8-8-2021

Comparative DNA Degradation Automation for Identifinders

Jennifer Pfaff
jp9863@rit.edu

Comparative DNA Degradation Automation for Identifinders

Jennifer Pfaff

A Thesis Submitted in Partial Fulfillment for the

Requirements for the Degree of Master of Science of Bioinformatics and Computational

Biology at the Rochester Institute of Technology.

Thomas H. Gosnell School of Life Sciences

College of Science

Rochester Institute of Technology

Rochester, NY

August 8th, 2021

Rochester Institute of Technology
Thomas H. Gosnell School of Life Sciences
Bioinformatics Program

To: Head, Thomas H. Gosnell School of Life Sciences

The undersigned state that Jennifer Pfaff, a candidate for the Master of Science degree in Bioinformatics, has submitted her thesis and has satisfactorily defended it.

This completes the requirements for the Master of Science degree in Bioinformatics at Rochester Institute of Technology.

Thesis committee members:

Name	**Date**
Gary R. Skuse, Ph.D. Thesis Advisor	
Colleen Fitzpatrick, Ph.D.	
Feng Cui, Ph.D.	

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Colleen Fitzpatrick who allowed me to work on this project and supplied the data needed for testing. I also want to recognize her work in the forensic genealogy field and thank her for her dedication to helping others.

I would like to thank all my professors who inspired a love of learning in me, my family for supporting me, and a special thanks to my husband who always believed in me.

Table of Contents

Abstract

The degradation of DNA in forensic genetic genealogy can hinder timely results in victim or perpetrator identification. Identification of an unidentified victim, a “Doe”, involves analyzing the degraded DNA against reference genomes. Organizations like the DNA Doe Project and Identifinders currently manually compares the genomes in a spreadsheet with data input slowing down the investigative process. I propose a program to automate the data input and comparison to increase the speed at which Identifinders can continue their research into identifying victims. Comparative DNA Degradation Automation (CDDA) automates the process of creating a simulated degraded DNA profile in less than a second on average. This program could be used not only by organizations like Identifinders and the DNA Doe Project, but also used in future research where natural DNA decomposition is not possible.

Introduction

Forensic work with DNA has been used in criminal cases for over thirty years (Arnaud, 2017) to help solve crimes by helping identify suspects for law enforcement, but it can also be used to help identify the victims of these crimes. Genetic fingerprinting, or DNA fingerprinting, is a technique that shows an individual has unique DNA sequences that could be used in identification (Jeffreys et al., 1985). Microsatellites, or short tandem repeats (STR) of about 100bp long, were used in 1990 to create DNA fingerprints from bone samples that had a small sample size (Zagorski, 2006). STR typing was used to aid in the identification of deteriorated skeletal remains (Jeffreys et. al, 1992). DNA fingerprinting with STR typing showed that this technique can be used on degraded and small sized DNA samples.

In England a murder case used DNA profiling with STR typing in 1986. Showing that two samples could be from the same individual, the police sought to confirm the confession of a suspected murderer. With no national DNA data banks at the time, the suspect's DNA was compared to the DNA found on victims and found that while the victims were both killed by the same person, it was not the current suspect (Cobain, 2016). The test was repeated and the results showed the consistency of DNA fingerprinting and proving a suspect's innocence.

In the United States, DNA fingerprinting helped solve a case just one year later (Calandro et al., 2005). A defense attorney argued that the DNA testing was unreliable and that there could be errors in the lab work but experts testified that the DNA fingerprinting in the case was reliable (New York Times, 1988). There have been standards that scientific evidence has to meet to be admissible in courts, as in 1923 the Frye standard was created where it was decided that scientific evidence provided must be "sufficiently established to have gained general acceptance in the particular field in which it belongs." (D.C. Cir., 1923).

It was not until 1994 that a standard was created for DNA testing, and with it a national database. The DNA Identification Act of 1994 set up state labs to perform DNA fingerprinting using a standardized procedure, going beyond the 1923 Frye standard (Edwards, 1994). This new act allowed for the creation of a national index where the government can now store DNA identification of criminals, the results of the analysis of samples from crime scenes and from unidentified human remains. On a federal level, the database and software used for criminal justice work is the Combined DNA Index System (CODIS), but CODIS is a generic term for the whole system. CODIS is made of different databases and tools, one is the National DNA Index System (NDIS) which is the part of CODIS that is what most people associate with DNA searches by the FBI, and there is one for states (SDIS) and at a local level, such as counties (LDIS).

The DNA information in the profiles is made up of short tandem repeats,and until 2017, 13 different loci were used in profiles. As of January 1st, 2017, there are 20 core STR loci. This expansion was to increase accuracy, international compatibility, and discriminatory power (Hares, 2015). As of June 2021, the NDIS contains over 14 million profiles and has helped over half a million cases according to the FBI (FBI n.d.).

While the CODIS program has helped many cases, there are still many cold cases that need to be solved. The rate of cold cases being solved has dropped to around 60% from 80% since the 1960's (Martin et. al, 2020). A new method to solve cases based on Single Nucleotide Polymorphisms (SNPs), forensic genetic genealogy, has become a more common technique to help investigations. Forensic genetic genealogy can be used in forensics to help track familial relations using DNA matches. Using a SNP profile, a family tree can be built using information from GEDmatch, which is a website that allows users to upload DNA files from different direct-to-consumer genetic testing companies and match genetic profiles to possible relations (DNA Doe Project, 2018).

In December of 2019 GEDmatch announced that Verogen had acquired it. Verogen is a company that works with forensic laboratories by supplying products and services based around Illumina products (Genomeweb, 2019). Verogen CEO said the acquisition was to improve the tools available and make the website user friendly (Taylor, 2019).

