A Mediocre Primer Finding Program

Johan Lehto

May 2022

Advanced Bioinformatics, 7.5hp

Contents

1 Abstract

To quickly identify potential pathogens or viral infections using PCR is needed in many different sectors, such as business and healthcare. This requires very specific primers in order to determine which or what species is present in the samples. The program in this paper, A Mediocre Primer Finding Program, achieved linear time complexity for aligning potential primers to the genome by using a recursive depth-first traversal algorithm to traverse a trie constructed from the genome. It can be used to generate precise primers for PCR amplification. The program is not suitable for larger genome due to its poor time scaling and is the most optimal for smaller viral or bacterial genomes.

2 Introduction

The bacteriophage is a type of virus that relies on bacteria for replication. This type of virus is usually highly specific to an individual species of bacteria, or to a specific strain within a species. The total amount of bacteriophages present in the biosphere is estimated to be 10^{31}, which makes them a candidate for the most abundant organism on earth [6][5]. The virus replicates by attaching itself to its target and injecting it with its genome, depending on the virus, the replication can either be lytic or lysogenic. During lytic replication, the virus utilizes the host bacterium's own biological machinery to construct copies of itself. Once the host bacterium die, it is lysed, releasing the new virus particles into the environment. In the lysogenic replication route, the virus genome is integrated into the host bacterium genome, this means that the virus DNA can be passed down to newer generations of bacteria, these infected bacteria are called prophages. The prophages can change to lytic replication [5].

Bacteriophages are of interest in food safety, agriculture and in the business of diary fermentation. Since bacteriophages only targets bacteria, and not mammalian cells, they can be used to target harmful bacteria during food production. This group of viruses have been proposed as an alternative to antibiotics [4]. Despite the potential usefulness of bacteriophages outlined above, they can be a problem in other fields, such as diary fermentation. This field relies on lactic acid-producing bacteria (LAB) to ferment the product [7]. Infection of starter cultures used for fermentation would both decrease the quality of the end product but it is also a financial burden on the company, due to potential spoiling of the product and the counter measures that need to be taken to prevent infection.

Viruses do not only target the food industry but they also cause problems for the global population. As the global SARS-CoV-2 pandemic has showed, quick and easy testing is needed to identify which virus or strain is behind the infection. These tests need to evolve alongside the virus due to the high mutation rate,

which is further amplified by the global scale of the pandemic. One of the most common methods of diagnosing a SARS-CoV-2 infection is by doing real-time RT-PCR, this method combines reverse transcription (RT) with polymerase chain reaction (PCR) [3]. These test need to be fast and accurate to identify which strain of SARS-CoV-2 is causing the infection, the sensitivity of the test directly depends on the primers used for the RT-PCR. Reverse transcription needs to be used together with the PCR due to SARS-CoV-2 being a ssRNA virus.

The program in this report aims to help solve some of these problems. The program can find primers from a user provided genome, these primers adhere to user implemented parameters, such as target binding temperature (Tm) and the minimum temperature difference between Tm and any other location in the genome. These primers can then be used for PCR amplification to quickly determine what bacteriophage, bacterium or other virus is present in the medium, thus allowing for a quick response in the event of viral infection. The program provides both normal primer pairs and circular primers pairs (pairs that span the end and beginning of the genome) suitable for plasmids or other forms of circular DNA. A viral genome needs to undergo reverse transcription before being handled by the program. A time analysis will also be done on the program to measure its time complexity to determine its suitability for primer finding in large genomes.

3 Material and Methods

3.1 Program information

This program only handles DNA and it assumes that only the four letters of the DNA alphabet (A, T, C, G) is present in the sequence given to the program. All DNA output from the program is in the 5' to 3' direction, if not otherwise stated. The program can only read from fasta files. All primer lengths from the program is 20 bases in length. the G/C content of the primers is within the range of 40% to 60% and the last base of each primer is always G or C. Primer specificity is ensured by not allowing any primer to bind perfectly anywhere in the genome more than once. Furthermore, it can not bind to another location in the genome within the user specified ΔT limit. The resulting primer pairs from the program is a pair which generates a fragment that is within a user specified length and, if possible, is digestible by EcoRI. The formula used for the Tm and Ta calculations is the Marmur Doty formula [8]:

$$Temperature = (\sum A + \sum T)2 + (\sum C + \sum G)4$$

which gives the temperature in $°C$.

