i]) && sequenceData1[i] != '-' )
        {
            score++;
        }
        else if( (sequenceData1[i] != sequenceData2[i]) && (sequenceData1[i] != '-' && sequenceData2[i] != '-') )
        {
            score--;
        }
        // else: if we find the gap '-', we add 0 to the score, so score remains the same
    }
    std::cout << "Score of the alignment is: " << score << std::endl << std::endl;
}

/**
 * running the program
 */
int main()
{
```

```
    allocateMemory();
    readMtDNA("AP008987.fasta",1); // Mammuthus primigenius (Mammoth)

    if(comparison == 1)
        readMtDNA("AJ428946.fasta",2); // Elephas maximus (Indian Elephant)
    else
        readMtDNA("AJ224821.fasta",2); // Loxodonta africana (African Elephant)

    std::cout << "Sequence Alignment of the parts of the mtDNA" << std::endl << std::endl;
    std::cout << "S1: " << sequenceName1 << std::endl;
    std::cout << "S2: " << sequenceName2 << std::endl << std::endl;

    fillDPArray();
    traceback();
    printSequences();
    printScore();
    system("pause");
    deallocateMemory();
    return 0;
}
```

Conclusions

In the "Output data" section we can clearly see alignments of the sequences and their scores, which are as follows:

No. of the alignment	1	2
Alignment	Mammuthus primigenius	Mammuthus primigenius
	Elephas maximus (Indian Elephant)	Loxodonta africana (African Elephant)
Score	-23	-29

We can observe that Elephas maximus (Indian Elephant) has better score than Loxodonta africana (African Elephant), so I can say that:

Mammoth is more closely related to the **Elephas maximus (Indian Elephant)** because of the better alignment score.

Bibliography

- "Introduction to Bioinformatics – Sequence Alignment" – lecture notes by Łukasz Olczak
- http://baba.sourceforge.net/ - Basic Algorithms of Bioinformatics Applet
- http://www.ludwig.edu.au/course/lectures2005/Likic.pdf - The Needleman-Wunsch algorithm for sequence alignment (7th Melbourne Bioinformatics Course) by Vladimir Likić, Ph.D.
- "Bioinformatics" by Andrzej Polański and Marek Kimmel