Though the idea of a genetic genealogical database is new, the idea of a genealogical database is not a new concept. In 1894 the Genealogical Society of Utah was founded by the Church of Latter-Day Saints to keep genealogical records (Durrant, 2010). The website FamilySearch.org was created in 1999 by the Church of Latter-Day Saints, and just three years later posted the 1880 census for the United States to help members trace their family tree. FamilySearch, as of 2020, has over 13 million users (FamilySearch, 2021). This website does not allow users to submit their own DNA profiles, but has a guide on their website explaining some of the process and linking to other companies' websites for interested users.

Direct-to-consumer (DTC) at home testing kits has become popular with over 30 million people purchasing kits by 2020 (Molla, 2020). As Ancestry and 23andMe both allow downloads of the raw data, potentially millions of entries could be uploaded to sites like GEDmatch if DTC users choose to upload their data.

DNA home testing kits are based on SNPs. The move to use SNPs over STR typing in home testing kits has been for a few reasons such as SNPs are abundant in genomes, have lower mutation rates than STR, and having variation at a SNP can be linked to phenotypes (Gray et al., 2000). As next-generation sequencing technology advances, the cost and time taken to process genotyping has decreased (Xiao et al., 2016). According to the International Society of Genetic Genealogy, 23andMe tests for fewer SNPs than Ancestry (*table 1*) (Janzen, 2021). There are millions of SNPs in a person's genome; some can be unique to a person or in groups of people (NIH MedlinePlus, n.d.). The variations that SNPs show can be used in at-

home genetic testing kits as markers for possible traits such as appearance or health (23andMe, n.d.). Forensic genetic genealogists have used GEDmatch's database to identify a John or Jane Doe or the perpetrator of a violent crime.

Table 1. *Number of SNPs markers in Ancestry and 23andMe*

	Autosomal	X	Y	Mitochondrial
Ancestry	637,639	28,892	1,691	263
23andMe	630,132	16,530	3,733	4,318

The organization Identifinders was started in 2008 while the DNA Doe Project is a more recent organization formed in 2017. Identifinders and the DNA Doe Project are contacted by law enforcement agencies to help their cases using genetic genealogy. Once a case is accepted, the DNA is genotyped by whole genome sequencing. The whole sequenced DNA profile is not needed so the results are converted to a file that can be read by GEDmatch by creating a DNA profile with DNA markers that GEDmatch recognizes. This file is then uploaded to GEDmatch to check for other profiles that have similarities which could lead to possible relatives. Using these methods to search for relatives, along with informational archives and public data, can lead to a tentative identification.

A press release from the DNA Doe Project on one of their early successes stated that after a few hours of using GEDmatch after the profile's upload, possible matches were found

with a possible first cousin even though the sample was highly degraded. Detective Steven Hickey said the case was forensically groundbreaking (Augenstein, 2018).

As of 2020, Dr. Colleen Fitzpatrick has left the DNA Doe Project that she had co-founded to focus on her original company, Identifinders. Identifinders has had many high-profile successes in identifying remains as well. Some notable cases Identifinders has worked on are the remains of the crash of Northwest Flight 4422, the “Unknown Child of the Titanic", and they have helped police departments identify suspects in violent crimes (Identifinders, 2021).

Identifinders has access to an assortment of tools that they use in their work. One tool Dr. Fitzpatrick used included spreadsheets to simulate degradation based on a non-degraded DNA profile. This tool allows comparison of results from degraded and non-degraded files to test validity of results from GEDmatch . This way to simulate degraded DNA was time consuming and could be improved. I have proposed a program to automate degradation simulation using parsing and hashmap comparison in Java.

Methods

Overview

The program Comparative DNA Degradation Automation (CDDA) was created in core Java and Swing, and uses the WiX toolset to build an executable package for native system support on Windows systems. The basic flow of the program is to parse, compare, then output the results (Figure 1). CDDA first parses two user-submitted files using a class that stores the information in string hashmaps. A third hashmap is then created to store the matching information from the two user submitted files. This third hashmap is saved via output to the user’s hard drive in the location from which the program was run. The program was written with

accurate results as the top priority, with speed being a secondary concern. Once accuracy and speed were within acceptable ranges a graphical user interface was added to increase accessibility of the software. WiX was required as an efficient method of compiling an executable file from Java (Mensching, 2021).