3.2 Material

The program is written in the programming language Python, version 3.9 [9]. The Python module Biopython [2] was used to read sequences from fasta files, to calculate the G/C content of a sequence using the $GC()$ function and the $RestrictionBatch()$ function was used to generate the EcoRI restriction sites. The depth-first search algorithm used to traverse the trie was taken from [10] and [1]. The search algorithm is a combination of the original algorithm calculating the hamming distance for amino acids and the depth-first traversal algorithm included in the trie datatype from [1]. The trie datatype was taken from [1], reduntant functions were removed but nothing else was changed.

3.3 Data pipeline

When a sequence is loaded into the program it is split into N sequences consisting of 20 bases. This is done by sliding a window of size 20 along the sequence. This is done both for the forwards and reverse strand The N sequences, from the forward and reverse strand, resulting in 2N sequences which are used to construct the trie. Each node in the trie is a nucleotide, and each path in the trie is sequence existing in the genome. Since all generated primers have a length of 20 bases, the depth of the trie is 20 nucleotides. The 2N sequences are also considered as potential primers, thus they will undergo a selection process to sort out bad primers. Only the primers with the specified Tm, G/C content within the specified range, and ends with a G/C, will be kept. The remaining primers well then undergo a depth-first traversal of the trie in order to align the primers to the genome, after this stage only primers with a ΔT value bigger or equal to user specified value are kept. The primers that are left after this stage is paired with a forward or reverse primer. The fragment yielded from the pair is then digested with EcoRI.

3.4 Search Algorithm

The search algorithm used to traverse the trie is a recursive depth-first traversal algorithm, named $hamming\_distance()$ and $recursive\_search()$ in the program code. The algorithm takes a primer and traverses the trie, counting mismatches along the way. The search starts at the root node of the trie, which is an empty string. The algorithm then looks at the root node's children and begins the search at the first child, then $hamming\_distance()$ does the initial recursive call to $recursive\_search()$. If the primer exceeds the user specified ΔT value, the

traversal of the current branch is stopped and it goes to the next branch. If the algorithm finds two locations in the genome that perfectly fits the primer, or if it finds a location within the user specified ΔT value, the search is aborted completely. If the search is not prematurely stopped, it will traverse all branches in the trie. The algorithm can be summed in three points:

1. Begin traversal from root node.
2. After each node: check if current cost $>$ specified ΔT value
   (a) If true: abort search of current branch
   (b) If false: continue search
3. If at end node: see if current ΔT is equal to 0
   (a) If true: continue search (every primer has at least one perfect match)
   (b) If false: 1 is added to the result list and the search is aborted completely for the current primer

4 Results

The speed of the program was tested using $Tm = 60°C$ and $\Delta T = 20°C$ for fasta files with randomized sequences and various sequence lengths. A test was also done the bacteriophage P2 (NC_001895.1) genome with $Tm = 60°C$ and $\Delta T = 17°C$, the size of this genome is 33593 bases. The size of the constructed trie for various genome sizes is shown in figure 1. The total size of the trie seems to grow linearly with the size of the genome.

The same graph, but for the amount of potential primers, are shown in figure 2. The number of potential primers displays the same pattern as the trie sizes in figure 1. The amount of potential primers grows linearly with the size of the genome used. This makes it possible to compare run-times for different fasta file sizes with out having to normalize the run-times with respect to trie size and the amount of potential primers. The total time for running the program using the same fasta files as in figure 1 and 2, is shown in figure 3. The time it takes for the program to complete seems to have a linear time complexity, which is to be expected since both the size of the trie and the amount of potential primers also grows linearly.

Figure 4 shows how much time the program takes to finish when using different ΔT values for the P2 bacteriophage genome. Even ΔT values (except $\Delta T = 20$) are omitted due to the nature of the formula used to calculate the ΔT value. Since the Tm and ΔT sums can only be even when using the Marmur Doty formula, the time difference between $\Delta T = 19$ and $\Delta T = 18$ is the same since

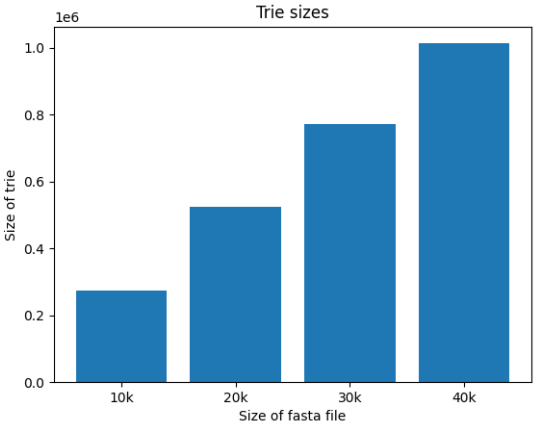

Figure 1: Size of the trie for different fasta file sizes.

the same primers will be discarded (and discarded at the same time point) if the ΔT limit is 19 or 18. The time seems to follow a linear pattern, at $\Delta T = 15$ the increase is minimal. This points to another linear scaling factor for the time it takes to traverse the trie. The time it takes to generate primer pairs is shown in figure 5. This task increases drastically when ΔT is decreased. This is the second most expensive task in the program, and it has a $O(n^2)$ complexity (2 for-loops inside eachother), which will be noticeable when running large genomes or using low ΔT values.

