Analysis of Greedy DNA Reconstruction Algorithms

Jose Torres Postigo

2023-11-17

1 Objectives

The objectives of this lab excercise is the implementation, evaluation and comparison of the performance of two DNA sequencing algorithm implementations: the NaiveGreedyPathSequencing and NaiveBidirectionalGreedyPathSequencing.

Both algorithms aim to reconstruct a DNA sequence from a set of fragmented reads obtained through Shotgun sequencing. The unidirectional naive algorithm follows a simple greedy approach, selecting the next fragment based on maximum overlap. The bidirectional variation extends the NaiveGreedyPathSequencing by considering extensions in both forward and backward directions.

Finding the best reconstruction is equivalent to finding the minimum Hamiltonian path (path that visits each vertex exactly once), if we imagine a weighted complete directed graph $G(V,\ E,\ w)$, where

- There is a vertex v_i for each fragment $f_i,\ 1 \leq n$.
- There is a directed edge $(v_i,\ v_j)$ for $1 \leq\ i,\ j \leq n$.
- The weight $w((v_i,\ v_j))$ of each edge is the overlap $\omega(f_i,\ f_j)$.

A generic greedy path-building approach for the Hamiltonian path problem would pick a starting vertex and then subsequently select the next vertex to visit in a greedy way.

Adapting this aproach for the DNA assembly problem, in the unidirectional implementation we pick an intial fragment at random and then in each step we assemble the fragment (among those still available) with the largest overlap to the last one picked. For the improved version of this, the bidirectional variation, it will be picked a fragment to be assembled to either of the two end, whatever it is better.

2 Experimental Setup

For these experiments, both algorithms will be run through a class given by the lecturer, which is going to run tests for them using sequences and reads of incresing number. The initial values for these tests are five sequence lengths, starting at a size of 128 and doubling each time, with read sizes from 4 to 8 and 100 tests per parameter combination; up to sequences with a size of 2048.

Table 1: Computational environment considered.

CPU	Intel® Core™ i5 11600KF, 16GB
OS	Windows 11 Home 22H2
Java	openjdk 17.0.7 2023-04-18

3 Empirical Results

A summary of the experimental results is provided in Tables 2 and 3 in the Appendix.

In Figure 1, it can be seen the plots regarding the time as a function of the sequence length for different reads lengths. Taking a look at the results, it can firmly be said that, although for small sequences both take approximately the same time, in general the naive implementation is very fast, while the bidirectional implementation is much slower.

This is because the naive implementation simply reads the DNA sequence from one end to the other, while the bidirectional implementation reads the sequence from both ends simultaneously. This makes the bidirectional implementation slower, but the length excess is going to be lower, because it is less likely to miss a better read.

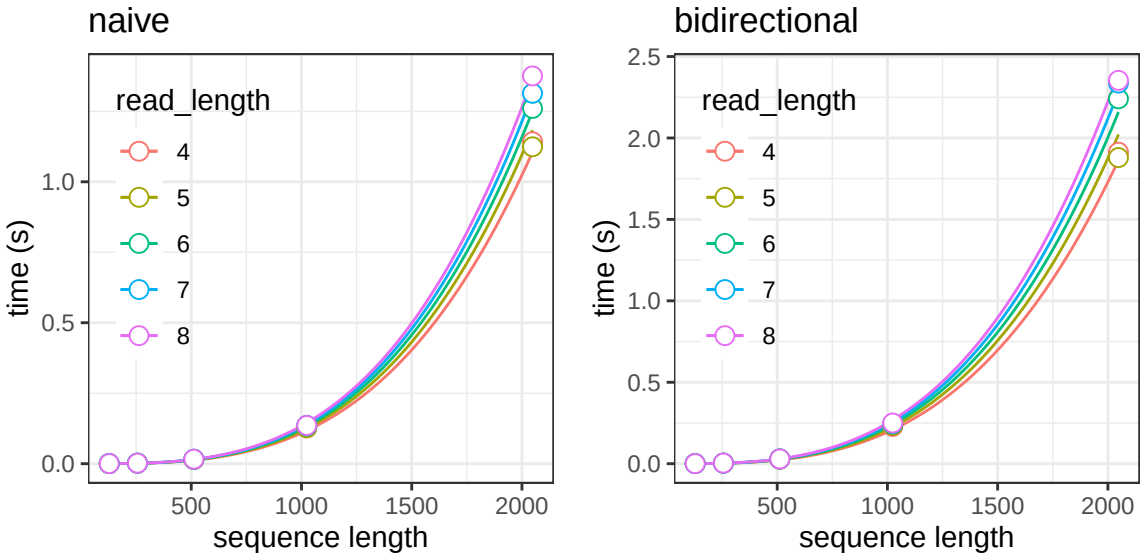
For both implementations, the time complexity seems to be $O(n\ log\ n)$. Studying the plots, the time complexity do not depend on the length of the fragments because they present the same growth rate for different read lengths. The sequence length of the input, however, is a key factor that affect directly to the time complexity.