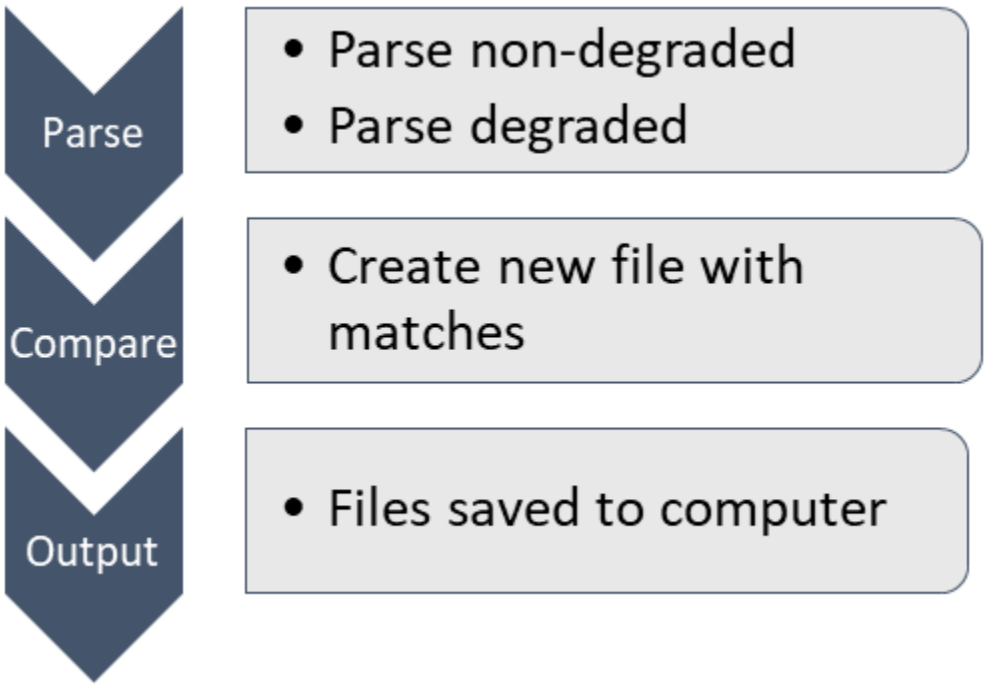

Figure 1*. Process flow of the Comparative DNA Degradation Automation program.*

DNA Profile Files

Dr. Colleen Fitzpatrick from Identifinders supplied full DNA profile files from her previous work for use in testing the software. Files were provided under a non-disclosure agreement for privacy reasons. All references to any personal demographics were removed from the project. Hereafter the files are referred to as the FullDNA file, the DegradedDNA file, and the ExpectedResults file.

I/O

The user interface version of the CDDA program requires the user to manually select the file location on the hard drive using the Swing toolset's file browser. Because of the way the file browser functions, it is not necessary to verify a file exists after user selection in the GUI as the user will be unable to select a file that does not exist. Once the file is selected, the name of the file is displayed above the selection button so that the user can visually verify that they have selected the appropriate project file. This is true of both the FullDNA file and the DegradedDNA file.

The original command line version of the CDDA program requires the user to type in the file location and then runs the code's fileCheck function to ensure the input file exists before it can move on to parsing (Figure 2).

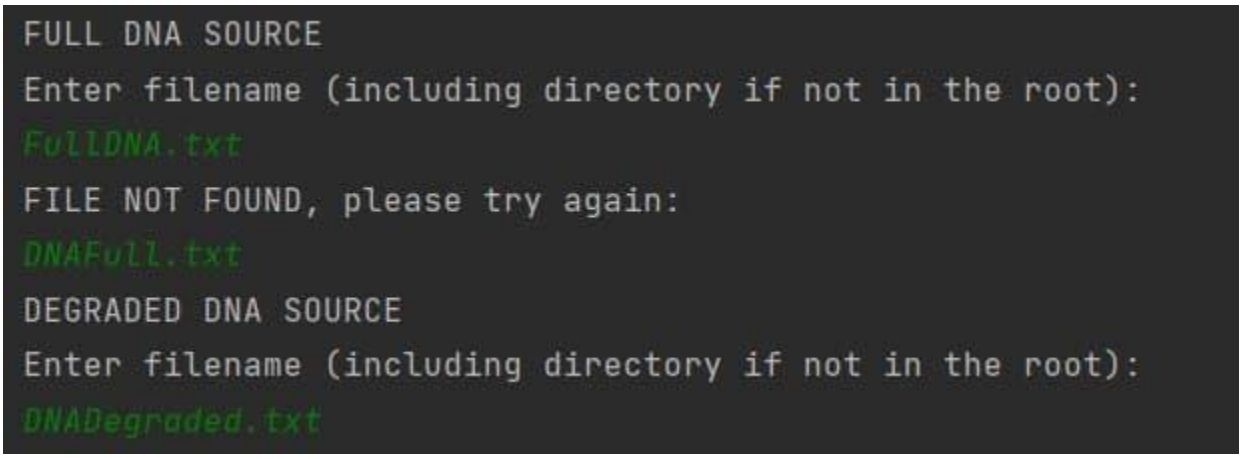

Figure 2. *CDDA gives an error when the input file is not found.*

The user input is pulled using Java's scanner tool, and in the event of a mistyped file name or missing file the fileCheck method will notify the user that their file couldn't be found and prompt them to enter their filename into the scanner again.

The output file name can be any that the user desires, and will default to being created in the working directory of the program.

Parsing

The next step was to take in the DegradedDNA profile and the FullDNA files and extract pertinent information by parsing the necessary data. The DNA reports used in this project were generated by a direct-to-consumer genetic testing company. The files from this company had excess information that did not need to be processed for the purposes of CDDA. Only the relevant information was needed: the RSID numbers, chromosome, position on chromosome, and the genotype. The information not needed was formatted in a way similar to commenting in code so that the lines started with a pound sign, or hashtag mark. These irrelevant lines were located at the top of the file. This information was searched for first and removed from the file parsing.

Before editing out this information, the DNA files were read into a new file in the program so as to preserve the original files. The Java BufferedReader class was used to wrap, or work with, the FileReader class to ensure the process was efficient to keep the program from taking a long time to run and any line that started with a pound sign was ignored while the rest of the file was read into a split string format using string arrays. Each line represented one sequenced SNP and each line was broken into the RSID, chromosome, location, and genotype columns. The files were mostly uniform, but there were times that multiple spaces were used instead of a tab, so parsing with a split at a tab would not work. Therefore, the split was used on any whitespace in the line that was larger than 2 spaces. Once broken up into sections by whitespace, the pieces were sorted into two parts, a key and a value.