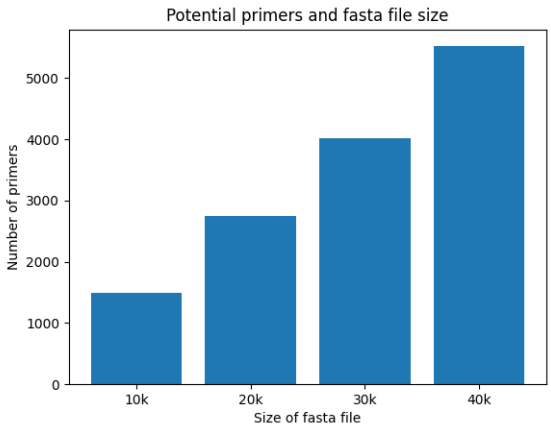

Figure 2: Number of potential primers at different fasta file sizes

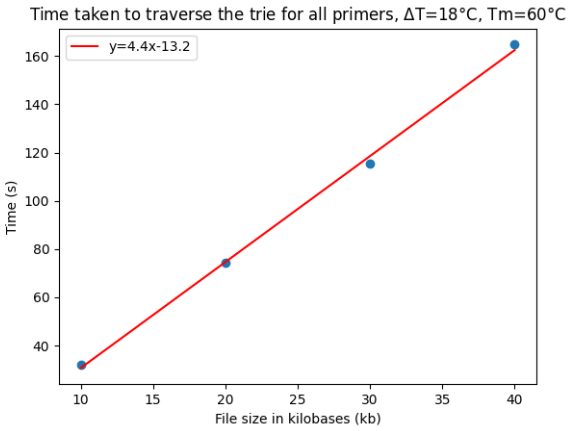

Figure 3: Time taken for the program for different fasta file sizes

A PCR test was done using the SARS-CoV-2 Wuhan variant (NC_045512.2) genome as DNA template together with primers generated from the program, this is shown in figure 6. The first lane to the right of the DNA ladder is the lane in which the primer pair from the program were loaded, named Lane 1 in the figure. The primer pair used was: 5'-TGATCTCAGCTGGCTTTAGC-3' (forward) and 5'-GATCTGAATCGACAAGCAGC-3' (reverse). This primer pair generates a fragment with the size of 1416 bases. Only a single band is visible at the appropriate location for the size of the fragment, which means that the program generated a good primer pair.

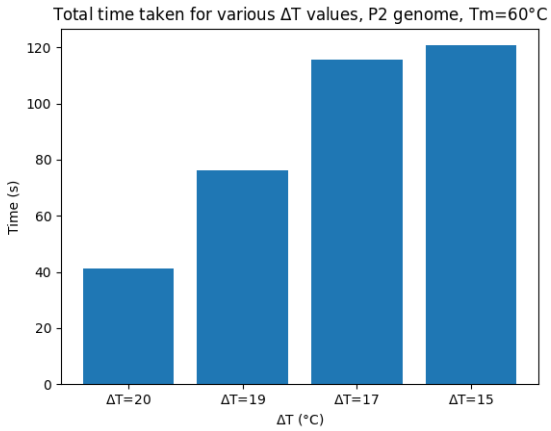

Figure 4: Time taken for the program to finish using different Ta ($^{\circ}C$) settings on the P2 bacteriophage genome.

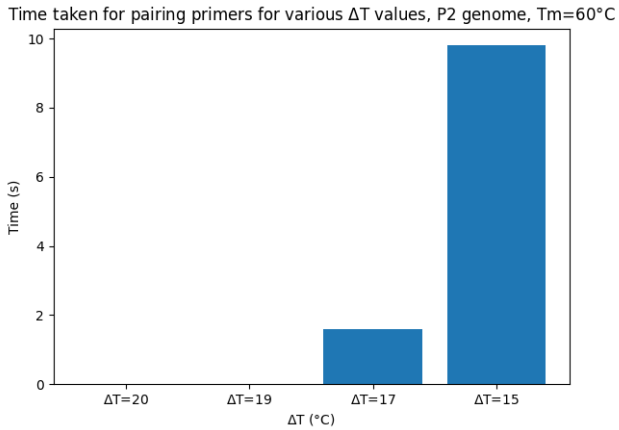

Figure 5: Time taken to generate primer pairs for different Ta values for the P2 genome.

