Automating Intron Landscaping For Fungal Mitochondrial Genomes

by

Tanya K. Pawliszak

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
December 2024

Thesis advisor

Olivier Tremblay-Savard

Author

Tanya K. Pawliszak

Automating Intron Landscaping For Fungal Mitochondrial Genomes

Abstract

Intron landscapes in mitochondrial genomes play a crucial role in the regulation of gene expression and overall mitochondrial function. However, the complexity and variability of intron structures across different fungal species present significant challenges for manual annotation and analysis. This thesis presents an automated approach for intron landscaping in fungal mitochondrial genomes, leveraging advanced computational tools to streamline the identification, characterization, and classification of introns. By integrating a combination of computational tools that are used in different areas of the identification of Group I Introns, my pipeline, HMMER Infernal Pipeline (or HIPipeline), enhances the accuracy and efficiency of intron detection, significantly reducing the need for labor-intensive manual curation. Indeed, our results on 25 mitochondrial genomes of fungi has shown high levels of sensitivity and precision for the positions of the introns. We validate the approach using a diverse set of fungal mitochondrial genomes, demonstrating its robustness in capturing intron subtype identification and positions. This automation not only accelerates the annotation process but can also facilitate comparative genomic studies and functional analyses of mitochondrial introns across different fungal species.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my academic supervisor, Dr. Olivier Tremblay-Savard, for his invaluable advice, wisdom, support and guidance throughout my M.Sc. thesis journey.

I would like to thank my internal examiner, Dr. Carson K. Leung, and my external examiner, Dr. Georg Hausner from Microbiology, for their insightful comments and suggestions during my M.Sc. thesis journey. Also thanks Dr. Shaiful Chowdhury for chairing my M.Sc. thesis oral defense.

I would also like to thank all of the students in the Bioinformatics Lab for their attentiveness, advice and feedback throughout my M.Sc. thesis journey and for helping me become a better public speaker. I would like to thank Alvan Wai for his help in this research project.

I would like to thank Anthony for the encouragement and support during my M.Sc. thesis journey. Finally, I am profoundly grateful to my mother and brother for their belief in me and their encouragement, which motivated me to complete my M.Sc. thesis.

This thesis is dedicated to my family and the people who are close to me, whose unwavering support and encouragement have been instrumental throughout my M.Sc. journey. I am deeply grateful to all of you for motivating me to complete this thesis.

Chapter 1

Introduction

In the realm of molecular biology, group I introns stand out as intriguing puzzles, challenging established ideas and inviting exploration. They were introduced in the 1980s, when in 1982 Thomas R. Cech and his team were able to demonstrate in Tetrahymena that an RNA molecule was able to cut and rejoin chemical bonds without any proteins [6]. Cech's team demonstrated the group I intron's ability to self-spice and act as ribozymes when a group I intron inserted within the nuclear large subunit rRNA gene in the protozoan Tetrahymena thermophila. During this time, there was also Francois Michel and his team who recognized that organellar group I introns can fold into conserved secondary structures at the RNA level [15]. Since the introduction of group I introns from these researchers, other researchers decided to examine the various aspects of group I introns, such as their structure, evolutionary significance, and their practical applications in fields like biotechnology.

However, the identifying and characterizing of group I introns over the years has presented a significant challenge in bioinformatics. There have been tools that helped

identify introns. Some include MFAnnot, RNAWeasel, Infernal, and HMMSearch, just to name a few. However, these tools can only take us so far with intron identification. Each tool utilizes different techniques and this results in varying rates of accuracy. We see variations in accuracy due some techniques being better at identifying introns with certain properties that may not necessarily be applicable to all introns. However, combining different techniques has the potential to achieve higher accuracy compared to each individual tool. The goal of this project is to create a pipeline, where we combine a series of individual tools, so that we can automatically identify introns with high accuracy.

One of the main issues of identifying group I introns and group II introns is that they can be looked at as silent parasites, meaning that they are present in the host genome without causing any disruptions. Another reason is that they are mobile elements, where they can duplicate themselves and reinsert in different locations on the host's genome. They are found in all domains of life, which are Bacteria, Archaea, and Eukarya, but they are hard to identify because their sequences are not well conserved, only certain domains of their structure are well conserved.

The identification, detection and classification of group I introns in fungal mtDNA has been tedious for the past few years, since manual curation of the results provided by existing tools is necessary. To fully identify these introns and their boundaries, comparing a newly sequenced and assembled mitochondrial genome to a reference genome is required. The large gaps and mismatches within the genes, indicating the presence of introns, would have to be manually realigned to identify the introns and where their exact insertion sites are located. Doing all of this requires a huge amount

of time and resources to complete a mapping for one genome.

1.1 Thesis Objective

In this M.Sc. Thesis I propose a pipeline named HIPipeline (HMMER & Infernal Pipeline) that can help to significantly lower the amount of work and manual steps needed to identify group I introns without sacrificing the accuracy of boundary detection. Identifying these introns would give us a better understanding of the organization of mtDNA in some fungi, which can be used in different applications in bioinformatics and drug development or fungicides. With this method we can identify group I Introns along with their precise location allowing us to annotate the genome.

1.2 Thesis Organization

In Chapter 2, I will discuss the Biological and Computational background for this thesis project. For the Biological background, I will look at introns, group I and group II, Open Reading Frames, Protein Families, and Fungi. For the Computational Background, I will focus on tools and databases there were involved in this project along with some extra tools and databases to provide some insight on where research has gone in this direction. Tools includes Infernal, HMMER, ORFipy, ORF Finder, RNAweasel, and many more. For databases, these include NCBI, InterPro, GISSD and so on. In this section, I will go into detail about what these sub-topics are and how they are related to group I introns.

Once I have established the Biological and Computational background, we can

then look at the methodology of my work in Chapter 3. There, I begin with the problem statement, and a discussion of the roadblocks that are encountered when studying group I Introns. These include the time it takes to identify one of them and how accurate these findings are. After establishing the problems faced by current research, I then proceed to present the methodology of my proposed pipeline. My proposed pipeline is using some of the tools that are discussed in the background section, specifically Infernal and HMMER. These are the 2 main tools I used to find our group I introns and gene best hit. I will also demonstrate how I am filtering out our data and updating some of the hits I have obtained.

After explaining our methodology, the results will be presented in Chapter 4. The results are based on a distinct dataset of 60 mitogenomes (mitochondrial genomes) not used for training. They will show how accurate and efficient my proposed pipeline is. In this section, I will also provide an overview of the output from my pipeline and a discussion of the significance of these results compared to other existing approaches.

Lastly in Chapter 5, I will present my conclusion by summarizing the results and their importance. I will also identify and discuss the limitations faced during the development of this pipeline and ideas for future work that could improve this pipeline further.

Chapter 2

Background

2.1 Biological Background

2.1.1 Introduction to Genomes and the Genetic Code

In order to present the focus of this thesis, introns, we have to introduce what they are and how they fit into the genome. We are specifically focusing on fungal mitochondrial introns in this thesis. I will begin by describing all the components surrounding the intron. The fungal genetic information is stored in a genome. In basic terms, the genome can be described as the entire genetic material present in an organism. The genome stores the genetic information in the form of DNA. The DNA, which stands for deoxyribonucleic acid, is a molecule that carries the genetic information using four different bases called adenine, cytosine, guanine, and thymine. These bases are connected to the DNA backbone. The DNA backbone is a sequence of alternating deoxyribose sugar, a sugar molecule containing 5 carbon atoms, and

phosphate groups. We call the base along with its portion of the backbone (a sugar molecule and a phosphate group) a nucleotide. Nucleotides are chained together, forming a strand. The DNA molecule is formed of two complementary strands going in opposite directions and attached by the bases. This means the nucleotides form pairs with each other in a predictable manner. Adenine always pairs with thymine and cytosine always pairs with guanine in DNA.

The DNA contains genes which code for proteins and non-coding RNA which are used for cellular function. At the basic level, a gene coding for proteins is a segment of DNA that starts with a start codon and ends with a stop codon. The codon is sequence of three nucleotides corresponding to a specific amino acid. The length of a gene can be described in terms of the number of nucleotides comprising it. The length may vary from gene to gene depending on the structure and function of the protein. Some genes could consist of several hundred nucleotides and some could be several thousand nucleotides long. It is also important to note the sequences between genes will vary. These variations give proteins variety in structure and functionality. Generally there are five main types of genomes including, nuclear genomes, mitogenomes (mitochondrial genomes), chloroplast genomes, prokaryotic genomes, and viral genomes. Here we will discuss only two of them, the nuclear genome and mitogenomes. Both nuclear and mitochondrial genomes have similarities, yet they both have characteristics that are unique to each other.

I will discuss the similarities and differences between nuclear and mitochondrial genomes. Both genomes store the genetic information in the form of DNA. The nuclear genome is located within the nucleus of a cell and the mitochondrial genome

is located in the mitochondria. The mitochondria are organelles required for ATP production within cells. ATP, also known as adenosine triphosphate, is the primary source of energy used within cells for various processes. Both the mitochondrial genomes and nuclear genomes are comprised of double-stranded DNA, however, the nuclear genome is generally linear and the mitochondrial genome is generally circular. Additionally, mitochondrial genomes are much more compact containing primarily genes required for mitochondrial function (e.g. genes coding for components of the electron transport chain, ATP synthesis, etc.), tRNAs and rRNAs. However, nuclear genomes contain a significant amount of non-coding genes producing non-coding RNAs that are essential for gene expression and regulatory purposes. To put the size into perspective, in humans the mitochondrial genome contains approximately 37 genes whereas the nuclear genome contains approximately 20,000 to 25,000 genes. Another key difference is the mitochondrial genome is inherited maternally whereas for the nuclear genome both parents contribute one set of chromosomes. The mutation rate differs between the two genomes as well. The mutation rate in the mitochondrial genome is generally higher due to exposure to unstable molecules containing oxygen and a simplistic DNA repair system [13]. However, in the nuclear genome the mutation rate is generally lower due to a robust DNA repair system. In summary, both mitochondrial genomes and nuclear genomes are vital for cellular function, however, the mitochondrial genome is utilized for energy production whereas the nuclear genome regulates most of the other cellular processes.

The central dogma describes the process of how DNA is used to synthesize protein molecules [2]. Generally, the central dogma states that genetic information flows from

DNA to RNA to protein. This is done using two processes called transcription and translation.

During the first step, transcription, the DNA is used by the RNA polymerase (RNApol) to synthesize a complementary RNA molecule called messenger RNA (mRNA). This newly synthesized transcript is sometimes referred to as pre-mRNA since it is unprocessed and will be missing the 5' cap, the poly-A tail and may contain both exons and introns. More details about the pre-mRNA will be discussed in section 2.1.2. The RNApol is the enzyme that synthesizes the mRNA by binding to the DNA. The region right before the gene is called the promoter. The promoter indicates to the RNApol where to begin transcribing the DNA into RNA. Once the RNApol binds to the promoter region, the DNA unwinds (due to the action of the RNA pol) and exposes the template strand used by the RNA pol to synthesize the transcript. The RNApol moves along the DNA template in a 5' to 3' direction. The 5' to 3' refers to the orientation and direction of the strand, and is based on a numbering of the carbon atoms on the deoxyribose of the DNA backbone. With regards to the transcript, the 5' end usually has phosphate groups attached and the 3' end is defined by a hydroxyl group (-OH). The mRNA consists of ribonucleotides. Ribonucleotides are similar to nucleotides, however, the (deoxy)ribose in DNA at the 2' position has a -H group (hence the name "deoxy") whereas in RNA the ribose at the 2' end has a -OH group attached. There are four different types of ribonucleotides, adenine, guanine, cytosine, and uracil (the uracil replaces thymine in RNA). The RNApol continues to traverse the DNA strand, continuously adding ribonucleotides to the mRNA strand until a stop signal is encountered. When this stop signal is encountered, the RNApol

releases the RNA molecule and detaches from the DNA. This transcript (pre-mRNA) may then mature into an mRNA. In eukaryotes, RNA processing involves the removal of introns (called *splicing*), the addition of a 5' cap, and the addition of a poly-A tail at the 3' end. After the RNA processing completes, we get a "mature" mRNA that can be translated in the cytoplasm of the cell.

In the second step, translation, the mRNA is used to synthesize the protein molecule by the ribosome. The ribosome is a complex molecule found in all living cells and is responsible for assembling the chain of amino acids, called the polypeptide, that form the protein molecule. The ribosome uses the reading frame to determine the sequence of amino acids to assemble. The reading frame refers to how a sequence of nucleotides is divided into groups of three known as codons. A codon is a sequence of three nucleotides that together specify the next amino acid in the chain, or a start signal, or a stop signal. The start codon is part of the signal in order to assemble the ribosome properly for translation. In eukaryotes we also need the 5' cap and the Kozak sequence, which may include the start codon. The Kozak sequence is a short sequence that functions as the protein translation initiation site. After the ribosome is assembled on the mRNA, it moves along the strand reading each codon and assembles the amino acid chain. Each transfer RNA (tRNA) carries a specific amino acid. The three nucleotide complementary sequence, called anticodon, on the tRNA determines the amino acid it is carrying. The ribosome uses the anticodon to determine whether a tRNA is carrying the correct amino acid corresponding to the codon of the mRNA. The tRNA with the correct anticodon enters the ribosome. The ribosome attaches the amino acid to the growing chain and the tRNA leaves the ribosome. The process

repeats until a stop codon is detected. After a stop codon is detected, the process enters the termination phase. The ribosome releases the assembled amino acid chain and the ribosome disassembles from the mRNA.

2.1.2 What are Introns and Exons?

Before 1977, it was believed the mRNA did not undergo any preprocessing before being translated by the ribosome into a protein [5]. Genes in eukaryotic organisms are much more complex. Berk and Sharp confirmed and provided evidence that the mRNA molecules were indeed shorter than their DNA counterparts [5]. It was determined that sections of the transcript are removed or "cut out" before being translated by the ribosome into a protein molecule [49]. Scientists classified the sections of the gene that are removed as introns and the regions that remain within the mRNA as exons. The initial mRNA strand synthesized through transcription is called the pre-mRNA. This strand is a complimentary copy of the DNA strand. However, the pre-RNA cannot be translated into a protein as is since it lacks the proper supporting structures such as the 5' cap and the Poly (A) tail at the 3' end. In addition, the pre-mRNA strand contains both introns (non-coding parts of a gene) and exons (the coding parts). After the introns are spliced out, the resulting pre-mRNA molecule is designated as the mature mRNA transcript. Splicing refers to the process of removing introns and joining the exons together to form a continuous coding sequence that can be used to synthesize a protein molecule.

2.1.3 Group I & Group II Introns

Group I and group II introns are special types of introns that are mobile elements: they have the ability to remove themselves from pre-mRNA molecules (sometimes with the help of maturases [37]). Group II introns are important since they are considered to be the ancestors of modern splicing systems [23]. Modern splicing systems are much more complex and involve many protein cofactors in the process. The splicing mechanism for group I and group II introns differ. Group I introns (see figure 2.1 for a representation of its secondary structure) use guanosine nucleotides that are freely floating within the cell to initiate the splicing process. The guanosine acts as a co-factor to initiate the self-splicing process which cuts the intron out from the pre-mRNA. However, group II introns (see figure 2.2 for a representation of its

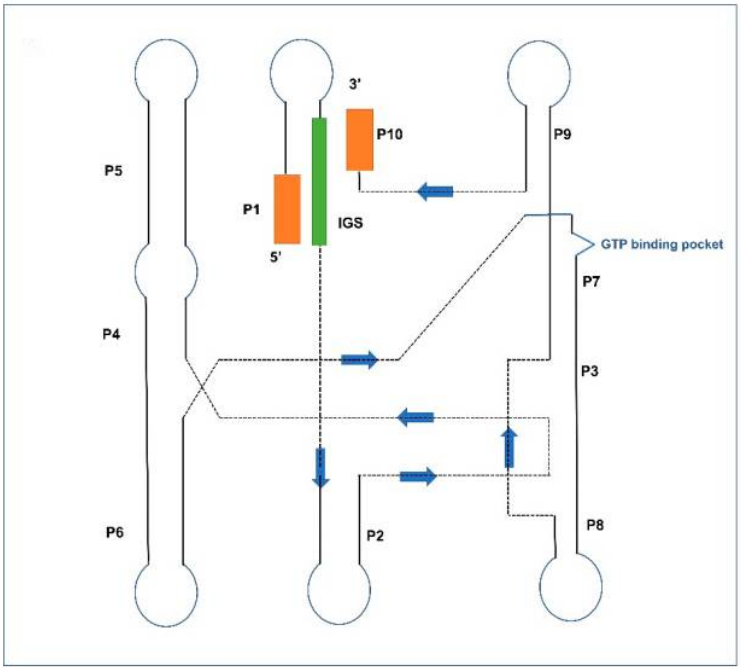


Figure 2.1: Secondary structure model for Group I intron (image taken from [36])

secondary structure) differ since these introns use an adensosine within the intron itself to initiate the splicing process. Group II introns do not require external cofactors to initiate the splicing. The intron forms a looped structure within itself resulting in the intron being excised as a lariat, which is a looped structure, from the pre-mRNA.