Java hashmaps require a single key value for each entry into their map, and the rest of the data is the value associated with the key. The information in the files had two unique identifiers that could be used as a key to organize the information, the RSID numbers and the chromosome position. Being that the RSID numbers are already considered an identification number, and that it was listed before the chromosome on the line, the RSID numbers were used

as the key value for the hashmaps. The rest of the information connected to the RSID was stored in the hashmap as one entity in the value. The parsing class did this process to each line in the DNA files until a line was read that was null. This indicated that the full file had been read and all data had been parsed, leaving the program with two hashmaps; one with the FullDNA data, and one with the DegradedDNA data.

Simulating DNA Data

With the two hashmaps populated by the parsing process, CDDA can begin creating a third hashmap that contains a simulated version of the FullDNA data as degraded dna using the FullDNA and DegradedDNA as sources. In most speed tests, it took less than a full second for the code to find matching RSIDs between the two data sources and piece together a simulated degraded DNA hashmap using those matched tags.

The program uses a simple loop that takes advantage of Java hashmap's abilities to compare key values for exact matches very quickly. The matching keys are kept along with the associated values, and those are stored in the third hashmap. The result of this efficiency is that the brunt of the program's work is usually done in a fraction of a second, leaving the software with a third and final hashmap ready to be outputted to the user.

Writing the Output

At this point in the process the simulated data that has been created, and a blank file that represents the desired output of the software, are used to output a tab delimited version of the final results from the software. By passing the Java class writeFile method a blank file, the user selects what the output will be named based upon their individual needs. It is advantageous to select a name that ends in ".txt" as it allows the file to be opened in any text

editor by default, but any name is permissible so that the output can match the user's desired preference.

In the case of the output file the program does not take into account the possibility of a pre-existing duplicate file in the working directory. This means that if the user inserts the name of a file that already exists in the directory, that file will be overwritten when the new output file is created.

Statistics and Prints

In the console version of the software, after the output file has been written, the software runs some minor statistical data for the user (Figure 3). The total RSIDs were counted from the full DNA sample as the software ran. The number of matching RSIDs between the full DNA and the Degraded DNA were also counted. This allows the program to determine a percent of matching RSIDs to the full source file, information that might be of immediate interest to someone running the CDDA program.

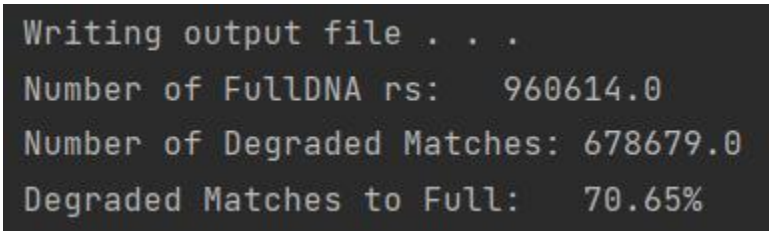

Figure 3*. The printed statistics at the end of a completed CDDA run.*

Rounding

Often when calculating a percentage the math leaves one with a long decimal number that is precise, but also difficult to read. The rounding method in this code is used to appropriately round a percentage to a given decimal place to give the user an easier to digest number to work with. This work is handled by Java's built-in math module, particularly the BigDecimal and RoundingMode classes.

Graphical User Interface (GUI)

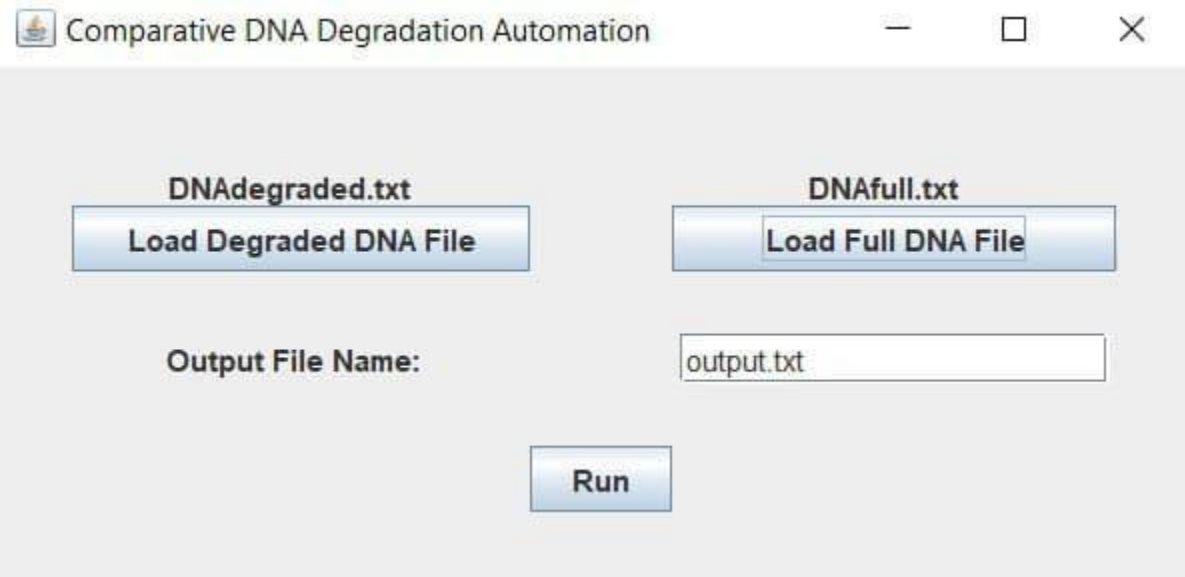

Figure 4. *The graphical user interface for CDDA.*

The GUI for CDDA consists of a list of visual components, a list of ActionListeners, and a storage component for static variables. CDDA's graphical interface is built using the Swing library for Java. Swing is a graphical library that contains prebuilt tools for creating visual elements that allows a programmer to build an easy and accessible frontend for their software. The visual components used in CDDA consist of a popup window made from JPanel that contains three JButtons, three JLabels, and a JFormatedTextField (Figure 4). The buttons are interactable objects that the user can press, two of which open JFileChooser, a Swing element that allows a user to browse their computer file system for a file (as in this case) or a file directory. The third JButton runs the software's primary function, which is the production of a simulated degraded DNA sample. Two of the JLabels are used to identify the file selected by the user so that they can verify they have selected the appropriate file. The other JLabel informs the user that they can insert their desired output file name in the JFormatedTextField that is directly adjacent to the label.