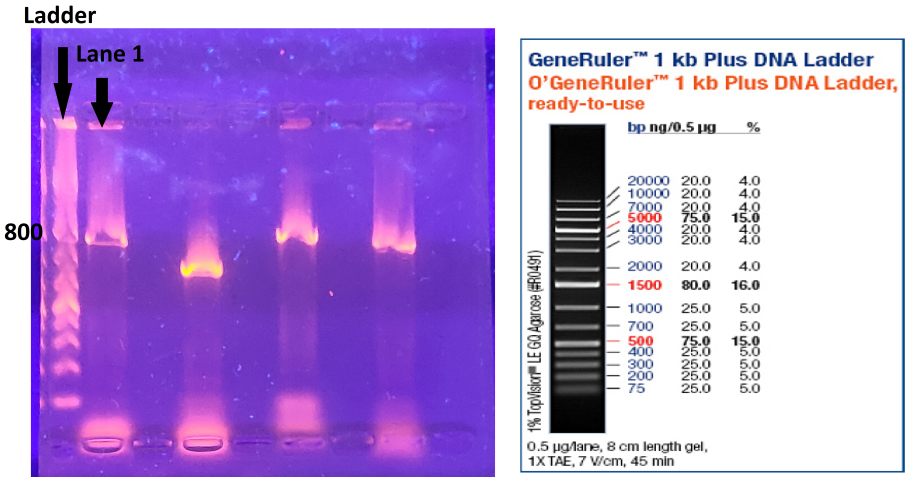

Figure 6: PCR test using a primer pair generated by the program. Despite low visual clarity of the DNA ladder, clear bands are visible at the appropriate location.The fragment size loaded in Lane 1 was 1416 bases, which is closed to the 8th band from the bottom according the DNA ladder.

5 Discussion

Due to time limitations and limit coding knowledge, the program's time complexity scales poorly with the number of primers and the total size of the trie. Since the size of the trie and the amount of primers seems to scale linearly with the size of the genome used, this program is not suitable for larger genomes. To combat the time it takes for the program to run, one can adjust the ΔT value, this means that more primers will be discarded faster due the stricter conditions placed on the primers. The major factors that affect the time complexity seems to be the size of the genome and the ΔT value, as indicated by figure 3 and 4 respectively. If the program is given a large genome it will take a long for it to run normally, if the ΔT value is lowered, this time will increase linearly for each meaningful step down in ΔT. Generating primer pairs is currently a big time sink if the amount of potential primers are large, finding a smarter way than the current $O(n^2)$ solution would increase the program efficiency when using low ΔT values or when finding primers for larger genomes.

Another strategy to decrease the time cost of the program is to do quality control of the genome before the program begins to generate primers, such as removing low complexity regions of the genome. Due to time limitations this was not done. This means that both the number of potential primers and the size of the trie decrease, thus resulting in a faster more efficient program. Another planned feature was to check the primers for nucleotide repeats using regular expression, primers with multiple repeats of the same nucleotide would discarded before genome alignment. In its current state, the program is also severely limited since it only generates primers of a certain length, which means that many potential primers are lost.

References

[1] Albert Au Yeung. (2022). SvartKaffe/data-structures-in-python: needed for reference (v1.0). Zenodo. https://doi.org/10.5281/zenodo.6584175

[2] Cock, P.J. et al., 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), pp.1422–1423

[3] Emery, S., Erdman, D., Bowen, M., Newton, B., Winchell, J., Meyer, R., Tong, S., Cook, B., Holloway, B., McCaustland, K., Rota, P., Bankamp, B., Lowe, L., Ksiazek, T., Bellini, W. and Anderson, L., 2004. Real-Time Reverse Transcription–Polymerase Chain Reaction Assay for SARS-associated Coronavirus. Emerging Infectious Diseases, 10(2), pp.311-316.

[4] García, P., Martínez, B., Obeso, J. and Rodríguez, A., 2008. Bacteriophages and their application in food safety. Letters in Applied Microbiology, 47(6), pp.479-485.

[5] Hatfull, G. and Hendrix, R., 2011. Bacteriophages and their genomes. Current Opinion in Virology, 1(4), pp.298-303.