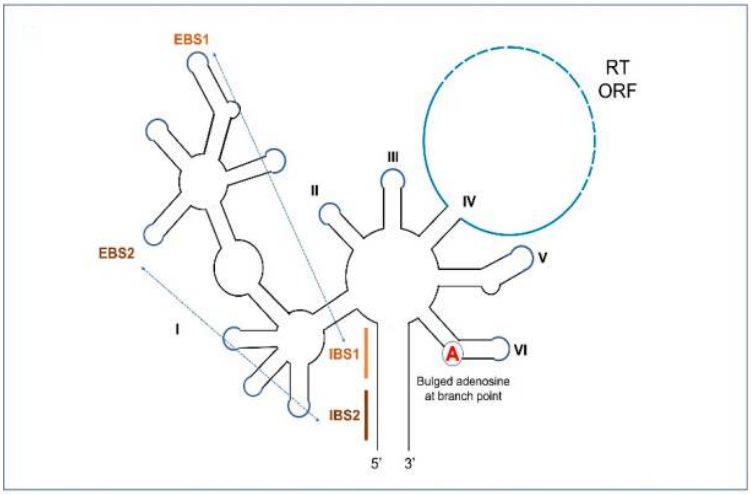


Figure 2.2: Secondary structure model for Group II intron (image taken from [36]). The Figure shows the position of the branch point (Adenine that initiates the first step in splicing) and the Exon binding sequences (EBS1 and EBS2) and the Intron binding sequences (IBS1 and IBS2, in the flanking exon) that are promote the RNA folding and interactions needed for splicing.

There are five types of group I introns, IA, IB, IC, ID, and IE, which are further divided into 14 subtypes (e.g. IA is subdivided into IA1, IA2 and IA3). In this thesis we will be focusing on four group I intron types: IA, IB, IC, and ID. We will not be focusing on IE because are found only in nuclear ribosomal DNA regions [20]. All the types and subtypes are categorized based on structural differences.

2.1.4 Twin-trons

The term twin-trons was first introduced in 1991 by Copertino and Hallick [10], and was defined as any set of nested introns, using the same splicing mechanism or not [12]. In a paper by Hafez and Hausner [19], the authors considered various types of twintron-like arrangements and sub-categorized them further into subtypes based on differences in their mode of splicing. In a study by Kavalecz *et al.* [26], the authors looked into spliceosomal twin introns (named stwintrons for short), which are introns where any of the three consensus sequences involved in splicing is interrupted by another intron (an internal intron). However, in this thesis we use a simpler definition of twin-trons, where we consider any two intron modules that occur in tandem or are separated by a "short" exon (micro or mini-exon, consisting of about 30 nucleotides or less). Figure 2.3 provides an visual representation of what we mean by twin-tron, where the two introns, represented in purple and blue, are divided by an small exon (mini-exon). As will be discussed later in the thesis, the mini-exons are difficult to identify since they are quite small compared to other exons. They could then be missed and that would result in the identification of unusually large introns that would require some post-processing to identify as twin-trons. I will go into more details about this in the Methodology section.

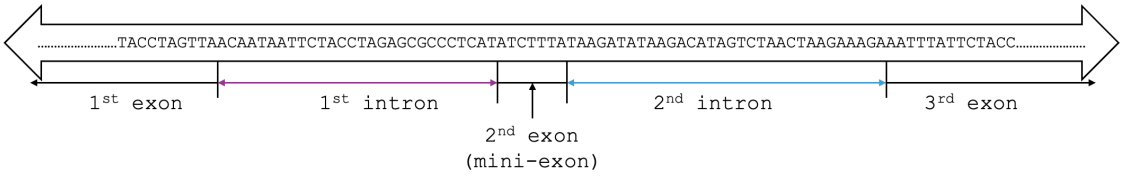


Figure 2.3: An example of a Twin-tron

2.1.5 Why are Introns worth Studying?

There are several reasons why introns are worth studying and I will explain some of them in the following paragraphs.

The first reason we are interested in introns is they provide a mechanism for precise control of gene expression within the cell [24]. Recall that introns are non-coding sequences, however, spliceosomal introns may contain regions denoted as silencers or enhancers. It is these regions that regulate gene expression by interacting with transcription elements. As indicated by the name, regions called enhancers increase gene expression levels. Enhancers contain sequences that serve as binding sites for transcription elements allowing for efficient binding of transcription binders resulting in increased initiation rates. Ultimately enhancers amplify the synthesis of specific mRNA transcripts resulting in increased gene expression levels. Silencers on the other hand inhibit gene expression by hindering the recognition of splicing sites or hindering the binding of transcription factors, which results in the gene not being expressed as much.

A second reason is that introns provide a mechanism for gene diversity [24]. Splicing is carried out by the spliceosome which consists of small nuclear ribonucleoproteins and auxiliary proteins. The spliceosome recognizes the boundaries between introns and exons. These sites are known as donor sites and acceptor sites. In addition, these also catalyze the excision of introns and the joining of exons for a continuous mRNA transcript. The diversity of a gene comes from alternative splicing. Alternative splicing is the possibility of having different combinations of exons being assembled together [24]. This leads to multiple mRNA isoforms from a single gene. These isoforms

lead to the synthesis of proteins and give rise to structural and functional alternative properties from a common precursor mRNA. It allows for proteomic diversity, tissue differentiation and development. There are different types of alternative splicing. When an intron is removed from an mRNA, this allows for different combinations of exons to be joined together. This mechanism, allowing exons to be joined together in different combinations, exponentially increases the number of diverse proteins that can be produced from a single gene.

A third reason is that group I and group II introns can contribute to the genetic mutations within the genome. These introns are mobile elements which means that they can insert themselves into new locations within the genome [48]. When an intron relocates to another region within the genome, it can possibly impact several cellular processes. An example of this is when an intron inserts itself near a promoter. An intron near a promoter may enhance or disrupt the expression of a gene by affecting how efficiently transcription factors are able to assemble to transcribe the gene. When a gene's expression level is altered, this can impact the amount of its product within the cell by either enhancing or inhibiting it which may effect the cells primary purpose. Overall, when mobile elements such as group I and group II introns move within the genome, it can impact the expression of a gene which can contribute to the evolutionary process of a genome.

2.1.6 Open Reading Frames

In the traditional definition, Open Reading Frames (ORFs) are segments of the genome with the potential to be transcribed and translated into a protein by the

cellular machinery. In the context of group I introns, there are segments inside them that code for specific proteins helping them to move around. These segments have been named simply "ORFs", but they do code for a specific protein. Identifying these ORFs require a bit more work than simply identifying start and stop codons in the same frame, since we also need to verify that they also code for a specific protein. This will be explained more in detail in the methodology section. Moreover, some ORFs may resemble gene sequences but do not encode functional proteins; these are known as pseudogenes. Pseudogenes are DNA sequences that closely resemble functional genes but have undergone mutations or other alterations that render them non-functional. ORFs can vary significantly in size, ranging from a few hundred to several thousand nucleotides. [50]

2.1.7 Fungi

Fungi are of great ecological and economic importance and, as most fungi are obligate aerobes (they require oxygen to grow), mitochondria and their genomes (also called mtDNA) are of the utmost importance [27]. Types of fungi include yeast, rust, stinkhorns, puffballs, truffles, molds, mildews, and mushrooms. A fungal mitochondrial genome can range in size from about 12 kbps to $\geq$ 500 kbps (kilobase pairs) [7], due to the presence of intergenic spacers, duplications/repetitions, and for most mitogenomes, due to the absence or presence of mitochondrial introns [52]. Furthermore, group I introns are more frequently encountered compared to group II introns in fungi [36]. In fungi, some mitochondrial introns in the cob gene have been associated with fungicide resistance. A recent example is from a study by Cinget and Bélanger

[8], where they researched group I-D introns and associated these introns with the cytochrome b (cob) genes in the mitogenomes of 169 fungal species. There has also been examples of drugs developed to target group I and group II introns in fungi, although they are not yet in use for treatment of fungal pathogens. In a recent study, Liu and Pyle [31] discovered a set of ubiquitous introns within thermally dimorphic fungi, which are genera of *Blastomyces*, *Coccidioides*, and *Histoplasma*, by identifying unique RNA structural signatures. The focus of this thesis will be to identify group I introns in fungi.

2.2 Computational Background

The field of bioinformatics primarily focuses on the analysis and interpretation of biological datasets using technology, mathematics and statistical analysis. We generally use computer software to manage biological data primarily because these data sets are very large and complex. The data can include genomic sequences, protein structures and gene expression patterns. There are several key areas that bioinformatics focuses on. The first area is genomics and transcriptomics which revolves around the analysis of DNA and RNA sequences to study gene function, expression patterns, and evolution [11]. The second area is proteomics where the protein structure, function and interactions are studied [55]. The third area of study is structural bioinformatics where scientists use prediction tools to infer the two-dimensional or three-dimensional structures of proteins and nucleic acids [55]. The fourth is systems biology where we analyze biological systems such as metabolic pathways or gene regulatory networks [11]. The fifth is comparative genomics where we create algorithms

to compare genomes in order to explain the evolution of species, protein folding and population genetics [4]. Finally the sixth area is data management where we store and organize large datasets for reliability and easy access. The following subsections will focus on the computational aspect of the thesis work.

2.2.1 Identification of Group I and II introns

Identifying these introns has been tedious. Both introns in group I and group II are difficult to identify due to very little conservation at the sequence level, meaning that they are difficult to detect with standard sequence similarity searches [29][42]. Nevertheless, this has not discouraged researchers from developing software or programs in helping partially identify if the given genome has introns and what subgroup they are under. Next, I will discuss some approaches and techniques related to the identification of group I and group II introns.

2.2.2 Tools

BLAST

There have been multiple programs that have helped identify certain parts of a given intron. The earliest software that has been used is called BLAST, which stands for Basic Local Alignment Search Tool. This tool is used to find regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of the hits (called high-scoring pairs). It takes a query sequence as the input, compares it to other sequences in the GenBank database and returns a list of possible sequences that

are similar to the query sequence [35]. This tool is useful for quickly searching for a query sequence in a large database and can be useful when searching for a sequence or pattern that is well-defined and conserved. There are multiple search tools under this, such as BLASTX, tBLASTn, lgBLAST, etc. BLAST can be used to look for group I introns from a given host gene, but it is not recommended. One of the issues in using BLAST is due to the relatively high levels of sequence variation in introns, especially in group I introns, which can degrade the quality of high-scoring sequence pairs, and thus lead to imprecise and fragmented hits. The other issue is that similarity comparisons are blind to secondary structure, which limits their capacity to bridge distant conserved motifs [45].

CITRON

CITRON is one of the pioneering programs developed for predicting group I introns. Introduced in 1994, it became widely used for this purpose. CITRON operates by using a nucleotide consensus matrix of the intron core and performs a hierarchical search of peripheral secondary structure elements. The program's predictions are based on data from 143 introns known at the time of its development. However, its primary limitation is that it only considers the common core features of group I introns, focusing primarily on helical or paired regions designated as P1 to P10 [22]. Unfortunately, CITRON does not appear to have received recent updates [30][29].

Covariance Models

A covariance model is a representation of patterns in both DNA and RNA sequences by utilizing both structural and sequence information. The model is able

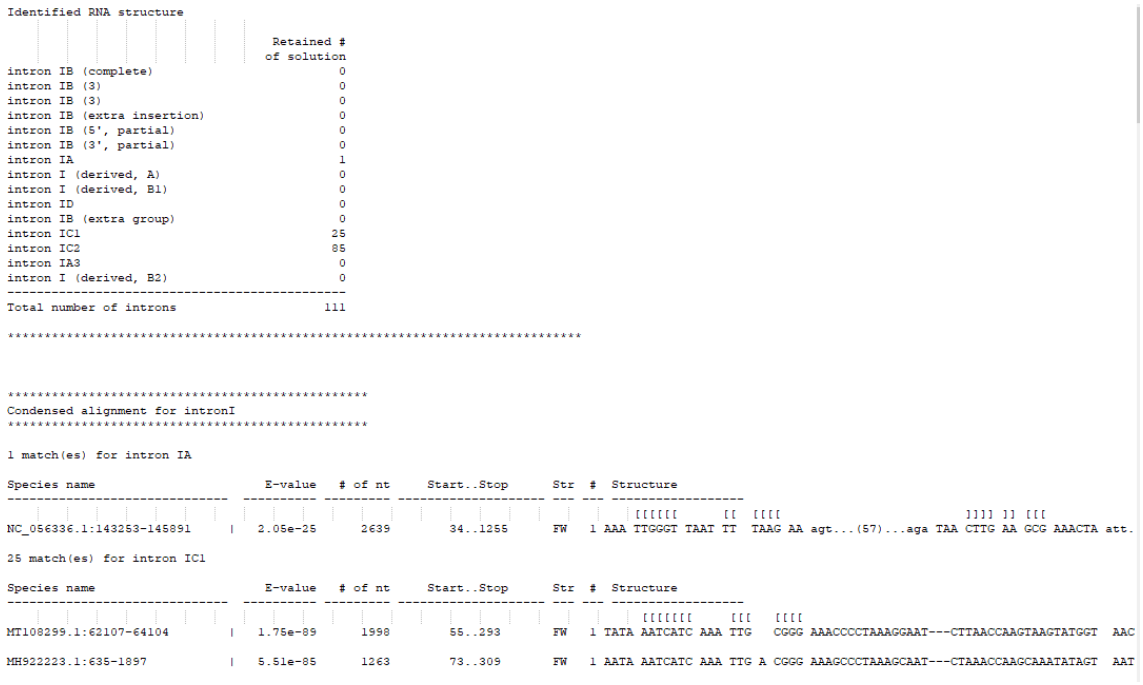
to identify fragments of sequences that are conserved across different genomes from different organisms. It also recognizes general nucleotide bonding patterns during RNA folding. These pairings are important since they ensure the structural integrity of the molecule. In addition, the covariance model is able to make predictions of how changes in one part of the sequence will affect another part of the sequence in order for structural integrity to be maintained for the molecule. Overall, the covariance model can predict the RNA structures by analyzing the sequence and its folding patterns.

RNAWeasel

Traditionally in the past, bioinformatics tools were developed to study sequences that code for protein molecules. However, scientists discovered that some cellular functions did not need a DNA sequence to be translated into a protein molecule in order for it to perform a function. In response to this, scientists needed tools that could identify RNA sequences that are not translated into protein molecules but are still functionally significant for cellular processes. RNAweasel was developed to identify and annotate these non-coding RNAs or ncRNAs [29]. An ncRNA is an RNA sequence that is not translated into a protein, however, they are still functionally important as they are required to perform other functions within the cell. Some examples of ncRNA include, transfer RNA (tRNA) which carry amino acids to the ribosome during protein synthesis, ribosomal RNAs (rRNA) which are the primary component of ribosomes, and microRNA which are used to regulate gene expression. RNAweasel utilizes both the sequence and the secondary structure during its analysis to identify ncRNAs [29].

RNAWeasel is able to identify conserved motifs within the genome using an alignment based approach [29]. A conserved motif is a short sequence of nucleotides within the genome that is widespread among many organisms and performs the same function in each genome. However, there are cases where ncRNAs with similar functions may not necessarily be conserved in terms of primary sequence. RNAWeasel uses covariance models to predict the secondary structure of a given sequence and classify whether these sequences are similar in terms of structure [29]. Sequences that have a similar secondary structure are assumed to be similar in terms of functionality. This allows RNAWeasel to utilize existing sequence and secondary structure information to identify and classify segments that have been studied in other genomes.