The functions for the GUI elements are controlled by a set of ActionListeners, which are

functions that “listen” for the input of the user, and do something based upon that input. In this case, the two JButtons that let the user select files are triggered when pressed, and those functions activate JFileChooser.

ActionListeners are non-static objects in Java. As such, to store information gathered inside of them in such a way that we can use it outside of the listeners, we must create a static instance of our input files that the ActionListeners can modify. The DNAWork method in the GUI code does this. The static variables within the method can be accessed and modified by directly calling DNAWork, so we can use the files stored there inside of any ActionListener.
The ActionListener attached to the “Run” button has been set to gather the full DNA file, the degraded DNA file, and the output filename so that it can send this information to our main simulation processing method.

Building For Windows

In order to create a version of the software that can be run natively in Windows, the installation of the WiX toolset (Mensching, 2021) is necessary. WiX makes building an executable Windows file from Java source code more efficient and can be set to allow integrated development environments (IDEs) to build new versions of the executable file with every new build of the software. This allows for quick revisions and testing of the code on multiple systems.

Building for Command Line

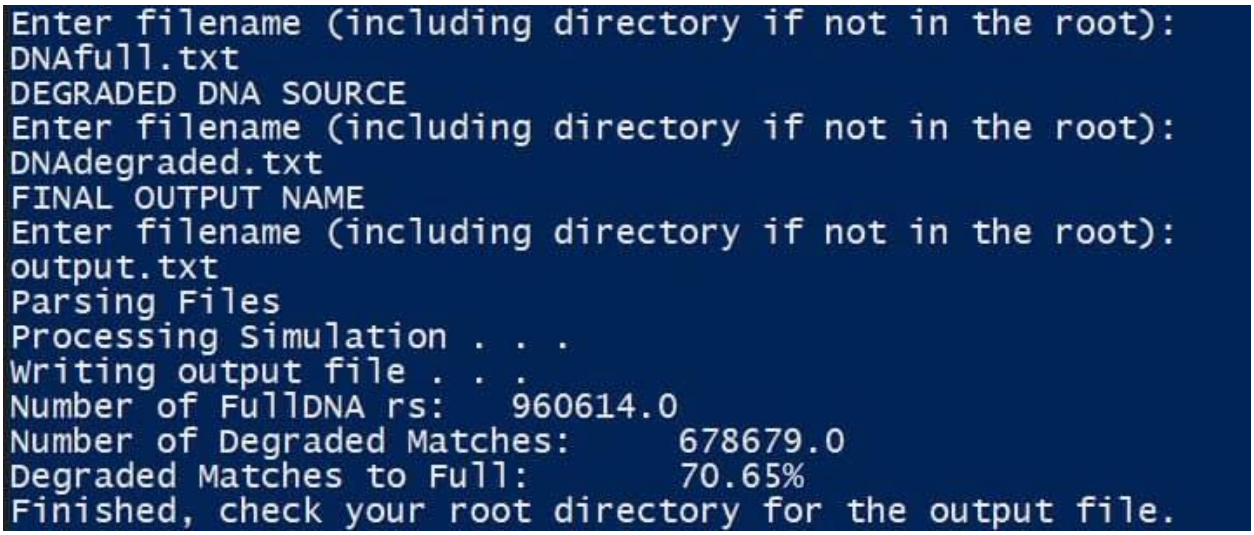

Figure 5. *CDDA run using PowerShell.*

Running CDDA using a GUI is not needed as it can also be run from the command line or with Windows PowerShell (Figure 5). The Main.java file can be used to run the code from any directory and is especially useful in cases where storage may be an issue, or where the user might be working from a Unix based operating system such as Linux or Apple's OSX. In these cases, as long as the system has a version of the Java SDK that is 16 or higher, a runnable class file can be compiled from the source Main.java file. The exact method of doing this will vary by operating system, but usually only takes a single command from the system's console.

Results

Speed

Three different computers ran the Comparative DNA Degradation Automation (CDDA) program, a desktop and two laptops. Each computer had different specifications (Supplemental data). All three machines ran the program with the looping method and the final hashmap version.

Table 2. *Average time taken for CDDA to complete a run in seconds.*

Computer	Loop Method (s)	Hashmap Method (s)
Desktop	47566.447	0.236
Lenovo Thinkpad	47128.510	0.394
Surface Pro	44849.298	1.086

The first version of CDDA that ran with a for-loop method took over 40,000 milliseconds on average on each of the three computers which came to 12 to 13 hours of run time per simulation (Table 1).

The second version of the program used matching key values in Java hashmaps which reduced the run time significantly. Of the three machines used, only one machine had a run time longer than one second. On the Surface Pro computer the first two runs took about one second each. The program was run 5 more times and those remaining runs were similar in time to the other two machines. Removing the two outlying large times, the Surface Pro average run time with the hashmap method was 388.25 milliseconds (Supplemental data).