In Figure 2, it can be seen the plots regarding the length excess of the reconstructed solution as a function of the sequence length for different read lengths. The first plot shows the results for the naive implementation of the algorithm, and the second plot shows the results for the bidirectional implementation.

The naive implementation of the algorithm is very fast, as has been pointed out above, but presents a larger percentage of length excess than the bidirectional implementation. This is because the naive implementation simply reads the DNA sequence from one end, missing possible better matches if this occurs at the other end of the sequence.

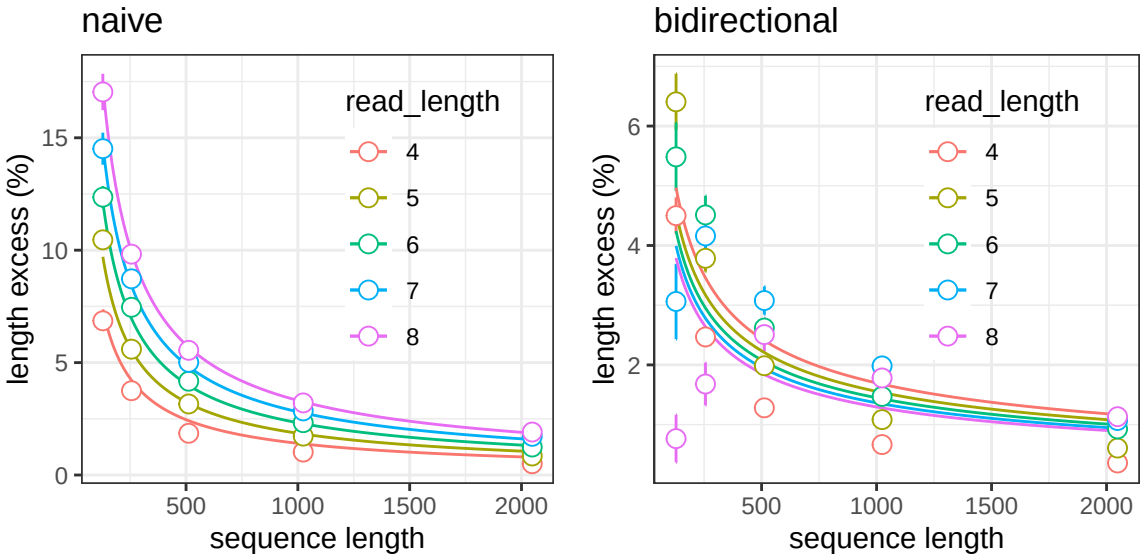
The bidirectional implementation of the algorithm, although is slower than the naive implementation, it is less likely to miss the best match in each case. This is because the bidirectional implementation reads the DNA sequence from both ends simultaneously.

It is a surprise that the bidirectional implementation is very inaccurate regarding the theoretical predictions of the length excesses. Some of the readings made in the execution are much lower than the theoretical prediction, others much higher.



(a) naive (b) bidirectional

Figure 1: Time as a function of the sequence length for different read lengths.



(a) naive (b) bidirectional

Figure 2: Length excess of the reconstructed solution as a function of the sequence length for different read lengths.

4 Discussion

Comparing the algorithms involves assessing their performance in terms of both time complexity and the length of the reconstructed sequence. The experimental results obtained have been analyzed from running both the unidirectional and bidirectional implementations under different scenarios.

After studying the results, it can be said that the naive implementation of the DNA sequence algorithm is probably the most commonly used implementation because it is fast enough for most applications. However, the bidirectional implementation is more accurate, especially for longer sequences. So it is up to each one to make the decision of what implementation is going to use, depending on what are their necessities.

In general the theoretical predictions are very accurate, except the one seen in Figure 2 for bidirectional implementation. This could be for some reasons: the Hardware of this experimental setup is not the appropiate one or there is some error in the implementation. It could not be, in my opinion, a problem with the regression model because the readings obtained are kind of random.

A Appendix

A.1 Data Summary

Summary of the experimental results for sequences and reads of different sizes. The mean and standard error are provided for the computational time and the length excess of the sequence generated over the true solution.