RNAWeasel is able to classify and identify group I or group II introns based on secondary structures. For group I introns, RNAWeasel specifically looks for P1 to P10 helical domains since those form into a structure that enables the intron to self-splice. RNAweasel also relies on finding conserved sequences such as the internal guide sequence (IGS) to identify group I introns. In order to identify group II introns, RNAWeasel relies on the secondary structure as well. Group II introns consist of six distinct domains DI to DVI. These domains are required for the self splicing mechanism so they are good characteristics to use for finding group II introns. RNAWeasel also searches for conserved sequences within these domains to identify group II introns. In summary, RNAWeasel is able to identify group I and group II introns based on both sequence and structure information. Figure 2.4 shows an example of an RNAWeasel output.



```
Identified RNA structure

                                  Retained #
                                 of solution
intron IB (complete)                       0
intron IB (3)                              0
intron IB (3)                              0
intron IB (extra insertion)                0
intron IB (5', partial)                    0
intron IB (3', partial)                    0
intron IA                                  1
intron I (derived, A)                      0
intron I (derived, B1)                     0
intron ID                                  0
intron IB (extra group)                    0
intron IC1                                25
intron IC2                                85
intron IA3                                 0
intron I (derived, B2)                     0
-----------------------------------------------
Total number of introns                  111

*******************************************************************************


**********************************************
Condensed alignment for intronI
**********************************************

1 match(es) for intron IA

Species name                      E-value   # of nt     Start..Stop      Str  #  Structure
-------------------------------  ----------  ---------  ---------------- --- --- ------------------
                                                                                 [[[[[[      [[  [[[[                          ]]]] ]] [[[
NC_056336.1:143253-145891     |   2.05e-25      2639        34..1255      FW   1 AAA TTGGGT TAAT TT  TAAG AA agt...(57)...aga TAA CTTG AA GCG AAACTA att.

25 match(es) for intron IC1

Species name                      E-value   # of nt     Start..Stop      Str  #  Structure
-------------------------------  ----------  ---------  ---------------- --- --- ------------------
                                                                                    [[[[[[[     [[[   [[[[
MT108299.1:62107-64104        |   1.75e-89      1998        55..293       FW   1 TATA AATCATC AAA TTG   CGGG AAACCCCTAAAGGAAT---CTTAACCAAGTAAGTATGGT  AAC

MH922223.1:635-1897           |   5.51e-85      1263        73..309       FW   1 AATA AATCATC AAA TTG A CGGG AAAGCCCTAAAGCAAT---CTAAACCAAGCAAATATAGT  AAT
```

Figure 2.4: RNAWeasel Sample Output

Infernal

Another tool that was developed for RNA sequence analysis is called Infernal. Infernal is also known as INFERence of RNA Alignment. The Infernal tool was designed to utilize covariance models during the RNA analysis. The covariance model allows Infernal to consider both the primary sequence and secondary structure of the sequence being analyzed [3]. This is an advantageous approach since some sequences are susceptible to mutations which results in alterations in the primary sequence. The primary sequence of ncRNAs from the same family (*i.e.* sharing a similar function) may not necessarily be conserved, however, their secondary structure generally is conserved in order maintain functionality.

An advantage of Infernal is that users can train the covariance models from pre-

existing data [40]. Essentially this allows us to integrate Infernal with databases that have collections of RNA families that have been previously identified and studied in other genomes. This allows Infernal to annotate genomic sequences that have not been previously studied [40]. If there are ncRNA sequences in the genome that are homologous to ncRNA sequences previously studied in other genomes, Infernal will be able to identify and annotate these regions based on either the primary sequence or the secondary structure. Overall, Infernal's incorporation of sequence alignment and secondary structure information improve its accuracy for both small scale and large scale analysis of RNA sequences. Figure 2.5 shows an example output of an Infernal search.

Infernal and RNAWeasel both appear to be similar, however, there are differences between the two tools. RNAWeasel was designed to identify and annotate medium-sized ncRNA sequences, such as tRNAs, group I and group II introns, using a combination of sequence and secondary structure information. However, Infernal is designed to perform analyses on a broad range of ncRNAs that can be more complex in terms of structure and of varying sizes compared to RNAWeasel such as ribosomal RNAs and microRNAs for example. Infernal does not come with built-in models, although researchers typically share the models that they have built. Indeed, the user is responsible for gathering sequences from previous experiments or databases and building and calibrating their own model that will be used for searching. In other words, Infernal offers a lot more possibilities and flexibility but it requires more work from the user in terms of collecting and curating a dataset to build a good covariance model. In summary, Infernal was designed to identify a broad range of ncRNAs

```
# cmscan :: search sequence(s) against a CM database
# INFERNAL 1.1.4 (Dec 2020)
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query sequence file:                   Group_IA.fasta
# target CM database:                    IAComplete.cm
# number of worker threads:              4
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       MT108299.1:73341-76166  [L=2826]
Description: Cladobotryum mycophilum mitochondrion, complete genome
Hit scores:
 rank     E-value  score  bias  modelname  start    end   mdl trunc   gc  description
 ----   --------- ------ -----  --------- ------ ------   --- ----- ----  -----------
  (1) !   3.9e-31  145.3  36.0  IA1         2516   2821 +  cm    no 0.27  -
  (2) !   2.3e-07   28.4  35.9  IA2         2516   2806 +  cm    no 0.26  -
  (3) !   6.5e-06   21.8   6.1  IA3         2680   2788 +  cm    no 0.26  -
 ------ inclusion threshold ------
  (4) ?       1.6    3.3   0.1  IA3         2537   2571 +  cm    no 0.17  -
  (5) ?       2.1    6.9   3.3  IA1         1878   1812 -  cm    no 0.21  -
  (6) ?       7.4    4.5   0.1  IA1         1072   1119 +  cm    no 0.25  -


Hit alignments:
>> IA1
 rank     E-value  score  bias mdl mdl from   mdl to       seq from      seq to       acc trunc   gc
 ----   --------- ------ ----- --- -------- --------    ----------- -----------      ---- ----- ----
  (1) !   3.9e-31  145.3  36.0  cm        1      385 []        2516        2821 + .. 0.89    no 0.27

                                         v                                   v                                                                     vv                              NC
                                    :::::<<<<<<<_____>>>>>>>,.<<<<<<<____>>>>>>>........,,,,,,,((((((,,,<<<<<<----<<<<<< CS
                          IA1     1 cgcUAgGGAUaaAauuuuuGUCCca.uuuuauuGuAAauaaaa........uAAuAAAUuGGGugAAUUGCAAgAAauCuuau 76
                                      C  :  :U::AA  U::G:  :  U:::::: +AA::::::A        +   AAAU:GGG  AAUU:CAA: A AU:U:AU
  MT108299.1:73341-76166 2516 AUCCUAACCUGUAA--UAUGGGCUCcUUAAAAAUAAAAUAUUUAuuuuuuagAGGCAAAUGGGGAAAAUUACAAAGACAUUUAAU 2598
                                    ************85..7*******676666666555545555555566667777799*****************************  PP

                                    v                                v                                                                                                    NC
                                    <<_......______...._>>>>>>>>--->>>>>><<<---<<<<<<<______________>>>>>>>>-->>>,,,,,,, CS
                          IA1    77 uau......aauaaa....uuaauaaGacAAcUUGCAGCGAAGcUuauuucaacuauauauaauugaaauaAgAACGUUCAACGA 151
                                     AU                    +U AU:A:A+ A:UUG:AGCG A CUUA:U:C  +U UAU+UAA +G:A:UAAG ACGUUCAACGA
  MT108299.1:73341-76166 2599 AAUauaccu------ccuaAUAAUUAAAUCAUUUGUAGCGGAACUUAAUUCGGUU-UAUUUAAGAGGAUUAAGGACGUUCAACGA 2676
                                    987655553......4555389****************************9884.444444467*******************  PP

                                       v                                            v       vv                                                                           NC
                                    ,,,<<<<<_____________________>>>>>,,,,,,)))))))<<<<<<<<<______________________________ CS
                          IA1   152 CUAgaaggUGAGUAauauuAACAAUAAUccuucCACGAGcgCCCaAcAuCuuauaaauaaaaauaaAaaaaaauuauaaaaauaa 236
                                    CUA :::GUGAGU+A+A+ AACAAUAA C::: CACGA   CCC:A A::UU:: ++U+
  MT108299.1:73341-76166 2677 CUAAUGAGUGAGUUAAACCAACAAUAAGCUCAACACGAUAACCCCAGAAUUUUA-UUUU-------------------------- 2734
                                    ****************************************************8.6665..........................  PP

                                                                                                                                                                             NC
                                    ____________________________________________________________>>>>>>>>>,,,,,,,,,,,,,((<< CS
                          IA1   237 aauuauauuaaaaaaaaaaaaaaauaaaaaaauaaaaauaauaaaaaaaaaaaauuaaaaauauaaGaUgAUGACAUAGUCUGAaCu 321
                                                                                                  ::AA::U A+GACAUAGUCU AA::
  MT108299.1:73341-76166 2735 -------------------------------------------------------------UAAAAUUCAAGACAUAGUCUAAAUA 2759
                                    .............................................................9*************************  PP

                                                         v                   v                                   NC
                                    <<<<<____>>>>>>>,,<<<<<<<________>>>>>>>,,,,,,,,,,,)):::::::::: CS
                          IA1   322 auuugGAAAcaaauaGAAauaaauauUaAAaAauauuuaucgaUAACAaAAuUGAACAGGcUaA 385
```

Figure 2.5: Infernal Sample Output

whereas RNAWeasel was designed to identify specific ncRNAs such as group I and group II introns.

HMMER

RNAWeasel and Infernal differ from HMMER because these two tools were specifically designed for analyzing and detecting ncRNAs that are conserved in both sequence and structure. HMMER, however, was designed to align and match DNA sequences that code for proteins. In order to determine whether the sequence in question belongs to the sequence family being studied, HMMER must determine whether the alignment is a good match. In order to make this decision, HMMER uses the Hidden Markov Models [16]. The Hidden Markov Models are models built using nucleotide frequencies of specific genes or sequence families of interest. The model is used to help us determine what patterns we can expect for the sequence family being studied [16].

There is always some variation between sequences within a family, however, there must be some conserved regions that determine the shared functions. It is these common conserved regions that determine whether the alignment is a good match or not. If the sequence we are analyzing does not contain any of the conserved regions that are common to the other sequences within the family, then it is not a good match. However, if the sequence in question does have matching conserved regions that are common to the sequence family, then it is a fair to assume we found a good match. Traditional alignment techniques generally use the whole sequence when scoring the alignment to determine whether it is a good match. Unfortunately this could mean that an alignment may get overlooked due to non-conserved regions negatively impacting an alignment score. HMMER is different since it focuses on the conserved regions in order to avoid penalizing mismatches that are irrelevant. Sequences within

```
# nhmmscan :: search DNA sequence(s) against a DNA profile database
# HMMER 3.4 (Aug 2023); http://hmmer.org/
# Copyright (C) 2023 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query sequence file:             WINM59.fasta
# target HMM database:             geneDatabasewoCountModel2
# per-seq hits tabular output:     GIH_Results.tblout
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       Graphilbum  [L=28810]
Description: parva WIN M 59 mtDNA coxl start
Scores for complete hit:
    E-value  score  bias  Model     start    end  Description
    ------- ------ -----  --------  -----  -----  -----------
          0 1733.5 267.5  rnl       14880  17197
          0 1734.7 320.5  nad5      25073  26970
          0 1327.1 354.4  nad2      20487  22148
          0 1217.7 289.3  nad4       6350   7807
          0 1162.5 107.2  coxl          3   1128
          0 1075.1 157.1  cob       27134  28290
   2.4e-296  982.0 157.3  cytb      27135  28287
   1.9e-286  947.8  72.5  rns        9540  10750
   2.1e-207  687.6 121.1  atp6       8492   9261
   9.3e-196  648.6  67.5  nad1       4124   4751
   3.3e-187  620.2 105.0  cox3      11327  12149
   3.3e-164  544.7  69.2  ND1        4118   4775
   7.6e-150  496.8  77.4  cox2      22992  23540
   7.5e-140  462.9 179.4  nad6      13871  14598
   2.2e-110  365.5  45.1  coxl       3503   3964
    1.7e-91  302.9  79.8  nad3      22149  22553
    7.1e-86  286.4 172.1  ND6       13877  14533
    3.1e-79  262.6  83.0  nad1       5807   6239
    1.4e-77  257.5  57.4  nad4L     24804  25071
    1.9e-75  244.7   7.4  rnl       18684  18902
    2.3e-65  216.5  21.8  atp9      22669  22892
    2.1e-54  181.2  21.7  cox2      24411  24607
    2.9e-54  182.2  63.3  ND3       22149  22501
    3.1e-53  171.0  64.0  rnl       18874  19992
    7.2e-45  146.0  27.3  rns       10744  11023
    7.6e-41  136.5  78.5  ND1        5808   6219
    8.9e-27   92.5  40.8  atp8       8241   8384
     0.0022   11.2  21.7  nad4      25756  25998
     0.0083    7.4  23.1  nad2      25884  26032
  ------ inclusion threshold ------
      0.037    6.0  19.7  nad5      21412  21512
       0.47    1.6   5.2  nad2       7236   7274
       0.52    2.9   9.0  nad3       5462   5515
       0.65    3.0  14.7  nad4      21446  21496
       0.97    2.0  15.0  nad3       6358   6406
          3    2.4  31.7  atp6       7116   7284
        3.8   -1.4  14.9  nad2      27654  27733
        4.6   -1.7  12.5  nad2      25080  25155
        4.8   -0.2   6.0  nad3       4521   4470
        5.3    0.1   4.4  nad6       9143   9167
        5.7   -0.5   6.7  nad3       6836   6801
          8   -1.2  14.5  cox3      24311  24372
        9.4   -0.7   2.1  nad6      11957  12031


Annotation for each hit  (and alignments):
>> rnl
    score  bias    Evalue   hmmfrom    hmm to     alifrom     ali to      envfrom     env to      mod len      acc
    ------ ----- ---------   -------   -------    --------- ---------    --------- ---------    ---------    ----
  ! 1733.5 267.5         0        11      2967 ..     14880     17197 ..     14865     17199 ..      5679    0.92

  Alignment:
  score: 1733.5 bits
          rnl    11 aataaataaaattattagaaaaataaatgatataaaataaaaaattatacaaaaatcaatatAagTatataaAgaaAttattatggaatttaagc 105
                    a  ++a+a a+tt+t agaaa+ataaatgata++aaataaaaa+ttataca+a a  +at+tAagTatataaAga+A+tatta+g aat ++agc
  Graphilbum 14880 ATATTAAATATTTTTAAGAAATATAAATGATACGAAATAAAAATTTATACATATAATTATGTAAGTATATAAAGAGACTATTAGGTAATACTAGC 14974
                    3444444444999999**********************************************************************************9 PP

          rnl   106 tatatattaaagagatttaatt..aaatgaaaaaatttatttaattaatttggataATaaActctaaaaaaaatg..attataataaaagaagtg 196
```

Figure 2.6: HMMER Sample Output

a family will have differences and the alignment should not be penalized for these differences. The conserved regions should impact the alignment score more than irrelevant mismatches since the conserved regions are the unique characteristics of a sequence family. In summary, HMMER is able to identify homologous sequences that may not have been identified with other techniques due to its ability to recognize conserved segments for a given sequence family. Figure 2.6 presents an example output from HMMER.