Accuracy

The output file generated by CDDA was compared with the ExpectedResults file supplied by Identifinders. The results matched in that all the RSID that were in both files were in the output file. The order of the matches do not descend in the same order as the ExpectedResults file due to the hashmap sort method. The output line order is consistent with each run and the file can be saved as a .csv or imported into a spreadsheet program and then sorted to the preference of the user.

Discussion

Efficiency

Manually comparing two files with between 600,000 and 1,000,000 lines of data and then preparing a file of the matches can be time consuming. The first version of the program that ran for around half a day would still be more efficient, but the second version of CDDA using hashmaps is significantly faster in comparison. With the first version taking a long time to compute, it left the program vulnerable to occurrences that would have stopped the program such as the computer restarting or a power outage, which would lose a large amount of work done. With CDDA's faster method the program was completed almost instantly and frees up a lot of time that would be used manually comparing files that can now be used on other aspects of their projects. As seen in the test runs, the speed at which the program completes can vary from run to run and by the computer specifications.

Working on projects with DNA degradation or a case at Identifinders can have users with varying technical skills running CDDA. To ensure the program was approachable no matter the technical background, a GUI was created. The program can be run by command line as well to fit into a pipeline as needed. To help with the approachability, the program can also be run as an executable file. CDDA does not need a formal install and registry access and instead can be used from the directory it was placed in.

Currently CDDA overwrites output files without asking the user if that is their intent. That is something that could be addressed in a future build. There are a few different methods for handling this problem. Checking to see if the file exists and rejecting the file name if it does is the most obvious method for dealing with this situation. Another way to handle this would be to automatically rename the file when the software determines a file with that name already exists. Appending a number to the end of the file name would work, though the file would then have to

be checked again and another number would need to be appended in the event of a further duplicate. Another solution would be to read the system time and date from the computer and add that to the output name given by the user, allowing a timestamp to track when the program had completed.

The outputs of CDDA had all the data that should be in the final output via the ExpectedResults file. Multiple runs of the provided DNA profile dataset have shown consistent output. Though all the data is contained in the output file, the order the data is output in the file differs from the ExpectedResults file. Further modifications to the program could adjust the final output to sort the RSID values the same way the ExpectedResults file was formatted. Opening the file in a spreadsheet program can adjust the way the results are sorted and the RSID values can be changed to ascend or descend in the order the user wishes.

Compiling and Privileges

Currently there are three versions of the software. For Windows users there is a compiled executable file that can be copied into an empty directory and run from anywhere. This version has a GUI. For Linux and Mac users there is a version of the file with a .java extension that can be compiled and run from the system console. This version is the command prompt only version of the software with no GUI. The third version is the IDE version of the software where the source code can be run in a user's IDE.

To create the Windows version of the software WiX was employed. WiX is a toolset designed with porting Java applications to a Windows system executable as the primary focus. The software was installed on the working machine, and then a "platform specific package" version of the program for Windows was selected as an additional output when building CDDA from an IDE. WiX creates an installable executable version of the software that can be added or removed using the built-in Windows installation wizard. Once installed, CDDA can be moved

and used from any directory, but it defaults to installing in C:/Program Files/DNA Gui/ on a Windows system.

To run the software from this install directory it is imperative that the user right-click the file and select “Run as Administrator” because “Program Files” is a protected folder and if CDDA is run without administrative privileges it will not have the file system access rights it needs to create the output file. Moving the program to an unprotected folder anywhere else on the user’s computer will allow the software to be run without administrative privileges.

Future Work

Batches

Working on large batches of files to improve efficiency would be a priority going forward with the software. The ability to set the software to run on a list of many jobs will allow for users to move on to other tasks while CDDA takes care of what was once time consuming work. To achieve this there are some modifications necessary to the current code that could be done in future work, though many of the current program’s methods can currently be adapted to running batches without needing to be changed in any way. Instead of taking two files as input, it would be most efficient to take in two directories. One of the directories would store the Full DNA sources, and the other would store the Degraded DNA sources, with files named in such a way that they would be directly associated. For example, the files in the Full DNA source folder might look like this: Case01Full.txt, Case02Full.txt, Case03Full.txt. The Degraded DNA sources would then look like this: Case01Degraded.txt, Case02Degraded.txt, Case03Degraded.txt. This will allow the software to easily pair up the appropriate files by the file names as it runs through the directories.

The output files could be automatically named so that the user does not have to input a name for each out from the individual pair of files. With automation the file name could be either a combination of the source file names, or a date and time stamp appended to a file format, or some combination of these two. It would be a high priority in batch work to ensure the user could easily match the input pairs to the output file.

One advantage to reading files in from a directory is that the program would not need to verify each file exists before moving on. Scanning the directory and cycling through the files it contains eliminates that need but it also introduces new complications. Instead of verifying a file exists, we now need to check that the file contains the proper information and format, and CDDA will need to make sure each file matched up with the appropriate file from the other folder. For instance, Case01Full needs to be paired to Case01Degraded or the results would be incorrect and the program would have to be run again. In the event that there are files in the folder that are not part of the batch that needs to be run, then those files will need to be ignored through the use of data verification with parsing.

Parsing provides an opportune time to scan the contents of a file to see if it contains the necessary data for CDDA to process the simulation. In its current version CDDA assumes that you are selecting valid data files for it to parse, but for the purpose of running batches of jobs it would be appropriate to create a parsing process that looks for exactly what CDDA needs to work, and rejects files that don't contain that data. For example, if the parser isn't locating RSID values, then it should reject that file and tell the software to move on to the next file. Additionally, if the parser is processing a file called "Case01Full" and then a file called "Case04Degraded.txt," it should be programmed to try and compare those file names so it only processes matches. Part of this future work could include giving the user the ability to control whether or not it can reuse samples from one folder or another in case there is a reason to run a group of degraded files versus a single full file, such as "Case01Full" paired with each different degraded file.