Table 2: Results for the naive greedy algorithm.

sequence	read	time (mean)	time (stderr)	excess (mean)	excess (stderr)
128	4	0.0005	0.000045	6.86	0.290
128	5	0.0003	0.000013	10.46	0.368
128	6	0.0003	0.000006	12.36	0.494
128	7	0.0003	0.000007	14.52	0.712
128	8	0.0004	0.000009	17.04	0.807
256	4	0.0020	0.000014	3.75	0.134
256	5	0.0020	0.000016	5.60	0.200
256	6	0.0020	0.000026	7.46	0.242
256	7	0.0019	0.000012	8.73	0.337
256	8	0.0019	0.000020	9.82	0.411
512	4	0.0148	0.000036	1.86	0.070
512	5	0.0155	0.000035	3.16	0.101
512	6	0.0159	0.000041	4.17	0.144
512	7	0.0159	0.000061	5.01	0.166
512	8	0.0159	0.000037	5.55	0.196
1024	4	0.1270	0.000524	1.02	0.036
1024	5	0.1278	0.000395	1.73	0.056
1024	6	0.1337	0.000667	2.33	0.060
1024	7	0.1361	0.000755	2.85	0.093
1024	8	0.1356	0.000554	3.21	0.116

2048	4	1.1420	0.012791	0.50	0.018
2048	5	1.1242	0.010147	0.85	0.029
2048	6	1.2602	0.007317	1.24	0.039
2048	7	1.3137	0.002762	1.72	0.041
2048	8	1.3754	0.009281	1.91	0.059

Table 3: Results for the bidirectional greedy algorithm.

sequence	read	time (mean)	time (stderr)	excess (mean)	excess (stderr)
128	4	0.0008	0.000053	4.50	0.305
128	5	0.0006	0.000003	6.41	0.469
128	6	0.0006	0.000012	5.48	0.570
128	7	0.0006	0.000014	3.06	0.630
128	8	0.0007	0.000013	0.77	0.397
256	4	0.0038	0.000039	2.47	0.152
256	5	0.0037	0.000017	3.79	0.222
256	6	0.0037	0.000013	4.51	0.320
256	7	0.0037	0.000023	4.16	0.400
256	8	0.0037	0.000035	1.68	0.355
512	4	0.0273	0.000049	1.28	0.076
512	5	0.0287	0.000084	1.99	0.119
512	6	0.0295	0.000055	2.62	0.154
512	7	0.0297	0.000073	3.08	0.234
512	8	0.0295	0.000060	2.51	0.277
1024	4	0.2286	0.000679	0.67	0.035
1024	5	0.2365	0.000757	1.08	0.059
1024	6	0.2437	0.000904	1.47	0.091
1024	7	0.2475	0.000947	1.98	0.114
1024	8	0.2500	0.001117	1.78	0.154
2048	4	1.9125	0.017376	0.36	0.019
2048	5	1.8797	0.016965	0.61	0.030
2048	6	2.2403	0.017137	0.93	0.049
2048	7	2.3369	0.003566	1.07	0.063
2048	8	2.3536	0.004522	1.13	0.078

A.2 Model Fitting

Summary of the statistical models found. In all cases, x corresponds to sequence length, and y to read length.

A.2.1 Naive Greedy Algorithm

A.2.1.1 Model for time

```
## Nonlinear regression model
##   model: z ~ a * y^b * x^c
##    data: data.frame(x = x, y = y, z = z)
##         a         b         c
```

```
## 1.398e-11 3.044e-01 3.236e+00
##  residual sum-of-squares: 0.005226
##
## Number of iterations to convergence: 16
## Achieved convergence tolerance: 4.003e-07
```

A.2.1.2 Model for length

```
## Nonlinear regression model
##   model: z ~ a * y^b * x^c
##    data: data.frame(x = x, y = y, z = z)
##       a       b       c
##  0.6601  1.2346 -0.8046
##  residual sum-of-squares: 0.0002279
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 2.782e-06
```

A.2.2 Naive Bidirectional Greedy Algorithm

A.2.2.1 Model for time

```
## Nonlinear regression model
##   model: z ~ a * y^b * x^c
##    data: data.frame(x = x, y = y, z = z)
##         a         b         c
## 3.970e-11 3.632e-01 3.157e+00
##  residual sum-of-squares: 0.03458
##
## Number of iterations to convergence: 15
## Achieved convergence tolerance: 2.269e-06
```

A.2.2.2 Model for length

```
## Nonlinear regression model
##   model: z ~ a * y^b * x^c
##    data: data.frame(x = x, y = y, z = z)
##       a       b       c
##  1.0752 -0.3901 -0.5222
##  residual sum-of-squares: 0.002805
##
## Number of iterations to convergence: 13
## Achieved convergence tolerance: 6.227e-06
```