MFAnnot

MFAnnot is different compared to the other tools since it was designed specifically to annotate mitochondrial genomes using both sequence and structural information [28]. In addition to group I and group II introns, it is designed to identify genes that are commonly found in mitochondrial genomes, such as genes related to ATP synthesis and tRNA genes. MFAnnot acts as a wrapper and utilizes external tools such as HMMER, Exonerate, and Erpin to annotate mitochondrial genomes [28]. HMMER utilizes the Hidden Markov Models as discussed in the previous section. Exonerate is a tool that utilizes dynamic programming to perform sequence alignments. Erpin uses dynamic programming to identify RNA motifs. An RNA motif allows us to group RNA fragments together based on the secondary structure rather than just the sequence [28]. MFAnnot generally does not work well on other genome types since the tool looks for patterns and characteristics that are specific to the mitochondrial genome. In order to identify group I and group II introns, MFAnnot generally utilizes both sequence comparison and secondary structure prediction. In summary, MFAnnot

is specifically designed to annotate mitochondrial genomes, including both the genes that are typically found there and group I/II introns. Figure 2.7 shows an example output from MFAnnot.

```
;; Masterfile modified automatically by mfannot
;; version Unknown
;; on Sun Dec 17 00:26:09 2023 by user shiny on host ce5d7c360d50
;;    - Gene Totals: 1
;;    - List of genes added:
;;      orf220
;;
;; end mfannot
;;


>MT479166.1:63686-65156 Trametes coccinea mitochondrion, complete genome gc=4
     1  TTTTGCATAATGGGTGAATTGCTTGCCCAAAAGAATTGAGAAATTCTTTAAATAACTGAC
    61  TCATGACCTTAAAAGGTCGGTATTATTTATTTGCTTAACAACAAAGATACA
;; mfannot:     /group=IA3 (7.32e-45)
   112  aataatgctaacggtggacccctaaggatattaatatctatggaaataccgtgggaagct
   172  gtaaatcgtaggaataagtaaaatatatttacttagtaagtttattctcttgtttaaatt
   232  tactctttttattagccctgtaacgacttggtggcaaaccaccagactgattgctcgcta
   292  ttgcgaataaaatcggtaacacgtcagcactcaaatattccaaaaaattatggtaatatt
   352  acttaaataaaaaaaataattaaaaaagttgctttttgcaaactagatagtgtatttttt
   412  atgtttcaaaattagtgttaaaaaaaaagaaaaaaattcaaatctattaaggttctttat
   472  tttaactcgagtagttatttttatttcttaatagtgttctatatttataggttaaaattt
   532  ttaagggataaaggtctatatatggatctctgctatgacctctttaaaatagttctagtt
   592  taaccatttaaattaaatatagaaagatgcattactagaatgtaaaggcagagattgtta
   652  gacgctaataaaaaatttattaaaaaaattaa
;     G-orf220 ==> start /note=LAGLIDADG ;; evalue:1.7e-21, first ATG found at position 687
   684  ataatgaaaaataatataaatttaatgtttgttagattttatagaaaatcttctaaatat
   744  gatttagaacttaatgctgatttgaaagaaatcatcataggtctaatgttaggagactta
   804  tttgctgaaaaaagaaatcctaattctaatacaagattacaatttaaacaatcaattaaa
   864  aataaagtatatattgaacatttatattctatatttaaagactattgtaattctgaacca
   924  aaaataactacatctatagataaaagacctggtaaaaaagatttaaatatttcaattaaa
   984  ttttggacacaaagtttaccttgttttaatcaatttagggaattattttatgatgaatta
  1044  ggtataaaatatatacctagtaatttagatgaaataataacacctagaagtttagcttat
  1104  tgggcaatggatgatggttataaatcaggtaaagggttttatttttgtacagaatcttat
  1164  actttagaagataatattaaattaagtcaaatactaaaaaatagatttaatttagaatgc
  1224  ggaatccataaacatacaaatggtcatagattgtacgtatttagtagctctaaagatatt
  1284  ttactagaattaattaaaccttatttaatcgaacatttttattataaatttgatttaaac
  1344  taa
;     G-orf220 ==> end
  1347  aaaaataaattatttttaaattaattttaacattagtaaagtaatttacctaacatttta
  1407  ttgagttgaaggtatagtctaatccaatatgaa
;; mfannot:
  1440  AATATTGGTATTAAATGCAGCAAGTTAATAGG
```

Figure 2.7: MFAnnot Sample Output

ORF Finder

ORF Finder is a tool designed to identify all possible open reading frames (ORFs) of a specified minimum size within a given nucleotide sequence or from sequences already in a database. It compares sequences using the BLAST server [46]. ORF Finder performs a six-frame translation of a nucleotide sequence given a particular genetic code, finding all possible ORFs. It then filters for ORFs above a certain threshold and exclude ORFs nested inside larger ORFs. Once it finds these possible ORFs, it returns a graphic with a table that indicates the location of each ORF found [53]. The graphic provides a visual bar chart that indicates where on the genome sequence the ORFs are found, and it differentiates if it is going from 5' to 3' or 3' to 5' in different colours. Below the graphic, it provides a table that lists detailed ORF data, such as the start and stop positions, the reading frame, and the length (see Figure 2.8). Users can input sequences via accession number, GI number, or nucleotide sequence in FASTA format. There are two versions of the tool: a web version that can take a query sequence up to 50 kb long and a stand-alone local

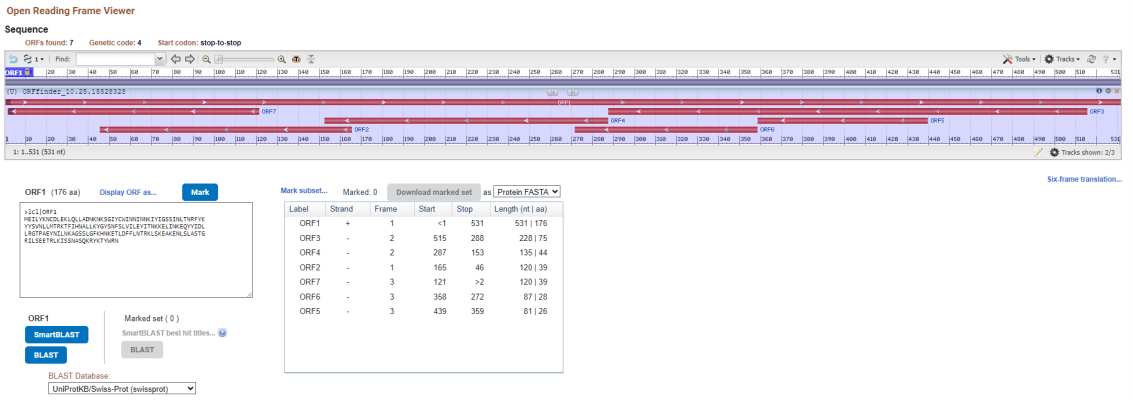


Figure 2.8: ORF Finder Sample Output

version, which has no length limitation and can be used on the terminal. However, ORF Finder is not meant to be used in eukaryotic genomes due to their complexity.

GetORF

GetORF is another tool that is used to find possible ORFs. Based on the documentation, it does seem to have a local version that you can download. However, this was unavailable for download at the time of writing. An online web version is also available. It can take as input a fasta file or a standard EMBOSS sequence query (also known as a 'USA' file). It can output the results in different formatting styles, such as GenBank format or fasta file. However, the output for GetORF does not provide a lot of information (see Figure 2.9). It does state if the ORF is in reverse complement (in the output it states as REVERSE SENSE) and it can translate the given input as protein, but it does not provide which reading frame it would fall under. This tool can be useful, however, over the course of this project, I ended up using the next tool that is being mentioned below.

ORFipy

ORFipy (part of the Galaxy platform) is another tool that is used to help identify ORFs. This tool was written in Python, and uses cython to implement the core ORF search algorithm to achieve fast execution times [51]. It uses the pyfastx library to help with the parsing of input fasta/fastq files. A fasta file contains only the sequence data whereas the fastq file contains both the sequence data and a quality score indicating the nucleotide accuracy. It contains a large variety of features to use in comparison to other programs such as GetORF. These features include outputting

EMBOSS explorer

OUTPUT FILE outseq

```
>_1 [2 - 46]
tggaaattttatataagaattgtgatttggaaaaattacaattat
>_2 [18 - 53]
gaattgtgatttggaaaaattacaattattagctga
>_3 [50 - 85]
ctgataataaaaataaatcaggtatttattgttgaa
>_4 [116 - 169]
gtagttctattaatcttacaaatagattttataaatattatagtgttaatttat
>_5 [173 - 205]
tgacacgtaaaacatttatacataatgctttat
>_6 [225 - 254]
tttttcattagttatattagaatatataac
>_7 [254 - 292]
ctaataagaaggaacttattaataaagaacaatattata
>_8 [279 - 323]
agaacaatattatatagatttattaagaggcacgcctgctgaata
>_9 [372 - 401]
agaaactttagatttttttttaaatacacg
>_10 [440 - 475]
ctagtacaggaagaattttatcagaagaaactagat
>_11 [479 - 529]
aaatatctagtaatgcttcgcagaaaaggtataaaacttactgaagaaact
>_12 [1 - 528]
atggaaattttatataagaattgtgatttggaaaaattacaattattagctgataataaa
aataaatcaggtatttattgttgaataaataatataaataataaaatatatataggtagt
tctattaatcttacaaatagattttataaatattatagtgttaatttattaatgacacgt
aaaacatttatacataatgctttattaaaatatggatatagtaatttttcattagttata
ttagaatatataactaataagaaggaacttattaataaagaacaatattatatagattta
ttaagaggcacgcctgctgaatataatatattaaataaagcaggttcttctttaggattt
aaacataataaagaaactttagatttttttttaaatacacgtaaattaagtaaagaagca
aaagaaaatttatctttagctagtacaggaagaattttatcagaagaaactagattaaaa
atatctagtaatgcttcgcagaaaaggtataaaacttactgaagaaac
>_13 [529 - 488] (REVERSE SENSE)
agtttcttcagtaagttttatacctttttctgcgaagcattac
>_14 [531 - 478] (REVERSE SENSE)
ttagtttcttcagtaagttttatacctttttctgcgaagcattactagatatttt
>_15 [439 - 362] (REVERSE SENSE)
ctaaagataaattttctttttgcttctttacttaatttacgtgtatttaaaaaaaaatcta
aagtttctttattatgtt
>_16 [515 - 291] (REVERSE SENSE)
gttttatacctttttctgcgaagcattactagatatttttaatctagtttcttctgataaa
attcttcctgtactagctaaagataaatttttctttttgcttctttacttaatttacgtgta
tttaaaaaaaaaatctaaagtttctttattatgtttaaatcctaaagaagaacctgcttta
tttaatatattatattcagcaggcgtgcctcttaataaatctata
>_17 [358 - 275] (REVERSE SENSE)
atcctaaagaagaacctgctttatttaatatattatattcagcaggcgtgcctcttaata
aatctatataatattgttctttat
>_18 [297 - 244] (REVERSE SENSE)
atctatataatattgttctttattaataagttccttcttattagttatatattc
>_19 [235 - 191] (REVERSE SENSE)
ctaatgaaaaattactatatccatattttaataaagcattatgta
>_20 [201 - 172] (REVERSE SENSE)
agcattatgtataaatgttttacgtgtcat
```

Figure 2.9: GetORF Sample Output

more information on the ORFs, such as exactly where it is located, the length of it, which frame it belongs to, what its start and stop codon are and so on. It has a default output of BED (Browser Extensible Data) format, which is a format to describe features on the genome and their positions. It can also output either a DNA, RNA or peptide/protein fasta file and BED12 which is a more detailed version of a BED file. [51]

Moreover, it allows the user to control the minimum and maximum size we would want to look for, since there can be some small non-functional ORFs (having lost their ability to encode functional proteins) that we would like to ignore. ORFipy has a translation table that can be used to help identify possible ORFs based on the most common start and stop codons for the user-selected genetic code. For instance, Mold, Protozoan, and Coelenterate Mitochondrial and Mycoplasma/Spiroplasma (Translation Table 4) have different start and stop codons, which are [TTA, TTG, CTG, ATT, ATC, ATA, ATG, GTG] and [TAA, TAG] respectively, compared to ones in Vertebrate Mitochondrial (Translation Table 2), which are [ATT, ATC, ATA, ATG, GTG] and [TAA, TAG, AGA, AGG]. ORFipy is more flexible in both inputs and outputs, letting us customize it in a way that worked perfectly for our purposes, as described in the Methodology chapter. It also provides a more readable output format (see Figure 2.10) compared to other open reading frame tools like GetORF and ORF Finder.

```
>NC_041448.1:9592-10593_ORF.1 [6-75](+) type:complete length:69 frame:1 start:ATA stop:TAG
ATAAACCCTTTGCTTAAAGGGGCATGATACATTATCATGAAAAAAAAGCTATATGCGAGAAA
ATTTAAA
>NC_041448.1:9592-10593_ORF.2 [171-294](+) type:complete length:123 frame:1 start:ATA stop:TAA
ATAAGTAGAACCTCAACGATCATACGCTTTTGCTTAAATTATTTTAACTTTAATTTTTTATT
TTTAAATAAAATTTTAATTGTTTTTAATAATTTATTCATAAAATGATCTAGTTTATTATAT
>NC_041448.1:9592-10593_ORF.3 [459-519](+) type:complete length:60 frame:1 start:TTA stop:TAA
TTATATATCAAGAAAAAACACAGAAAATGCTTCTATAAGATTTGAACAATCTATAATTCA
>NC_041448.1:9592-10593_ORF.4 [657-738](+) type:complete length:81 frame:1 start:CTG stop:TAG
CTGAACTTCATAATTAATTTTATCATGAGGACAGAAAAGCAGTTCCTTTTAAAATTGAAACT
TTACTTACCGAAAAAAGTT
>NC_041448.1:9592-10593_ORF.5 [19-145](+) type:complete length:126 frame:2 start:TTA stop:TAA
TTAAAGGGGCATGATACATTATCATGAAAAAAAAGCTATATGCGAGAAAATTTAAATAGTCG
TAAGACCTTAAGCAATACAACATTCTTTGCTAAATTTAAACAACTCGCAGGTAATATTTTAT
AT
>NC_041448.1:9592-10593_ORF.6 [292-667](+) type:complete length:375 frame:2 start:ATT stop:TAA
ATTAATAGACAGTTTATGCAACTGACCTTTTGTATGTTTACACGGAATATCTCTACCACAAG
TTCAAATTTTCGTAATGTAAAACCAGGAGGTTATAGCAAAAGTAGTTTACCTCTTATAAATA
AAGAAGACTATGCAATTTTGTATGGTCTCATTTTAGGAGATGCTTATATATCAAGAAAAAAC
ACAGAAAATGCTTCTATAAGATTTGAACAATCTATAATTCATAAAGAATATCTTGAACATTT
ATTTGATATATTTAAATATTTATGTAAAGATGGTGCGACAGTTAAATCAGCAGAAAGAAAAA
AACTATCTACTAAATCTTTTTATTTACTTTGTAACTCGCCAATTAGTAGCTATAACTGAACT
TCA
>NC_041448.1:9592-10593_ORF.7 [775-841](+) type:complete length:66 frame:2 start:ATC stop:TAA
ATCTGGTTATATACTAGATACTTGTTCATTTACTTTGGATGATTTACATTTATTAGTCTCAG
CACT
>NC_041448.1:9592-10593_ORF.8 [95-176](+) type:complete length:81 frame:3 start:ATA stop:TAG
ATACAACATTCTTTGCTAAATTTAAACAACTCGCAGGTAATATTTTATATTAAACATCTAAT
TTTTTGTTTATTTTATAAG
>NC_041448.1:9592-10593_ORF.9 [287-371](+) type:complete length:84 frame:3 start:ATT stop:TAA
ATTATATTAATAGACAGTTTATGCAACTGACCTTTTGTATGTTTACACGGAATATCTCTACC
ACAAGTTCAAATTTTCGTAATG
>NC_041448.1:9592-10593_ORF.10 [518-671](+) type:complete length:153 frame:3 start:ATA stop:TAA
ATAAAGAATATCTTGAACATTTATTTGATATATTTAAATATTTATGTAAAGATGGTGCGACA
GTTAAATCAGCAGAAAGAAAAAAACTATCTACTAAATCTTTTTATTTACTTTGTAACTCGCC
AATTAGTAGCTATAACTGAACTTCATAAT
>NC_041448.1:9592-10593_ORF.11 [710-974](+) type:complete length:264 frame:3 start:ATT stop:TAA
ATTGAAACTTTACTTACCGAAAAAAGTTTAGCTTATTGAGCAATGGATGATGGAGATAACCA
TAAATCTGGTTATATACTAGATACTTGTTCATTTACTTTGGATGATTTACATTTATTAGTCT
CAGCACTTAAAAATAAATTAAATTTAGATTGTTTTATTCATAGTAGAAACAGACTTTATATA
AAAGCTAATTCTAAAAAAGATTTTATTAATTTAGTTAAACCTCACTTTCATCCGACTATGCT
ATACAAACTTAATACA
>NC_041448.1:9592-10593_ORF.12 [726-804](-) type:complete length:78 frame:-1 start:ATG stop:TAA
ATGAACAAGTATCTAGTATATAACCAGATTTATGGTTATCTCCATCATCCATTGCTCAATAA
GCTAAACTTTTTTCGG
>NC_041448.1:9592-10593_ORF.13 [462-615](-) type:complete length:153 frame:-1 start:TTA stop:TAA
TTAGTAGATAGTTTTTTTCTTTCTGCTGATTTAACTGTCGCACCATCTTTACATAAATATTT
AAATATATCAAATAAATGTTCAAGATATTCTTTATGAATTATAGATTGTTCAAATCTTATAG
AAGCATTTTCTGTGTTTTTTCTTGATATA
>NC_041448.1:9592-10593_ORF.14 [291-369](-) type:complete length:78 frame:-1 start:TTA stop:TAA
TTACGAAAATTTGAACTTGTGGTAGAGATATTCCGTGTAAACATACAAAAGGTCAGTTGCAT
AAACTGTCTATTAATA
>NC_041448.1:9592-10593_ORF.15 [57-138](-) type:complete length:81 frame:-1 start:ATA stop:TAG
ATATTACCTGCGAGTTGTTTAAATTTAGCAAAGAATGTTGTATTGCTTAAGGTCTTACGACT
ATTTAAATTTTCTCGCATA
>NC_041448.1:9592-10593_ORF.16 [860-926](-) type:complete length:66 frame:-2 start:ATT stop:TAA
ATTAATAAAATCTTTTTTAGAATTAGCTTTTATATAAAGTCTGTTTCTACTATGAATAAAAC
AATC
>NC_041448.1:9592-10593_ORF.17 [410-482](-) type:complete length:72 frame:-2 start:CTG stop:TAA
CTGTGTTTTTTCTTGATATATAAGCATCTCCTAAAATGAGACCATACAAAATTGCATAGTCT
TCTTTATTTA
```

Figure 2.10: ORFipy Sample Output

2.2.3 Databases

In order to test or gather data, we need several databases that can provide us with genome sequences, allow us to search for certain protein sequences corresponding to ORFs frequently found in group I introns, and that have annotated mitogenome sequences indicating which sections are genes or group I introns. Each database provides their advantages and disadvantages depending on what needs to be extracted from them. The following is a list of databases that I have used throughout the process of this thesis work and I also go over a few that would be interesting to look into in

the future.