The graphical user interface for the current application will need to undergo several changes before it can be used for batch files. An option to switch between batch mode and individual jobs could make the process easier than creating two separate versions of the program. This could be created as either a radial button or a dropdown menu. Activating batch mode would switch the file loading buttons to directory loading buttons, allowing the user to choose input folders where their files would be located. Instead of allowing them to name the output directly, giving the user a dropdown menu with different naming convention options would be a user-friendly way of automating the output process. A button will be necessary as well to allow the user to pick the desired output folder, though defaulting it to the root directory could also be an option.

Automation

Beyond batch jobs, automation of processes could make the handling of raw files faster and more efficient. For future work, it may be possible to create a version of CDDA that handles the processing of files with no more direct input necessary from the user. By modifying the CDDA program so that it stays in system memory and periodically scans certain directories, the program could automatically work in the background. Scanned files could be read and sorted by the app before processing and outputting the necessary file into a “finished” folder so they are not automatically processed repeatedly.

The program could log the process to a file so that it could be periodically checked or referenced by the user. Files could be dynamically named based upon the source files and the date or time of processing, and the finished output could be stored as separate files, a combined file, or packaged into a storage container like a zip archive for keeping all the files together. The only input that would be necessary for the user is dropping the files into the appropriate folders. CDDA could be set up in a way that the user could adjust the file directory paths as needed.

Multithreading

After batch jobs and automation are built into the software, the next upgrade for the software in the future could be to include multithreaded tasking. If the software was to run multiple jobs at one time, then it could more efficiently finish the work if it took advantage of the modern CPU's ability to create multiple threads of work. Most modern CPUs have at least two cores, and often those cores are capable of running additional work on their own separately. Many high end machines can run 16 operations at a time, which could be a substantial reduction in calculation times.

Currently CDDA can finish a single run in under a second most of the time, but with a larger workload such as thousands of samples to run through the timing can greatly increase. Multicore and multithreaded processing allowing more than one file at a time could show a significant increase in speed versus single core operation.

Building in batch and automation support would be a priority in future work, but threaded workloads are a logical next step once that has been implemented. Keeping split workloads in mind as the batch processes are programmed into CDDA will make for an easier transition to this kind of computing in the future.

Conclusion

In conclusion, Comparative DNA Degradation Automation (CDDA) successfully automated the process of simulating degraded DNA profiles from a provided full DNA profile. The completed program processes a pair of DNA profiles in less than a second on average. Use of the program by groups that work in forensics or genetic genealogy, such as Identifinders, can free up time to focus on different parts of their projects. The program could also be used in future research when waiting for a DNA sample to degrade and then sequencing the profile

would take too much time. Future work can continue to improve the efficiency of the program and allow for more data to be processed at once.

References

23andMe. *23andMe - Genetics 101: What Are SNPs?* https://www.23andme.com/gen101/snps/. Accessed 7 Aug. 2021.

Arnaud, Celia. “Thirty Years of DNA Forensics: How DNA Has Revolutionized Criminal Investigations.” *Chemical & Engineering News*, 2017. *cen.acs.org*, https://cen.acs.org/articles/95/i37/Thirty-years-DNA-forensics-DNA.html.

Augenstein, Seth. “‘Buck Skin Girl’ Case Break Is Success of New DNA Doe Project.” *Foresnic Magazine*, 12 May 2018, https://web.archive.org/web/20180512010806/https://www.forensicmag.com/news/2018/04/buck-skin-girl-case-break-success-new-dna-doe-project.

Calandro, Lisa, et al. “Evolution of DNA Evidence for Crime Solving - A Judicial and Legislative History.” *Foresnic Magazine*, 6 Jan. 2005, https://web.archive.org/web/20170502224346/https://www.forensicmag.com/article/2005/01/evolution-dna-evidence-crime-solving-judicial-and-legislative-history.

Cobain, Ian. “Killer Breakthrough – the Day DNA Evidence First Nailed a Murderer.” *The Guardian*, 7 June 2016, http://www.theguardian.com/uk-news/2016/jun/07/killer-dna-evidence-genetic-profiling-criminal-investigation.

D.C. Cir., 293 F. 1013. *Frye v. United States, 293 F. 1013*. 1923, https://casetext.com/case/frye-v-united-states-7.

DNA Doe Project. “Buckskin Girl Press Release.” *DNA Doe Project Cases*, 11 Apr. 2018, https://dnadoeproject.org/press-release-for-buckskin-girl/.

Durrant, George. *Genealogical Society of Utah - The Encyclopedia of Mormonism*. 18 May 2010, https://eom.byu.edu/index.php/Genealogical_Society_of_Utah.

Edwards, Don. *H.R.829 - 103rd Congress (1993-1994): DNA Identification Act of 1993*. 30 Mar. 1993, https://www.congress.gov/bill/103rd-congress/house-bill/829. 1993/1994.

FamilySearch. *Privacy Policy — FamilySearch.Org*. 6 Apr. 2021, https://www.familysearch.org/legal/privacy.

F.B.I. “CODIS - NDIS Statistics.” *Federal Bureau of Investigation*, https://www.fbi.gov/services/laboratory/biometric-analysis/codis/ndis-statistics. Accessed 1 Aug. 2021.

