DNA Trace Reconstruction

Rishi Roy, Hrishi Narayanan, Udvas Basak

1 Introduction

The primary objective is to reconstruct the original string from m random traces(subsequences) of the original string, where each trace is obtained by independently deleting elements of the original string. The algorithm needs to be optimized so that the reconstruction takes place in lower order time.

We investigate a number of algorithms for the abovementioned problem, and arrive at conclusions regarding their feasibility and applicability in different situations and values of parameters. The most preliminary algorithm for this is the Bitwise Majority Algorithm(BMA) which we investigated, and then compared the performance of the modified algorithms with BMA. All the investigations include simulations, followed by analysis of the results.

Even though the objective is to work with DNA, which is a quaternary alphabet, our simulations start with a binary alphabet, which we can very easily transport into a quaternary one.

2 Terminologies

The original string that has to be reconstructed is referred to as the ***transmittedString***.

Every string that is obtained after independently deleting each element with a predefined ***deletion probability***, is referred to as a ***trace*** or a ***transmission***.

The array, each of whose rows is one trace obtained from the same transmittedString, is referred to as the ***DataSet*** or the ***Binary Dataset Matrix***. The matrix has l rows, where l is the number of traces, and c columns, where c is the length of the longest trace generated(sometimes padded with extra characters).

Whenever percentage error is being calculated, 1000 independent samples are generated, and the percentage of samples that return a wrong answer is being calculated.

3 Bitwise Majority Algorithm

3.1 Main Algorithm

With a thorough understanding of the algorithm, the BMA is coded.

The pseudo code is as follows:

```
Take the binary dataset matrix with size (l,c) as input

Create an array of pointers #p of length l, initialized to zero.
Create a binary array #transmittedString of length c.

Loop k from 0 through c:
    Take a majority vote #mv along all members[i, p[i]] ##i runs from 0 to
    l.
    transmittedString[k] <- mv
    If p[i] not equals mv
        Increment p[i] by 1

Return transmittedString
```

Listing 1: BMA Algorithm PseudoCode

3.2 DataSet Generation

For simulating the results and also to test whether the algorithm can successfully reconstruct the trace, we need to generate a random dataset, from a randomly generated arbitrary binary string. A code is generated for the same, which takes the parameters *length of the original string(c), no. of traces to generate(l), and the deletion probability(p).*

The pseudocode is as follows:

```
Generate a random c-length binary string #rb.
transmittedString <- rb

Generate an empty matrix of size(l,c) #dataset.

For each trace:
    Generate the trace by independently deleting elements from
   transmittedString with probability p.
    If length of trace is less than c, pad with zeros at the end.
    dataset[i] <- trace

Return transmittedString, dataset
```

Listing 2: DataSet Generation for BMA

3.3 Results of the Analysis

For the analysis, we shall run the algorithm for a range of lengths, a range of the number of traces, a range of deletion probabilities, and investigate the percentage error and the time taken for the algorithm to work.

The length ranges from 10 to 100, in steps of 5.
The number of traces range from 3 to 50.
The deletion probabilities are (0.1, 0.3, 1.0, 3.0, 10.0).

The results are analyzed and the plots are presented.

3.4 Important Features and Results

- Since the algorithm can work on random strings, we can input a string of any kind, and can expect a correct result when the conditions are favourable.