NCBI

The first database we will look at is from NCBI, short for National Center for Biotechnology Information. NCBI stores the information of an organism in a server and users can get this information from the various tools it has to offer. Some applications are included such as BLAST, which allows to search in resources such as GenBank and RefSeq databases, which both contain nucleotide sequence information. NCBI has much to offer, and is used by many researchers for testing and evaluating their programs [47]. The database can be found at the given URL: `https://www.ncbi.nlm.nih.gov/`

Rfam

Rfam is an RNA family database that has been used in the context of research on group I and group II introns. This database is essentially a collection that stores multiple sequence alignments from different RNA families. This database helps with annotating some RNA families [18]. RNAs belonging to the same family can have relatively weak sequence conservation but can exhibit a high degree of secondary structure conservation. Rfam is a excellent resource to build covariance models, since it contains curated sequences and alignments for over 4,000 families. Since group I introns have very little conservation at the sequence level, a large number of sequences from different species is necessary to build accurate models. Rfam builds the different families using information from experimentally validated sequence families obtained from the literature. The Rfam team also searches public databases using tools like

the Infernal software package [54] to augment existing families. This database has been updated constantly ever since it was first introduced. The developers published a paper recently, where they presented how they expanded the coverage of metagenomic, viral, and microRNA families [25]. It is currently on version 15.0 as of 2024. The database can be found at the given URL: `https://rfam.org/`

GISSD

There is a database called GISSD, which stands for Group I Introns Sequence and Structure Database. This database keeps track of the different group I introns that have been discovered throughout the past few years. It uses a combination of databases from Comparative RNA Website (CRW), GenBank, and Rfam to gather information about group I introns. The group I intron data from this database can be used with the Infernal program to help separate the data collected from this dataset into the five subgroups [54]. Researchers would submit any new possible group I introns that they may have discovered in their research and these submissions are constantly reviewed by the maintenance team. Some research work has used GISSD as a dataset to test their program, such as in Nawrocki *et al.* [41], Hausner *et al.* [21] and Park *et al.* [43], to name a few. This database can be very useful in training models for group I intron identification. It can be found at the given URL: `http://www.rna.whu.edu.cn/gissd/`.

InterPro

InterPro is a tool and a database used to identify protein domains. It takes protein or translated sequences as input and searches through a comprehensive set of mem-

ber databases to find and annotate potential protein sequences. InterPro provides information on the location of protein domains within the query sequences and identifies the proteins associated with these regions. It integrates data from 13 member databases: CATH-Gene3D, CDD, HAMAP, MobiDB Lite, Panther, Pfam, PIRSF, PRINTS, Prosite, SFLD, SMART, SUPERFAMILY, and NCBIfam. By leveraging these databases, InterPro can identify various protein sequences within the input data.

InterPro offers two versions: an online web version and a version that can be installed locally. The online version is accessible to anyone but has limitations on the amount of data that can be inputted at one time. The local version, which can be used offline and does not have input limits, requires a Linux machine to run. The database can be found at the given URL: `https://www.ebi.ac.uk/interpro/`

AlterORF

AlterORF is a database that contains information regarding alternate open reading frames. This database contains information on over 1.5 million genes in 481 prokaryotic genomes. The authors' goal was to provide a database that could help improve genome annotation and help with the identification of prokaryotic genes that potentially encode proteins in more than one reading frame [44]. However, while this could potentially help us identify or give us a foundation in searching for open reading frames in group I introns, the URL they had provided, `http://www.alterorf.cl/`, does not seem to be accessible and may have been taken down.

Chapter 3

Methodology

3.1 Problem Statement and Research Objectives

Before going into my methodology, I will recapitulate what are the problems that researchers are currently facing when identifying group I introns. Identifying these introns has been tedious for quite some time. The main issue about some of the tools that were mentioned is that they usually do not give an accurate output of the subgroup and positions of group I introns. Some researchers still had to manually confirm if the possible intron sequences that these programs had found could be under group I introns. Moreover, a large portion of these prediction methods are unable to accurately predict the exact start and end positions of group I introns. The predictions made by these tools must be manually curated by a research team to identify the exact boundaries of the introns. This is usually done by aligning an intronless gene with the gene in which an intron was found, and visually inspecting that alignment to identify those boundaries. Currently, there is no software that

would automate identifying group I introns and their exact boundaries with ease. As a result, identifying, detecting, and classifying group I introns requires a huge amount of time and resources to complete one mapping for one genome. Mapping introns is useful because it can help us understand the full picture of how genes work. It helps us know where they are exactly in a gene (which helps to name them, because the naming convention often involves the start position), how they influence the gene's expression, and understanding their location can help researchers discover how genes are regulated.

The 2 main objectives for this thesis project are:

1. Automate the identification of group I intron sequences in fungi mitogenomes and accurately classify them into their subtype.

2. Automate the detection of specific insertion sites (boundaries) of each of the introns inside their host gene.

I wanted to focus on these objectives on fungal mitochondrial genomes, the reason being that introns in fungal mitochondria can be quite complex and numerous. Studying them helps understand how these organisms are affected by these specific mobile elements. Now that I have explained what the current issues are, I can begin with the methodology of this pipeline. I have developed a pipeline that achieves these objectives by building better group I intron models based on new datasets of fungi group I introns and by using a combination of tools.

3.2 Methods

The program, HIPipeline, has two phases of implementation. In the first phase, I needed to improve the existing models of group I introns. The best model available to date was the covariance models of Nawrocki *et al.* used in the context of archaea [41]. Previous research had established that using separate models for each group I intron subtype was more likely to capture the structural diversity of all the subtypes [29], so Nawrocki *et al.* followed the same approach [41]. My goal was to build on the covariance models of Nawrocki *et al.* [41] and add more group I intron subtype data from fungal mitogenomes so that we could get a higher sensitivity and specificity in these types of genomes. The second phase was to build a pipeline that connects Infernal with HMMER, where Infernal would be used to search for all of the possible group I intron hits with the augmented covariance models, and HMMER would be used to identify the gene segments (exons). Clearly identifying exon boundaries, which are easier to detect because coding sequences are well conserved, will help identify where the group I introns are located exactly, since they occupy the space between exons. I will begin by discussing the detailed process of augmenting the group I intron models, before explaining how we create gene models for HMMER.

3.2.1 Phase One: Creating the Models

Phase One A: Augmented Covariance Models

I needed to improve the current models of group I Introns. The latest and best covariance models available at the time were from research work focusing on finding

group I introns in archaea [41]. To improve the current covariance models, I needed to collect more data for each group I intron type that we find in fungi mitogenomes (IA, IB, IC and ID). The data I collected was from fungal mitochondrial sequences that were annotated and had manually curated group I introns. The idea was to use the already identified group I introns and add them to the current covariance models. I also extended the range of the identified introns by an extra 15 nucleotide long at each end. Since it is difficult to identify the exact boundaries of the group I introns, the extra flanking regions were added to make sure that we do not miss any important domain of the group I introns as we build the models. I gathered all the identified group I introns from 60 fully annotated fungal mitogenomes that were publicly available in GenBank (see Table 3.1 for the full list of genomes used for data collection).

Next, I organized the sequences into different fasta files according to type, IA to ID. As I have mentioned before, there is a type IE, and as far as we know, group IE introns are not found in fungi. During the organization, I found that some introns were identified in the annotation as group I in general, but did not specify any type. These ones were discarded to make sure that we do not disrupt the augmented intron models by incorrectly assigning a group I intron to the incorrect type. I have only collected introns that stated at least a type (e.g. IA) or a type and subtype (e.g. IA1). The Table 3.1 shows a summary of how many group I intron types have been collected for each genome. Note that these numbers do not include the group I intron sequences that were discarded for not having their type specified.

The next step involved classifying each group I intron by subtype (either IA1,

Genome	GenBank ID	IA	IB	IC	ID	Date of download
Amoebidium parasiticum mitochondrial chromosome 0555iay58_mtg_cplt	AF538043	0	1	0	0	July-10-2024
Kluyveromyces lactis mitochondrion	AY654900	0	0	0	0	July-10-2024
Chaetomium thermophilum var. thermophilum strain DSM 1495 mitochondrion	JN007486	0	1	0	0	July-10-2024
Peltigera malacea mitochondrion	JN088164	3	8	4	2	July-10-2024
Peltigera membranacea mitochondrion	JN088165	2	7	3	4	July-10-2024
Marssonina brunnea f. sp. 'multigermtubi' mitochondrion	JN204424	0	1	0	0	July-10-2024
Dipodascus magnusii mitochondrion	JQ236859	1	8	0	4	July-10-2024
Microbotryum cf. violaceum BFL-2013 mitochondrion	KC285587	2	8	0	1	July-10-2024
Candida labiduridarum strain NRRL Y-27940 mitochondrion	KC993196	0	1	0	0	July-10-2024
Annulohypoxylon stygium mitochondrion	KF545917	0	1	0	0	July-10-2024
Ganoderma sinense mitochondrion	KF673550	0	0	0	1	July-10-2024
Magnusiomyces magnusii mitochondrion	KJ459953	1	7	0	3	July-10-2024
Hirsutella minnesotensis strain 3608 mitochondrion	KR139916	2	4	2	2	July-10-2024
Sclerotinia sclerotiorum 1980 UF-70 mitochondrion	KT283062	0	1	0	0	July-10-2024
Chrysoporthe deuterocubensis mitochondrion	KT380884	1	7	4	2	July-10-2024
Hypomyces aurantius mitochondrion	KU666552	2	10	4	1	July-10-2024
Epichloe hybrida strain Lp1 mitochondrion	KX066187	0	11	1	2	July-10-2024
Tolypocladium ophioglossoides isolate L2 mitochondrion	KX455872	1	5	1	1	July-10-2024
Saccharomyces kudriavzevii mitochondrion	KX707787	0	1	0	0	July-10-2024
Colletotrichum siamense strain YT02 mitochondrion	KX885103	0	0	0	1	July-10-2024
Bryoria tenuis voucher 46176 mitochondrion	KY369195	0	0	0	0	July-10-2024
Bipolaris cookei strain LSLP18 mitochondrion	MF784482	0	0	0	1	July-10-2024
Ganoderma calidophilum mitochondrion	MH252535	0	0	0	1	July-10-2024
Annulohypoxylon stygium isolate E mitochondrion	MH620794	4	19	6	8	July-10-2024
Taiwanofungus camphoratus mitochondrion	MH745717	3	11	0	3	July-10-2024
Cordyceps cicadae strain CCAD02 mitochondrion	MH922223	3	13	3	3	July-10-2024
Ilyonectria sp. strain s56 mitochondrion	MH924828	0	0	0	1	July-10-2024
Aspergillus pseudoglaucus mitochondrion	MK202802	0	0	0	1	July-10-2024
Saprochaete suaveolens strain NRRL Y-17571 mitochondrion	MK373801	1	12	0	4	July-10-2024
Dactylella sp. strain YMF1.01838 mitochondrion	MK550697	3	10	1	2	July-10-2024
Fomitiporia mediterranea strain MF3/22 mitochondrion	MK623258	0	0	0	1	July-10-2024
Schizopora paradoxa strain KUC8140 mitochondrion	MK623261	0	0	0	1	July-10-2024
Orbilia brochopaga mitochondrion	MK820635	3	4	1	0	July-10-2024
Amanita thiersii mitochondrion	MK993561	1	10	1	4	July-10-2024
Bipolaris oryzae culture ATCC:44560 mitochondrion	MN148434	0	0	0	1	July-10-2024
Ophiocordycipitaceae sp. mitochondrion	MT019333	0	0	0	0	July-10-2024
Cladobotryum mycophilum mitochondrion	MT108299	1	11	4	2	July-10-2024
Trametes coccinea mitochondrion	MT479166	6	20	5	4	July-10-2024
Phallus echinovolvatus mitochondrion	MT528241	1	6	1	2	July-10-2024
Pisolithus microcarpus mitochondrion	MT577034	1	1	0	2	July-10-2024
Ophiostoma himal-ulmi culture CBS:374.67 mitochondrion	MW250274	7	18	13	3	July-10-2024
Ophiocordycipitaceae sp. isolate KR_11D mitochondrion	MW376862	0	0	0	0	July-10-2024
Macrophomina phaseolina mitochondrion	MW557546	1	5	3	3	July-10-2024
Peltigera rufescens mitochondrion	MW711788	0	0	0	1	July-10-2024
Tuber brumale mitochondrion	MW924652	0	1	0	0	July-10-2024
Tuber microsphaerosporum mitochondrion	MW924654	0	1	0	0	July-10-2024
Drechslerella brochopaga mitochondrion	NC_041248	1	4	2	2	July-10-2024
Orbilia dorsalis strain 1835 mitochondrion	NC_041448	0	2	0	0	July-10-2024
Dactylella tenuis mitochondrion	NC_042947	1	16	2	2	July-10-2024
Fomitiporia mediterranea strain MF3/22 mitochondrion	NC_044676	0	1	0	0	July-10-2024
Nemania diffusa mitochondrion	NC_049077	0	0	0	0	July-10-2024
Tuber calosporum mitochondrion	NC_053885	0	2	0	0	July-10-2024
Erysiphe necator strain C mitochondrion	NC_056146	0	1	0	0	July-10-2024
Erysiphe pisi strain Palampur-1 mitochondrion	NC_056147	0	1	0	0	July-10-2024
Clavaria fumosa mitochondrion	NC_056336	4	24	19	4	July-10-2024
Ceratocystiopsis pallidobrunnea strain WIN(M)51 mitochondrion	OM681508	1	3	0	0	July-10-2024
Leptographium aureum culture CBS:438.69 mitochondrion	OQ851464	6	14	13	3	July-10-2024
Grosmannia fruticeta strain WIN(M) 1600 mitochondrion	OQ851465	6	10	7	2	July-10-2024
Leptographium sp. strain WIN(M) 1376 mitochondrion	OQ851466	3	13	9	2	July-10-2024
Podospora anserina	X55026	0	0	0	1	July-10-2024

Table 3.1: List of genomes with corresponding number of group I introns subgroup that were found and added to the dataset.

IA2, IA3, IB1, IB2, IB3, IB4, IC1, IC2, ID). To do this, I used a combination of tools to identify and cross-check the results. First, I used the original Nawrocki *et al.* [41] covariance models with Infernal to classify each intron I had collected to one of these subtypes. However, in some cases, multiple different subtypes would be returned for the same intron. In this case, I selected the best ranked hit (with the best E-value) and used that to classify each intron.

To double check that I was identifying the introns with the correct subtype, I also used the RNAWeasel tool. If both Infernal with the Nawrocki *et al.* [41] covariance model and RNAWeasel gave the same result, then I would keep these introns for augmenting the models. However, if I had mixed results between the two approaches, such as the intron not appearing in RNAWeasel, not being identified with a subtype in Infernal, identified 2 or more subtypes in one large intron, or had a mismatch of subtype, then I discarded the intron.

Once I had finalized filtering and subdividing the list of introns collected, I separated them into different fasta files based on their subtypes. Now that I had organized the collected data, I began to remove possible Open Reading Frames in each intron. Since the goal was to identify only the folding domains corresponding to group I introns, the Open Reading Frames possibly contained in them had to be removed.

I have used multiple resources to help me with identifying any possible Open Reading Frames that each intron I had kept could have. I have tested and used the following tools to help identify any possible Open Reading Frames (ORF): ORF Finder, getORF, and ORFipy. The local version of ORF Finder was difficult to download and install properly, while the web version was only able to take in one

intron sequence at a time. Therefore, I moved on to the next ORF search tool. The second tool, GetORF, was able to take in a few intron sequences, however, it required to be in a specific input format, which was inconvenient to produce easily. It did take some sequences as input and printed out all the possible ORFs it was able to identify with what it was able to take in. It provided different results compared to the previous tool ORF Finder had printed out. In the end, the third tool, ORFipy, was the one I decided to go with to look for the ORFs.