[6] Kasman LM, Porter LD. Bacteriophages. [Updated 2021 Sep 28]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK493185/

[7] Marcó, M., Moineau, S. and Quiberoni, A., 2012. Bacteriophages and dairy fermentations. Bacteriophage, 2(3), pp.149-158.

[8] Marmur, J. and Doty, P., 1962. Determination of the base composition of deoxyribonucleic acid from its thermal denaturation temperature. Journal of Molecular Biology, 5(1), pp.109-118.

[9] Van Rossum, G. Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace. Available at https://www.python.org/

[10] volpato30. (2022). SvartKaffe/hamming-d-search: Release so i can reference it (v1.0). Zenodo. https://doi.org/10.5281/zenodo.6584083

Appendices

A Appendix A

Code location and GUI information

The code is located in the Github repository named PCR-primers. Clone the repository and run the Main.py file to start the GUI, if you have the required modules (biopython, numpy and tkinter). Press "Choose a file" to load a fasta file, enter the wanted values for ΔT, Tm, max fragment size and min fragments size and press check "Check input values" to validate the values. Finally press "Run the program" to start, once finished, press "View primers" to view the results. There's two "Select" buttons which copies the primers into one string to the clipboard, one button for each window. When the program is running, the GUI freezes which is normal.

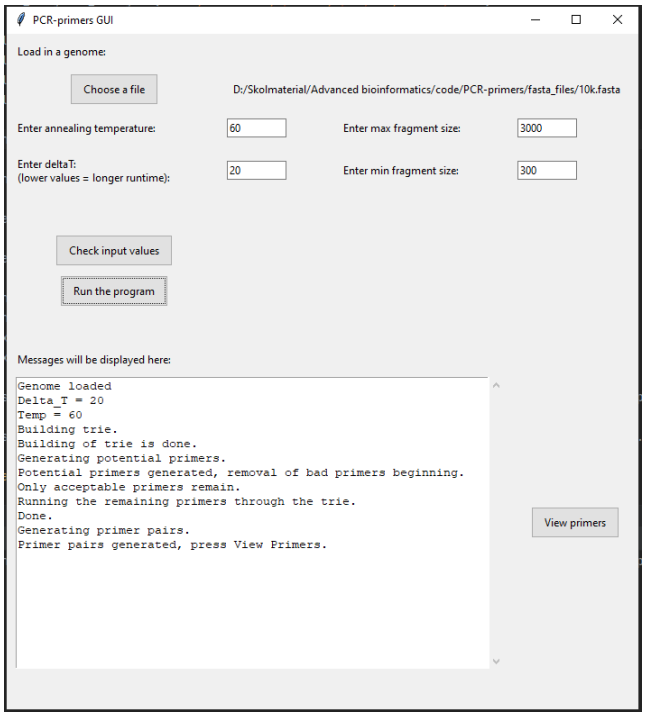

Figure 7: The first window of the GUI after it has finished.

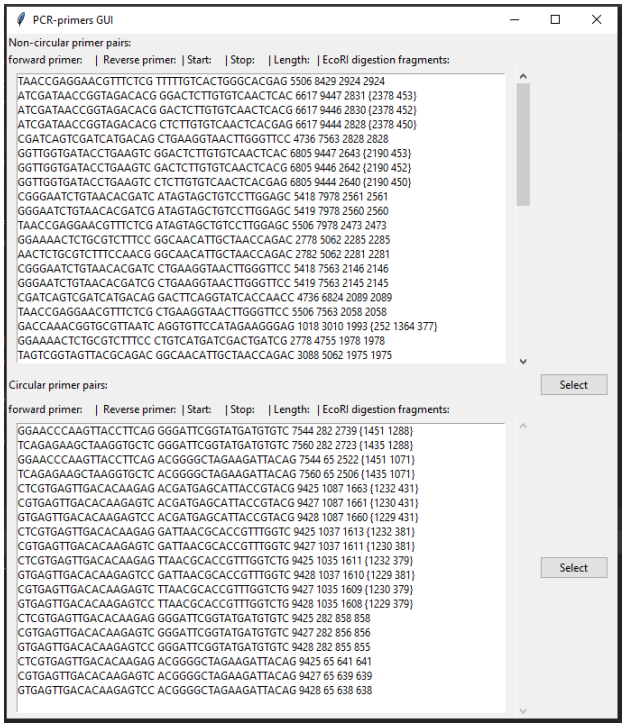

Figure 8: The other window which displays the found primer pairs. This window is shown after the program has finished and the "View primers" button is pressed. The upper window displays normal primer pairs, the lower window displays primer pairs generates a fragment that spans the end as well as the beginning of the genome.