Genomeweb. “Forensic Genomics Firm Verogen Acquires Genealogy Website GEDmatch.” *GenomeWeb*, 2019, https://www.genomeweb.com/sequencing/forensic-genomics-firm-verogen-acquires-genealogy-website-gedmatch.

Gray, Ian C., et al. “Single Nucleotide Polymorphisms as Tools in Human Genetics.” *Human Molecular Genetics*, vol. 9, no. 16, Oct. 2000, pp. 2403–08. *Silverchair*, https://doi.org/10.1093/hmg/9.16.2403.

Hares, Douglas R. “Selection and Implementation of Expanded CODIS Core Loci in the United States.” *Forensic Science International: Genetics*, vol. 17, July 2015, pp. 33–34. *DOI.org (Crossref)*, https://doi.org/10.1016/j.fsigen.2015.03.006.

Identifinders. *IdentiFinders - Case Histories*. http://www.identifinders.com/histories.html. Accessed 15 Apr. 2021.

Janzen, Tim. “Autosomal DNA Testing Comparison Chart - ISOGG Wiki.” *International Society of Genetic Genealogy*, 22 June 2021, https://isogg.org/wiki/Autosomal_DNA_testing_comparison_chart.

Jeffreys, Alec J., Maxine J. Allen, et al. “Identification of the Skeletal Remains of Josef Mengele by DNA Analysis.” *Forensic Science International*, vol. 56, no. 1, Sept. 1992, pp. 65–76. *DOI.org (Crossref)*, https://doi.org/10.1016/0379-0738(92)90148-P.

Jeffreys, Alec J., John F. Y. Brookfield, et al. “Positive Identification of an Immigration Test-Case Using Human DNA Fingerprints.” *Nature*, vol. 317, no. 6040, Oct. 1985, pp. 818–19. *DOI.org (Crossref)*, https://doi.org/10.1038/317818a0.

Martin, Eric, et al. “Serial Killer Connections Through Cold Cases.” *National Institute of Justice*, 15 June 2020, https://nij.ojp.gov/topics/articles/serial-killer-connections-through-cold-cases.

Molla, Rani. “Why DNA Tests Are Suddenly Unpopular.” *Vox*, 13 Feb. 2020, https://www.vox.com/recode/2020/2/13/21129177/consumer-dna-tests-23andme-ancestry-sales-decline.

New York Times. *Rapist Convicted on DNA Match - NYTimes.Com*. 6 Feb. 1988, https://web.archive.org/web/20171106131127/https://www.nytimes.com/1988/02/06/us/rapist-convicted-on-dna-match.html.

NIH MedlinePlus. *What Are Single Nucleotide Polymorphisms (SNPs)?: MedlinePlus Genetics*. https://medlineplus.gov/genetics/understanding/genomicresearch/snp/. Accessed 7 Aug. 2021.

Taylor, Michelle. “Verogen CEO: ‘GEDmatch Will Be Improved, Not Changed.’” *Forensicmag*, 2019, http://www.forensicmag.com/559058-Verogen-CEO-GEDmatch-Will-Be-Improved-Not-Changed/.

Xiao, Shijun, et al. “Whole-Genome Single-Nucleotide Polymorphism (SNP) Marker Discovery and Association Analysis with the Eicosapentaenoic Acid (EPA) and Docosahexaenoic Acid (DHA) Content in Larimichthys Crocea.” *PeerJ*, vol. 4, Dec. 2016, p. e2664. *PubMed Central*, https://doi.org/10.7717/peerj.2664.

Zagorski, Nick. “Profile of Alec J. Jeffreys.” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 24, June 2006, pp. 8918–20. *PubMed Central*, https://doi.org/10.1073/pnas.0603953103.

Supplementary Data

Testing Computer Specifications

Supplementary Table 1. *Desktop computer specifications*

Processor	AMD Ryzen 5 3600 6-Core Processor 3.60 GHz
Ram	16.0 GB
System Type	64-bit operating system, x64-based processor

Supplementary Table 2. *Lenovo Thinkpad computer specifications*

Processor	Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz 2.59 GHz
Ram	8.0 GB
System Type	64-bit operating system, x64-based processor

Supplementary Table 3. *Surface Pro 4 computer specifications*

Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz
Ram	4.0 GB
System Type	64-bit operating system, x64-based processor

Program Run Times

For-loop method

Supplementary Table 4. *Run time for the for-loop version of the program in milliseconds*

	Desktop	**Lenovo**	**Surface**
Run 1	47566447	47128510	44849298

Run 2	47566447	---	48353074

The Lenovo was not available for a second run of the program.

Hashmap Method

Supplementary Table 5. *Run times for the hashmap version of the program in milliseconds*

	Desktop	**Lenovo**	**Surface**
Run 1	255	254	3348
Run 2	249	363	4409
Run 3	252	685	282
Run 4	206	466	391
Run 5	245	316	406
Run 6	201	461	251
Run 7	202	436	485
Run 8	258	341	328
Run 9	247	312	531
Run 10	245	309	432

Instructions for Compiling CDDA in Windows

For running this from Windows the commands are as follows:

1. Open a command prompt by either typing “cmd” in the windows search bar and clicking on the Command Prompt app, or open power shell by using File Explorer to browse to the directory containing Main.Java and right-clicking while holding down shift. This will bring up a menu that has the option “Open PowerShell window here.”
2. Make certain the command prompt is in the same directory as your Main.java file. If it is not, you can use the “cd” command to change directories such as “cd yourDirectory”.

3. Type “javac Main.java” and press enter. Wait for compiling to finish. The command line will move to a new line when it is finished.
4. Type “java Main” and press enter again. This will start the software. Follow the on-screen prompts from this point.