ORFipy was the tool that was able to take in all the intron sequences I currently had in fasta format and was able to identify all the possible ORFs I wanted to look for. I was able to define the minimum size of the ORF, such that I would not have many false positive ORFs that were not likely to code for anything. I kept the size to be at least 50 nucleotides long, since in the research study 'Group I introns are widespread in archaea' [41] had used this minimum range. Once I had a list of all the possible ORFs for each intron sequence from ORFipy, I began to identify what was considered a true positive ORF.

In order to confirm which ORFs are considered a true positive, I used another tool call InterPro. The ORFs needed to contain at least one of the two protein families which are: GIY-YIG or LAGLIDADG. The reason why we are searching for these is because ORFs within group I introns are known to contain code for proteins with GIY-YIG or LAGLIDADG motifs [34]. These proteins are assumed to be essential for mobility (as DNA cutting enzymes) and they may also serve as maturases (promoting splicing). If the identified ORF had at least one of the two protein families returned by InterPro, then I considered it as a true positive ORF and removed them from the

intron sequence. Note that InterPro requires as input translated protein sequences. I used ORFipy to translate the ORF outputs from nucleotide format into amino acid sequences. Once I had the translated sequences, I was able to proceed using the tool InterPro. I then downloaded the results in JSON format and converted it to Excel format. I went through the results and collected all the ORFs that contained a protein family in them. I have also used MFannot to cross-check the results returned by InterPro, as it could also report the presence of GIY-YIG or LAGLIDADG motifs.

Once I had identified the ORFs that contained at least one of the protein families, the next step was to remove these ORFs from the corresponding group I introns sequences. I created a Python script that was taking as input the intron sequences and the start and end positions of the located ORFs to automate this process. After this, only the intron sequences without ORFs remained and were used to augment the covariance models. Table 3.2 shows how many sequences of each subtype I ended up keeping after filtering them following the protocol described here.

After the filtering step, I then started to combine the original covariance models with this new data. To do this, I needed the original fasta files that Nawrocki *et al.* [41] had used for their own models, which were available in the supplementary data of the article. I concatenated my new filtered intron sequences with the ones of [41] in fasta format. I then used the existing covariance model of [41] to convert the fasta files to Stockholm files through Infernal version 1.1.5 [39]. Once I had

Subtype	IA1	IA2	IA3	IB1	IB2	IB3	IB4	IC1	IC2	ID
Total introns found	59	0	1	64	6	3	194	25	82	82

Table 3.2: The number of subtypes that were used at the end of the filtering process described here.

the augmented Stockholm files, I used the Infernal `cmbuild` command to build my augmented covariance models. I then calibrated each model using the `cmcalibrate` command before I can use them to search or scan a genome fasta file. This calibration step is mandatory because it is used to calibrate the E-value parameters for the covariance model. The statistical parameters necessary for reporting E-values are estimated and stored in the calibrated covariance model file. The calibration step takes some time to complete depending on the sequences I provided, such as how conserved the sequences are. With the use of the shared compute clusters of the University of Manitoba and an increased number of cores that would help calibrate the models faster, I was able to complete the calibration step in just a few days.

Once I had calibrated these augmented covariance models, I began some testing to try searching for all possible group I introns in a few mitogenomes that were not used for training (Table 4.1 lists all the genomes that were used for testing). However, a couple of the models I had updated were not able to find any group I intron for those subtypes. The two covariance models I had issues with were IC2 and ID. I carefully double-checked all the sequences that were used for augmenting these two models and looked for outliers such as sequences with unusual sizes, but I could not identify any clear problematic sequences. In the end, I decided to just use the original covariance models of [41] for these two subtypes, as they had better sensitivity. It is possible that the new augmented models for these two subtypes were unable to identify any sequences because they became too strict or they were affected by incorrectly labeled sequences added to them, which caused the sensitivity to drop significantly. Furthermore, note that I did not obtain any new group I intron data

for the subtype IA2 (as shown in Table 3.2), so this model was not augmented and the original model from [41] was used for IA2 detection.

Once I had created the final version of the augmented covariance models with these adjustments, I was able to get good sensitivity for each subtype. I combined and compressed the augmented covariance models under one covariance model, using the `cmpress` command. Here is a summary of everything that was added to the final version of the augmented covariance model (with the exception of IA2, IC2 and ID where nothing was changed for any of them):

- It contains all of the original sequences that were in the original covariance models of [41].
- It contains the updated intron sequence data I had collected with the removed ORFs.
- It contains intron sequences that had no ORFs in the first place.

I then moved on to collect the gene sequences from the same fungal mitogenomes used for training (listed in Table 3.1).

Phase One B: Creating Gene Hidden Markov Models

The idea behind the creation of HMM gene models for fungi mitogenomes is to automate the detection of gene segments (exons), which are easier to identify because their sequences are well conserved, so that we can then accurately find the start and end positions of the group I introns, which should occupy all the space between these exons. I created a Python script to extract all the genes from all the fungal

mitogenomes used for augmenting the intron covariance models (see Table 3.1). From this, I collected 35 different gene types. Out of the 35, I kept 21 of them to use for building a distinct Hidden Markov Model for each of them. The other 14 gene types were discarded because only a single gene sequence was collected for them, which could be an indication of a naming error for the gene or a very rare gene that rarely appears in fungal mitogenomes. Table 3.3 lists the gene names that I have used and collected from the 60 fungal mitogenomes used for training.

Once I had a fasta file for each gene, I aligned them using Clustal Omega [33]. Once I had the alignment files, I then created and built the Hidden Markov Models

Gene Name	Total number of occurrences
atp6	51
atp8	45
atp9	45
cob	46
cytb	5
cox1	48
cox2	48
cox3	48
nad1	48
nad2	48
nad3	46
nad4	49
nad4L	48
nad5	48
nad6	47
rnl	40
rns	43
dpo	2
ND6	2
ND3	2
ND1	2

Table 3.3: List of genes used to create the gene Hidden Markov Models along with the total number of genomes in which a sequence was collected.

with HMMER version 3.4 [14]. To build the Hidden Markov Models, I used the HMMER `hmmbuild` command. These models do not need to be calibrated like in Infernal. Since I had models for many genes, I combined and compressed them under one Hidden Markov Model using the `hmmpress` command.

3.2.2 Phase Two: Building HIPipeline

Once I had the augmented covariance model to use with Infernal and the Hidden Markov Model of genes to use with HMMER, I combined them both into one pipeline. The pipeline takes as input a fasta file in DNA sequence format corresponding to a single genome. The user needs to have both programs Infernal [39] and HMMER [14] installed and as well as Biopython [9] or Anaconda [1].

The pipeline first uses Infernal to scan the given fasta file to provide the user with all of the possible group I intron hits based on the augmented covariance model. Once it finds all of the possible intron hits, it stores all of that data into a dataframe. A dataframe is a 2-dimensional data structure, like a table with columns and rows, where the columns represent the labels and the rows represent the data. Once it has converted the Infernal data into a dataframe, the pipeline then starts filtering the results. It first sorts the Infernal hits by the column '`seq from`' in ascending order. The '`seq from`' column represents the starting position of the intron hit that was identified. Sorting all the hits by their start position allows us to easily identify overlapping hits. When an overlap is detected, the pipeline keeps the hit with the best (lowest) E-value and discards the others. After this first filtering step, no overlapping intron hits remain in the dataframe.

The next step focuses on identifying all the gene segments (i.e. the exons) from the input genome. To do this, the pipeline uses HMMER with the combined Hidden Markov Model of genes, along with a threshold E-value set to 15, instead of the default threshold of 10. The reason for this compromise was to get as many of the gene segments as possible, including mini-exons which can be hard to identify (since they can be less than 30 nucleotides long), while still not increasing it too much to avoid getting too much noise in the results. Once HMMER has identified all of the possible gene segments from the given fasta file, it stores them in a dataframe similar to the one used for the Infernal results. A filtering process is also implemented to get rid of overlapping hits and noise.

More specifically, the HMMER results contain four columns that indicate two different start positions and two different end positions for each gene sequence it identifies. The four columns in question are '`alifrom`' and '`ali to`' (indicating the positions of the alignment found), and '`envfrom`' and '`env to`' (indicating positions of the "envelope" of the hits, which adds a bit of flanking regions on both sides of the alignment). Based on the data from these columns, I have used the columns '`alifrom`' and '`ali to`', since those positions were more in line with the positions found in the GenBank annotations for a few datasets used in early testing. Similarly to the Infernal filtering step, the pipeline first organizes the exon hits by the column '`aliform`' in ascending order. If any of the exon hits overlap, it selects the exon that has the best (lowest) E-value and discards the others.

Once the dataframe has been filtered out in this manner, the second filtering step involved eliminating outliers. It was important here to identify exons that were form-

ing pairs, for example I would want two cob exons as a pair where their hmm position values are consecutive. The hmm values are the start and end points of the reported local alignment in the Hidden Markov Model profile. They are represented under two columns of the HMMER results, which are '`hmmfrom`' and '`hmm to`'. The pipeline would use these values to check if the identified exons of the same gene are consecutive, comparing the first exon '`hmm to`' value with the second exon '`hmmfrom`' value. To account for some variability in the sequences and possible insertions/deletions, we set a threshold of 50 nucleotides between the hmm positions of two hits to be considered as a consecutive pair. Note that the pipeline can identify genes with any number of exons, by continuing to chain together consecutive pairs (*i.e.* the second element of a pair can be the first element of the following pair and so on). Exons identified without a "partner" would be removed, since introns are located in between exons of the same gene, and cannot appear before or after a lone exon. Once the pipeline removed these outliers, it would return a dataframe that lists the best exons that have been identified as groups of consecutive pairs.

Once the pipeline has the two dataframes filtered and organized, one dataframe being the best intron hits and the other being the best groups of exon pairs, the pipeline begins to combine these two sources of information to identify the most accurate start and end positions of the introns. For this, the pipeline checks if either the start or end positions of each intron hit falls in between a pair of exons of the same gene. When that happens, the pipeline determines that the intron should cover the whole space between the exon pair, and its start and end positions are adjusted accordingly (see Equations 3.1 and 3.2). In case several intron hits would fit in

between the same pair of exons, the pipeline would select the intron with the best E-value and discard the other hits.

```
intron start position = end position of the previous exon + 1    (3.1)
```

```
intron end position = start position of the next exon - 1        (3.2)
```

Once the pipeline has mapped the introns to the gaps in between exons of the same gene and updated their start and end positions, both the gene and intron dataframes are merged to produce the final pipeline output. This output takes the form of a table reporting all the information of each exon and group I intron that was found in the given input genome. Figure 3.1 shows an example of the printed output table representing this merged data.

Here is a breakdown of what each column represents in the output table:

- `target name`: represents the name of the exon or group I intron identified in that segment. For example, CMIB4S23 and IC2 represent group I introns (IB4 and IC2, respectively), while cox1, nad1, atp6 and rnl represent exons.
- `query name`: represents the name of the query genome. This would be using the GenBank ID if the input fasta file contains it. Otherwise, it would be using part of the fasta header.
- `from`: represents the start position of the exon or intron. We retrieved this from the '`alifrom`' and '`seqfrom`' columns from their respective dataframes.
- `to`: represents the end position of the exon or intron. We retrieved this from the '`ali to`' and '`seq to`' columns from their respective dataframes.

# target name	query name	from	to	E-value	score	Possible Twin-tron found
cox1	Ceratacystiopsis	3	213	3.3e-58	194.2	-
CMIB4S23	Ceratacystiopsis	214	1718	6.2e-27	146.5	No
cox1	Ceratacystiopsis	1719	2000	1e-81	272.1	-
CMIB4S23	Ceratacystiopsis	2001	3547	3.3e-22	123.4	No
cox1	Ceratacystiopsis	3548	3788	2.7e-73	244.2	-
CMIB4S23	Ceratacystiopsis	3789	4835	6e-18	102.5	No
cox1	Ceratacystiopsis	4836	4971	8.4e-29	96.7	-
CMIB4S23	Ceratacystiopsis	4972	7140	9.2e-14	82	No
cox1	Ceratacystiopsis	7141	7177	0.00022	15.8	-
CMIB4S23	Ceratacystiopsis	7178	8188	8.1e-08	52.9	No
cox1	Ceratacystiopsis	8189	8348	2.2e-40	135.1	-
CMIB4S23	Ceratacystiopsis	8349	9639	3.6e-27	147.7	No
cox1	Ceratacystiopsis	9640	9708	6.6e-10	34.1	-
CMIB4S23	Ceratacystiopsis	9709	11938	2.4e-09	60.4	No
cox1	Ceratacystiopsis	11939	12113	6.6e-36	120.3	-
CMIB4S23	Ceratacystiopsis	12114	13034	9.7e-09	57.4	Yes
CMIB4S23	Ceratacystiopsis	13035	15028	4.3e-18	103.2	Yes
cox1	Ceratacystiopsis	15029	15323	3.2e-61	204.2	-
nad1	Ceratacystiopsis	15470	15608	5.3e-32	107.8	-
CMIA1S23	Ceratacystiopsis	15609	18610	7.9e-15	153.9	Yes
nad1	Ceratacystiopsis	18611	18764	2.4e-40	135.4	-
IC2	Ceratacystiopsis	18765	20107	1.1e-61	230.6	No
nad1	Ceratacystiopsis	20108	20458	9.5e-103	342.1	-
CMIB4S23	Ceratacystiopsis	20459	22329	1e-21	121	No
nad1	Ceratacystiopsis	22330	22763	2.8e-79	264.4	-
atp6	Ceratacystiopsis	27258	27590	3.3e-75	252	-
CMIB4S23	Ceratacystiopsis	27591	28921	4.7e-25	137.3	No
atp6	Ceratacystiopsis	28922	29104	2e-45	153.5	-
IC2	Ceratacystiopsis	29105	30142	1e-59	223.5	Yes
IC2	Ceratacystiopsis	30143	32377	3e-59	221.8	Yes
atp6	Ceratacystiopsis	32378	32584	4.7e-51	172.1	-
rnl	Ceratacystiopsis	46174	46812	3.1e-152	501.3	-
CMIC1S23	Ceratacystiopsis	46813	48304	1.1e-53	170.7	No

Figure 3.1: HIPipeline sample output

- `E-value`: represents the statistical significance of the hit, and more specifically the number of hits one could expect to find by chance in a database of a particular size. The lower the value is, the more significant the hit is. The E-value is related to the '`score`' column, where if the E-value is low, the bit score would be high and vice versa. Note that the E-value reported here directly comes from the HMMER or Infernal result in the corresponding dataframe. Even if the intron positions are adjusted during the mapping to the gaps in between exons, the E-value is not recalculated.

- `score`: represents the bit score, which is the log-odds score for the hit. Higher

scores denote longer and stronger matches. The bit score doesn't depend on the size of the sequence database, only on the profile and the target sequence. This value was retrieved this from the '`score`' column of the corresponding dataframe. As mentioned above, the score was not recalculated when the positions of the introns were adjusted.

- `Possible Twin-tron found`: represents if the intron identified is a possible twin-tron (this will be explained in the next section).

Figure 3.2 provides a step-by-step overview of what HIPipeline is doing internally.

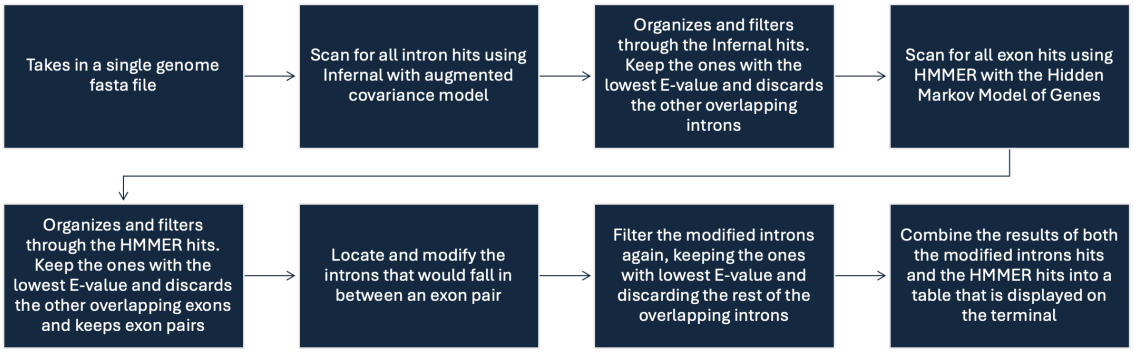


Figure 3.2: Step-by-step overview of the HIPipeline method.

Identifying possible twin-trons

Since mini-exons found in the middle of twin-trons cannot always be identified by HMMER, we wanted to be able to detect regions that could potentially be "tandem" twin-trons and inform the user for them to take a closer look. To accomplish that, the pipeline checks for intronic regions that are larger than the largest introns used for training the augmented covariance model. The threshold value used to determine if

the space between an exon pair is too large is 2850 nucleotides. When these instances are detected, a twin-tron flag is outputted in the corresponding column of the table (see Figure 3.1 for an example).

Whenever a possible twin-tron is identified, the pipeline also checks if multiple introns hits were reported by Infernal in this region. More specifically, it looks for the best two intron hits that are located on either side of the midpoint position of the twin-tron region. The midpoint position, *i.e.* the position that will determine where the first intron ends and the second one starts, is calculated by taking the average of the end position of the first intron hit and the start position of the second intron hit, as determined by the original Infernal hit (see Equation 3.3). The reason for using this formula instead of just taking the actual center of the twin-tron region is that it is unlikely that the two introns have exactly the same size. The assumption we are making is that positions returned by Infernal have the potential of capturing the core regions of the two introns, and finding the midpoint between these two cores could lead to a more accurate placement of the boundary between them.

$$\texttt{midpoint} = \frac{\texttt{1st intron end position} + \texttt{2nd intron start position}}{2} \tag{3.3}$$

More specifically, the pipeline checks how many introns are under each identified possible twin-tron region and applies the following steps.

If there is only one intron hit in that region, the pipeline extends the intron start and end position to occupy the whole gap in between the exon pair. However, it still flags the region to the user as a potential twin-tron in the output table. We do this so that the user can manually double-check for the potential presence of another undetected intron.

If there are exactly two intron hits, the pipeline first sorts them in ascending order based on their start position. It then calculates the midpoint, as explained earlier. Once it has identified the midpoint, it can then extend the two introns' start and end positions. For the first intron, its start position is stretched to the end of the first exon of the pair plus one (see Equation 3.4) and its end position to the midpoint (see Equation 3.5). For the second intron, its start position is stretched to the midpoint plus one (see Equation 3.6) and its end position is stretched to the start of the second exon of the pair minus one (see Equation 3.7).

$$\texttt{1st intron start position = 1st exon of pair end position + 1} \quad (3.4)$$

$$\texttt{1st intron end position = midpoint} \quad (3.5)$$

$$\texttt{2nd intron start position = midpoint + 1} \quad (3.6)$$

$$\texttt{2nd intron end position = 2nd exon of pair start position - 1} \quad (3.7)$$

Finally, if there are more than two intron hits in the twin-tron region, the pipeline picks the best two intron hits based on the E-values, discards the rest, and follows the procedure described for two intron hits.

Figure 3.3 provides a diagram illustrating how the twin-trons are extended. The green and orange lines represent the original hits of the two introns that were identified

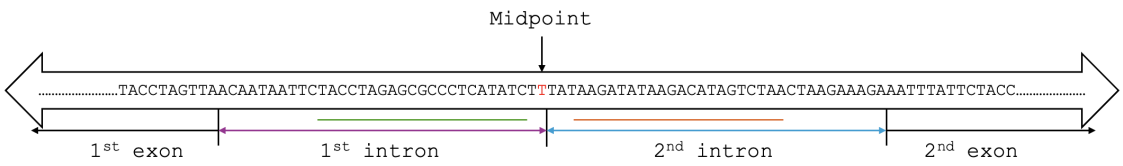


Figure 3.3: Example of Twin-trons being extended.

with Infernal. The midpoint, calculated using the Equation 3.3, is represented by the red nucleotide. The purple and blue lines represent the two introns after extending their start and end positions.

Chapter 4

Results

4.1 Dataset

We used for evaluation a different set of fungal mitochondrial genomes sequenced and annotated by Dr. Hausner's group at the University of Manitoba, but that are not yet published. Studies have shown that the mitogenomes of the Ophiostomatales can vary greatly in size ranging from 23.7 kbps to about 150 kbps [38]. This in part is due to the absence/presence of introns and intron encoded open reading frames (ORFs). A set of mitogenomes representing members from the genus Certosytiopsis and allied taxa were studied as this set contains "small" (30 kbps) and "large" (up to 100 kbps) mitogenomes; representative of sizes one frequently encounters among the Ascomycota. The Ascomycota includes many economic and ecologically important fungi. The long term goal is to understand the evolutionary dynamics of mobile introns and applying mitogenomes for evolutionary investigations to study the taxonomy/evolution of these fungi. The full list of fungal mitogenomes used for evaluation

can be found in Table 4.1. The search methods we are comparing are running Infernal with only either Nawrocki *et al.*'s [41] covariance models or the augmented covariance model, RNAWeasel and the full HIPipeline.

4.2 Results and Findings

4.2.1 Accuracy

In order to compare the accuracy of these search methods, I created a script that would check the precision and sensitivity of the hits when comparing them to the annotated files. To calculate the sensitivity and precision, we used the following equations (Equations 4.1 and 4.2) and their variables:

$$\texttt{sensitivity} = \frac{\texttt{True Positive}}{\texttt{True Positive + False Negative}} \texttt{ * 100} \tag{4.1}$$

$$\texttt{precision} = \frac{\texttt{True Positive}}{\texttt{True Positive + False Positive}} \texttt{ * 100} \tag{4.2}$$

- `True Positive`: Represents the number of annotated intronic nucleotides that were correctly predicted as such by the software.
- `False Negative`: Represents the number of annotated intronic nucleotides that were missing from the software prediction.
- `False Positive`: Represents the number of nucleotides incorrectly predicted to be part of the intron by the software.

We also report how many intron hits were found, how many of these were matched to the correct subtype based on the provided annotation, how many intron hits were

missed, and how many extra hits were identified. The extra hits represent additional intron hits reported by the software but not present in the provided annotation. Note that the values of sensitivity and precision are calculated based on the positions of the intron hits that were found, without including any of the introns that were missed completely. We are also ignoring in our results the group II introns that were in the provided annotations, since HIPipeline was designed to identify group I introns only.

When looking at the results for just the use of Infernal alone, using the Nawrocki *et al.* [41] covariance models (see Table 4.1), the sensitivity results are the worst when comparing them to the other three methods, with the highest value at only 19.73%. This tells us it was missing a significant portion of the intron positions. The precision results were also the worst, but not by much when you compare it to the other three methods, meaning that it predicted some additional nucleotides that were identified incorrectly. It has found a huge number of extra hits, although it was not the method that found the most. It was able to find a large number of the annotated introns, as can be seen in the "Found" column, and it found the entire set of introns in about 88% of the genomes. However, its matched results were lacking, where all the introns were matched to the correct subtype in only 28% of the genomes.

When looking at the results of just using Infernal with the augmented covariance model that I built (see Table 4.2), the sensitivity results were slightly better for certain genomes when comparing it to the previous approach. Still, a large portion of the intron positions were missed, as shown by the sensitivity values being around 21% or less. The precision results were better for certain genomes compared with the previous approach (e.g. *Ceratocystiopsis rollhanseniana* strains WIN(M)101, WIN(M)110

Genome	GenBank ID	Found	Matched	Extra Hits	Sensitivity	Precision
Ceratocystiopsis longispora strain WIN(M)48	Unpublished	9/10	9/9	32	17.970	97.254
Ceratocystiopsis concentrica strain WIN(M)53	Unpublished	8/8	7/8	29	16.031	97.771
Ceratocystiopsis conicicollis strain WIN(M)54	Unpublished	4/4	4/4	10	19.730	97.291
Ceratocystiopsis fasciata strain WIN(M)56	Unpublished	4/4	3/4	11	11.542	97.335
Graphilbum tubicolle strain WIN(M)57	Unpublished	3/3	2/3	3	13.810	96.497
Ceratocystiopsis crenulata strain WIN(M)58	Unpublished	4/4	3/4	6	12.578	98.340
Ceratocystiopsis parva strain WIN(M)59	Unpublished	4/4	3/4	7	14.083	96.450
Ceratocystiopsis rollhanseniana strain WIN(M)101	Unpublished	15/15	13/15	51	16.686	89.847
Ceratocystiopsis rollhanseniana strain WIN(M)110	Unpublished	15/15	13/15	51	16.686	89.847
Ceratocystiopsis rollhanseniana strain WIN(M)113	Unpublished	15/15	13/15	51	16.686	89.847
Ceratocystiopsis rollhanseniana strain WIN(M)117	Unpublished	6/6	5/6	19	15.013	96.541
Ceratocystiopsis manitobensis strain WIN(M)214	Unpublished	23/23	23/23	74	17.731	86.475
Ceratocystiopsis minuta-bicolor strain WIN(M)480	Unpublished	34/34	30/34	102	15.356	96.691
Ophiostoma sp. strain WIN(M)532	Unpublished	20/21	20/20	69	15.007	83.692
Ceratocystiopsis collifera strain WIN(M)908	Unpublished	19/19	19/19	69	14.285	83.570
Ceratocystiopsis minuta strain WIN(M)1453	Unpublished	14/14	13/14	48	17.143	86.555
Ceratocystiopsis minima strain WIN(M)1501	Unpublished	2/2	1/2	3	7.722	96.785
Ceratocystiopsis minuta strain WIN(M)1507	Unpublished	41/42	41/41	126	16.059	93.409
Ceratocystiopsis minuta strain WIN(M)1508	Unpublished	3/3	2/3	14	10.610	96.520
Ceratocystiopsis minuta strain WIN(M)1510	Unpublished	12/12	11/12	42	16.547	85.125
Ceratocystiopsis minuta strain WIN(M)1518	Unpublished	13/13	12/13	46	18.660	91.994
Ceratocystiopsis minuta strain WIN(M)1519	Unpublished	14/14	12/14	49	18.237	92.317
Ceratocystiopsis minuta strain WIN(M)1532	Unpublished	25/25	24/25	93	16.509	86.210
Hyalorhinocladiella strain WIN(M)1643	Unpublished	26/26	26/26	87	17.635	87.353
Hyalorhinocladiella strain WIN(M)1644	Unpublished	7/7	6/7	14	15.521	95.847

Table 4.1: List of fungal mitochondrial genomes with number of introns found by Infernal with the Nawrocki *et al.* [41] covariance models only.

and WIN(M)113), but also slightly worse for other genomes (e.g. the first three genomes at the top of the tables). The augmented covariance model also identified a larger portion of the annotated introns, as all the introns were found in 92% of the genomes. It was able to identify all the introns found to the correct subtype in 84% of the genomes. In general, these results are showing an improvement over the original covariance models, demonstrating the benefit of spending time on creating the augmented covariance models. However, using Infernal alone, even with better models, is clearly not providing the level of accuracy that we are looking for, which motivates the development of the full HIPipeline.

For RNAWeasel (Table 4.3), there was a significant improvement for the sensitivity results compared to the previous two methods, although the best value ob-

Genome	GenBank ID	Found	Matched	Extra Hits	Sensitivity	Precision
Ceratocystiopsis longispora strain WIN(M)48	Unpublished	9/10	9/9	49	19.029	96.069
Ceratocystiopsis concentrica strain WIN(M)53	Unpublished	8/8	8/8	44	16.846	95.557
Ceratocystiopsis conicicollis strain WIN(M)54	Unpublished	4/4	3/4	20	21.035	93.680
Ceratocystiopsis fasciata strain WIN(M)56	Unpublished	4/4	4/4	23	12.905	98.477
Graphilbum tubicolle strain WIN(M)57	Unpublished	3/3	3/3	18	15.439	98.403
Ceratocystiopsis crenulata strain WIN(M)58	Unpublished	4/4	4/4	17	14.407	98.218
Ceratocystiopsis parva strain WIN(M)59	Unpublished	4/4	4/4	14	15.483	89.154
Ceratocystiopsis rollhanseniana strain WIN(M)101	Unpublished	15/15	15/15	58	18.278	94.211
Ceratocystiopsis rollhanseniana strain WIN(M)110	Unpublished	15/15	15/15	58	18.278	94.211
Ceratocystiopsis rollhanseniana strain WIN(M)113	Unpublished	15/15	15/15	58	18.278	94.211
Ceratocystiopsis rollhanseniana strain WIN(M)117	Unpublished	6/6	6/6	26	14.439	96.64
Ceratocystiopsis manitobensis strain WIN(M)214	Unpublished	23/23	22/23	87	19.347	90.709
Ceratocystiopsis minuta-bicolor strain WIN(M)480	Unpublished	34/34	32/34	120	16.704	96.281
Ophiostoma sp. strain WIN(M)532	Unpublished	21/21	21/21	67	15.967	90.191
Ceratocystiopsis collifera strain WIN(M)908	Unpublished	19/19	19/19	73	15.907	88.597
Ceratocystiopsis minuta strain WIN(M)1453	Unpublished	14/14	14/14	66	18.588	94.659
Ceratocystiopsis minima strain WIN(M)1501	Unpublished	2/2	2/2	10	9.723	97.429
Ceratocystiopsis minuta strain WIN(M)1507	Unpublished	41/42	41/41	120	17.234	89.967
Ceratocystiopsis minuta strain WIN(M)1508	Unpublished	3/3	3/3	17	10.832	98.175
Ceratocystiopsis minuta strain WIN(M)1510	Unpublished	12/12	12/12	57	18.068	95.544
Ceratocystiopsis minuta strain WIN(M)1518	Unpublished	13/13	13/13	61	19.093	94.070
Ceratocystiopsis minuta strain WIN(M)1519	Unpublished	14/14	14/14	69	18.297	94.352
Ceratocystiopsis minuta strain WIN(M)1532	Unpublished	25/25	25/25	103	19.008	95.777
Hyalorhinocladiella strain WIN(M)1643	Unpublished	26/26	26/26	97	19.762	91.529
Hyalorhinocladiella strain WIN(M)1644	Unpublished	7/7	7/7	31	16.620	94.022

Table 4.2: List of fungal mitochondrial genomes with number of introns found by Infernal with the augmented covariance model.

tained was still relatively low at 57.088%, and most values were much lower than this. RNAWeasel produced hits that were almost always fully contained within the annotated intron boundaries, as demonstrated by the precision values which are all 100% except in two genomes, where the values were still just over 99%. In terms of identifying all of the introns in the annotation, RNAWeasel did not perform as well as the two previous approaches. It only found all the introns in 56% of the genomes tested. However, in 96% of the genomes (*i.e.* all of them except one), RNAWeasel was able to match all the detected introns to the correct subtype. Only one additional hit was identified in just one of the genomes used for testing (*Ceratocystiopsis minuta-bicolor* strain WIN(M)480), which clearly shows how RNAWeasel is a lot more conservative than the two previous approaches in making its predictions.

Genome	GenBank ID	Found	Matched	Extra Hits	Sensitivity	Precision
Ceratocystiopsis longispora strain WIN(M)48	Unpublished	9/10	9/9	0	31.019	100
Ceratocystiopsis concentrica strain WIN(M)53	Unpublished	8/8	8/8	0	52.668	100
Ceratocystiopsis conicicollis strain WIN(M)54	Unpublished	4/4	4/4	0	32.204	100
Ceratocystiopsis fasciata strain WIN(M)56	Unpublished	3/4	3/3	0	47.511	100
Graphilbum tubicolle strain WIN(M)57	Unpublished	2/3	2/2	0	47.426	100
Ceratocystiopsis crenulata strain WIN(M)58	Unpublished	4/4	4/4	0	33.241	100
Ceratocystiopsis parva strain WIN(M)59	Unpublished	3/4	3/3	0	33.697	100
Ceratocystiopsis rollhanseniana strain WIN(M)101	Unpublished	15/15	15/15	0	37.535	100
Ceratocystiopsis rollhanseniana strain WIN(M)110	Unpublished	15/15	15/15	0	37.535	100
Ceratocystiopsis rollhanseniana strain WIN(M)113	Unpublished	15/15	15/15	0	37.535	100
Ceratocystiopsis rollhanseniana strain WIN(M)117	Unpublished	6/6	6/6	0	57.088	100
Ceratocystiopsis manitobensis strain WIN(M)214	Unpublished	22/23	22/22	0	23.370	100
Ceratocystiopsis minuta-bicolor strain WIN(M)480	Unpublished	34/34	33/34	1	22.624	99.444
Ophiostoma sp. strain WIN(M)532	Unpublished	20/21	20/20	0	20.590	100
Ceratocystiopsis collifera strain WIN(M)908	Unpublished	18/19	18/18	0	16.470	100
Ceratocystiopsis minuta strain WIN(M)1453	Unpublished	14/14	14/14	0	39.139	100
Ceratocystiopsis minima strain WIN(M)1501	Unpublished	1/2	1/1	0	42.012	100
Ceratocystiopsis minuta strain WIN(M)1507	Unpublished	42/42	42/42	0	24.336	99.541
Ceratocystiopsis minuta strain WIN(M)1508	Unpublished	2/3	2/2	0	57.917	100
Ceratocystiopsis minuta strain WIN(M)1510	Unpublished	12/12	12/12	0	42.939	100
Ceratocystiopsis minuta strain WIN(M)1518	Unpublished	13/13	13/13	0	41.740	100
Ceratocystiopsis minuta strain WIN(M)1519	Unpublished	14/14	14/14	0	44.914	100
Ceratocystiopsis minuta strain WIN(M)1532	Unpublished	25/25	25/25	0	31.993	100
Hyalorhinocladiella strain WIN(M)1643	Unpublished	25/26	25/25	0	27.383	100
Hyalorhinocladiella strain WIN(M)1644	Unpublished	6/7	6/6	0	22.813	100

Table 4.3: List of fungal mitochondrial genomes with number of introns found by RNAWeasel.

Now looking at the results for HIPipeline (Table 4.4), the sensitivity values are 2 to 5 times higher than the other three methods, reaching close to 100% for the majority of the genomes. This means that it was almost predicting the entirety of the intron positions for the introns identified on the evaluation dataset. The precision results were not as close to perfection as with RNAWeasel, but they were still very high. Indeed, HIPipeline almost achieved a 100% precision score for each genome, with only five genomes reporting a precision of less than 99%. This means that our new proposed approach is getting a very high sensitivity without overpredicting too many nucleotides outside of the annotated boundaries. For 84% of the genomes tested, all of the annotated introns were found. Note that for the other 16% of the genomes, only 1 to 4 introns were missing. For 88% of the genomes tested, the

Genome	GenBank ID	Found	Matched	Extra Hits	Sensitivity	Precision
Ceratocystiopsis longispora strain WIN(M)48	Unpublished	9/10	9/9	0	99.668	99.974
Ceratocystiopsis concentrica strain WIN(M)53	Unpublished	8/8	8/8	0	99.770	99.990
Ceratocystiopsis conicicollis strain WIN(M)54	Unpublished	4/4	3/4	0	99.611	99.977
Ceratocystiopsis fasciata strain WIN(M)56	Unpublished	4/4	4/4	0	99.684	99.983
Graphilbum tubicolle strain WIN(M)57	Unpublished	3/3	3/3	0	99.699	99.975
Ceratocystiopsis crenulata strain WIN(M)58	Unpublished	4/4	4/4	0	99.869	99.984
Ceratocystiopsis parva strain WIN(M)59	Unpublished	4/4	4/4	0	99.879	99.983
Ceratocystiopsis rollhanseniana strain WIN(M)101	Unpublished	15/15	15/15	0	99.243	99.984
Ceratocystiopsis rollhanseniana strain WIN(M)110	Unpublished	15/15	15/15	0	99.243	99.984
Ceratocystiopsis rollhanseniana strain WIN(M)113	Unpublished	15/15	15/15	0	99.243	99.984
Ceratocystiopsis rollhanseniana strain WIN(M)117	Unpublished	6/6	6/6	0	99.869	99.988
Ceratocystiopsis manitobensis strain WIN(M)214	Unpublished	23/23	23/23	0	99.740	93.371
Ceratocystiopsis minuta-bicolor strain WIN(M)480	Unpublished	31/34	30/31	1	97.124	97.196
Ophiostoma sp. strain WIN(M)532	Unpublished	20/21	20/20	0	99.819	91.733
Ceratocystiopsis collifera strain WIN(M)908	Unpublished	19/19	19/19	0	99.728	99.989
Ceratocystiopsis minuta strain WIN(M)1453	Unpublished	14/14	14/14	0	99.709	99.982
Ceratocystiopsis minima strain WIN(M)1501	Unpublished	2/2	2/2	0	99.897	99.974
Ceratocystiopsis minuta strain WIN(M)1507	Unpublished	38/42	37/38	1	95.589	88.221
Ceratocystiopsis minuta strain WIN(M)1508	Unpublished	3/3	3/3	0	99.899	99.980
Ceratocystiopsis minuta strain WIN(M)1510	Unpublished	12/12	12/12	0	99.705	99.993
Ceratocystiopsis minuta strain WIN(M)1518	Unpublished	13/13	13/13	0	99.745	99.981
Ceratocystiopsis minuta strain WIN(M)1519	Unpublished	14/14	14/14	0	99.673	99.982
Ceratocystiopsis minuta strain WIN(M)1532	Unpublished	25/25	25/25	0	99.737	99.994
Hyalorhinocladiella strain WIN(M)1643	Unpublished	26/26	26/26	0	99.735	93.994
Hyalorhinocladiella strain WIN(M)1644	Unpublished	7/7	7/7	0	99.744	99.698

Table 4.4: List of fungal mitochondrial genomes with number of introns found with HIPipeline.

subtypes were matched perfectly for all the introns found. Overall, the results are showing a clear improvement compared to previous approaches, with our proposed pipeline only struggling a bit more with the genomes *Ceratocystiopsis minuta-bicolor* strain WIN(M)480 and *Ceratocystiopsis minuta* strain WIN(M)1507.

Moreover, it was also restrictive on the additional hits, where it only identified two extra hits overall in the genomes *Ceratocystiopsis minuta-bicolor* strain WIN(M)480 and *Ceratocystiopsis minuta* strain WIN(M)1507. These 2 extra hits were instances where HIPipeline identified group II introns as potential group I introns. In the *Ceratocystiopsis minuta-bicolor* strain WIN(M)480 genome, a group IA2 is reported at the position of a group II intron, which was in the ranges from 54225 to 56450 in between a pair of rnl exons. In the *Ceratocystiopsis minuta* strain WIN(M)1507

genome, we had a hit for a group IB4 at the location of a group II intron which was in the ranges from 2917 to 3510 in between a pair of cox1 exons; this region was also identified as a possible twin-tron.

4.2.2 Running Time

Table 4.5 shows the runtime of every method used in the evaluation on each fungal mitochondrial genome. The running time for each method varies greatly. All of the tests for each method were done on a 2.3 GHz Dual-Core Intel Core i5 computer with 8 GB of memory. From the fastest to the slowest in order, it was RNAWeasel, Infernal with the Nawrocki *et al.* [41] covariance models, Infernal with the augmented covariance models, and HIPipeline. RNAWeasel was the fastest methods out of them, taking a matter of seconds, while for HIPipeline, it varies in ranges from about 30 minutes to about 1 hour and 15 minutes, depending on how large the files are. Note that the majority of the running time of HIPipeline is dominated by the Infernal search with the augmented covariance model, as can be seen by comparing the two corresponding columns. In other words, the extra steps added in the pipeline, such as running HMMER and completing the different filtering steps, only adds a couple of minutes to the runtime.

4.2.3 Discussion

As can be observed from the results, using only Infernal with the Nawrocki *et al.* [41] covariance models has done the worst compared to the other three methods, having very low sensitivity and somewhat high precision. Comparatively, using

Genome	GenBank ID	NCM Infernal	ACM Infernal	RNAWeasel	HIPipeline
Ceratocystiopsis longispora strain WIN(M)48	Unpublished	00:02:38	00:28:29	00:00:18	00:30:13
Ceratocystiopsis concentrica strain WIN(M)53	Unpublished	00:01:19	00:26:31	00:00:22	00:27:57
Ceratocystiopsis conicicollis strain WIN(M)54	Unpublished	00:00:58	00:22:39	00:00:12	00:24:05
Ceratocystiopsis fasciata strain WIN(M)56	Unpublished	00:02:25	00:25:07	00:00:12	00:26:39
Graphilbum tubicolle strain WIN(M)57	Unpublished	00:06:06	00:20:28	00:00:12	00:21:57
Ceratocystiopsis crenulata strain WIN(M)58	Unpublished	00:01:53	00:28:26	00:00:14	00:29:54
Ceratocystiopsis parva strain WIN(M)59	Unpublished	00:02:04	00:24:04	00:00:14	00:25:28
Ceratocystiopsis rollhanseniana strain WIN(M)101	Unpublished	00:01:34	00:33:04	00:00:22	00:34:51
Ceratocystiopsis rollhanseniana strain WIN(M)110	Unpublished	00:01:34	00:33:07	00:00:20	00:34:54
Ceratocystiopsis rollhanseniana strain WIN(M)113	Unpublished	00:01:33	00:32:46	00:00:18	00:34:32
Ceratocystiopsis rollhanseniana strain WIN(M)117	Unpublished	00:00:48	00:26:12	00:00:16	00:27:44
Ceratocystiopsis manitobensis strain WIN(M)214	Unpublished	00:01:22	00:42:41	00:00:20	00:44:48
Ceratocystiopsis minuta-bicolor strain WIN(M)480	Unpublished	00:02:31	01:05:35	00:00:34	01:08:53
Ophiostoma sp. strain WIN(M)532	Unpublished	00:01:28	00:42:36	00:00:26	00:45:10
Ceratocystiopsis collifera strain WIN(M)908	Unpublished	00:02:40	00:41:57	00:00:20	00:45:16
Ceratocystiopsis minuta strain WIN(M)1453	Unpublished	00:01:21	00:28:41	00:00:26	00:30:29
Ceratocystiopsis minima strain WIN(M)1501	Unpublished	00:02:26	00:23:37	00:00:12	00:25:53
Ceratocystiopsis minuta strain WIN(M)1507	Unpublished	00:02:46	01:10:22	00:00:42	01:13:40
Ceratocystiopsis minuta strain WIN(M)1508	Unpublished	00:01:43	00:20:39	00:00:15	00:21:57
Ceratocystiopsis minuta strain WIN(M)1510	Unpublished	00:02:44	00:25:58	00:00:16	00:27:36
Ceratocystiopsis minuta strain WIN(M)1518	Unpublished	00:01:25	00:25:29	00:00:16	00:27:11
Ceratocystiopsis minuta strain WIN(M)1519	Unpublished	00:01:27	00:29:49	00:00:18	00:31:47
Ceratocystiopsis minuta strain WIN(M)1532	Unpublished	00:02:13	00:41:06	00:00:24	00:43:18
Hyalorhinocladiella strain WIN(M)1643	Unpublished	00:01:46	00:44:13	00:00:16	00:46:25
Hyalorhinocladiella strain WIN(M)1644	Unpublished	00:00:38	00:21:20	00:00:16	00:22:51

Table 4.5: List of running time for each genome (Hours:Minutes:Seconds). The tests involving Infernal only are presented in columns "NCM Infernal" and "ACM Infernal", where NCM stands for the Nawrocki *et al.* [41] Covariance Models and ACM stands for Augmented Covariance Models.

Infernal with the augmented covariance models did slightly better in both precision and sensitivity, but it is clear that a large portion of the introns are missed by these approaches. However, the new augmented covariance models showed a significant improvement in its ability to map the introns identified to the correct subtype, as shown in the "Matched" column (see Table 4.2). This result demonstrates the main benefit of creating better covariance models that are more adapted to the genomes of interest (fungal mitogenomes).

RNAWeasel obtained better results in general compared to using Infernal with either of the covariance models. The sensitivity RNAWeasel for identifying intron positions was clearly improved, although still not being ideal with all values below 58%.

Moreover, it obtained the best precision scores out of the four methods. However, it was missing more intron hits from the annotation (11 hits missed in total). Another important result is the substantial reduction in extra hits, from several hundreds in total for Infernal down to only 1 for RNAWeasel in the entire evaluation dataset.

Our proposed approach, HIPipeline, performed the best overall by getting robust results in each category. First of all, it obtained by far the best sensitivity scores for the intron positions out the four methods. This was made possible by combining the Infernal program with the augmented covariance models, and HMMER with the profile Hidden Markov Models of the genes commonly found in fungal mitogenomes. While it has missed a few of the introns that other approaches were able to find, it still missed fewer introns in total than RNAWeasel (9 compared to 11). All the introns found were matched to the correct subtype except 3 of them in total, which is just 2 fewer than RNAWeasel. Moreover, our proposed approach was reporting almost no extra hits, similarly to RNAWeasel. This is a huge advantage as it means that HIPipeline is able to achieve great sensitivity without producing a lot of noise in the results.

Another important observation is that HIPipeline seemed to struggle specifically with two genomes, which were genomes *Ceratocystiopsis minuta-bicolor* strain WIN(M)480 and *Ceratocystiopsis minuta* strain WIN(M)1507. These two genomes were longer and a bit more complex compared to the other genomes, as they contained more group I introns. Upon inspection of the results in these two genomes, it seems that it contained several small exons that were missed by HMMER. Since small exons cannot produce high match scores, they can easily be ignored by the approach.

Moreover, because HIPipeline needs to identify pairs of exons from the same gene to map introns in between, these missed small exons could be the cause of the missed introns. Note that besides these two genomes, the pipeline only missed 2 introns in total.

Interestingly, all the four approaches missed one intron in the genome *Ceratocystiopsis longispora* strain WIN(M)48. However, they were not all the same group I intron. All the methods except RNAWeasel missed the same group I intron of type IB, that ranged from positions 4834 to 6064 and was located in between a pair of cox1 exons. On the other hand, RNAWeasel was able to identify a part of that group IB, but it missed instead a group IA intron that ranged from positions 21239 to 22722 and was located in between a pair of rnl exons.

When considering the running times for all four methods, HIPipeline was the slowest. As mentioned above, most of that runtime comes from running the Infernal search using the new augmented models. When considering that HIPipeline can still complete a search in a fungal mitogenome in about an hour or less and provide results that are orders of magnitude more sensitive and accurate than existing approaches, the trade-off seems to be completely reasonable. Indeed, we argue that using HIPipeline would dramatically reduce the amount of manual curation required by research teams to get a complete and accurate intron landscape, which would save a lot more time overall.

Chapter 5

Conclusion

5.1 Conclusion

One of the main objectives of this thesis was to automate the detection and classification of group I introns into their subtypes with high accuracy, which was achieved. Although a few intron hits were missed (mostly in two genomes, as discussed earlier), HIPipeline was able to identify the majority of them without producing a lot of noise (*i.e.* with very few extra hits). Our results also showed that the augmented covariance models were helping to identify the correct subtypes compared to the existing covariance models used in a study focusing on Archaea [41].

The second main objective was to automate the detection of specific insertion sites (*i.e.* boundaries) of each of the introns inside their host gene. This is where HIPipeline really shined compared to existing approaches. Its accuracy for identifying the exact positions of the introns was demonstrated by very high values for both sensitivity and precision, which no other method could match.

Another goal that was not our main focus in this thesis was to produce a tool that would also run in a reasonable amount of time. In this case, it was achieved as well. Even though our proposed approach was the slowest, it still provides significant time savings when compared to manually annotating a genome and curating the results. HIPipeline has the potential to make it easier for researchers to identify group I introns and proposes a framework that could be replicated to facilitate the search for other types of introns in the future. Once again, it is important to identify group I introns, since they can have important potential applications, such as being "druggable" targets in human pathogenic fungi that are very hard to treat [17] [32].

5.2 Limitations

Throughout the process of this thesis, I was faced with a few challenges. The first challenge was creating augmented models for subtypes IC2 and ID. As I was collecting the intron data to augment the models, IC2 and ID were the models that did not see a benefit from being augmented. Actually, the sensitivity of these augmented models decreased significantly, to the point where no hits were found in some genomes used for testing. When I tried searching for group I introns with the original covariance models for these subtypes, the sensitivity was much higher, finding the potential hits for these subtypes. This tells me that the model I had augmented became too restrictive. I have tried removing some potential data points / sequences that might have caused the issue, such as ones that were way longer than average group I introns of these subtypes. But removing those sequences did not change the results I was getting. It is possible that some of the added sequences from our training dataset for

these subtypes were misannotated, misaligned, or included some ORFs there were not detected or removed, which could have affected the ability to create a good augmented model. However, due to the time it would take to re-calibrate the models for each removal of a set of sequences, it was considered unfeasible to look at all the new sequences to discover the culprits within the timeframe of this thesis. At the end, we decided it was best to use the original covariance models for IC2 and ID from Nawrocki *et al.* [41] in my combined augmented covariance model. In the future, it would be great to have an improved covariance model for subtypes IC2 and ID, so that we can get more accurate hits for these subtypes in fungal mitogenomes.

The second challenge was related to the detection of "tandem" twin-trons, where we have a large intronic segment that actually contains two introns. The reason why this happens is because there is a mini-exon that would be located within the large intronic region, and this is difficult to identify with my Hidden Markov Models when searching for these genes. I was able to overcome this issue by making it such that if the length of the intron is too large, at least over 2850 nucleotide long, then the pipeline would indicate that there is a possible twin-tron and shorten the length of the intron by their midpoints. But the drawback to this is that if there is no twin-tron at that location and it is a large group I intron, this would be giving a false twin-tron flag. At least, with the possible twin-tron column that was added in the results of HIPipeline, the user can take a closer look at it and determine if there are two possible group I introns. Similarly, it seems that some other very small exons were missed in some genomes, which ultimately caused our approach to miss some group I intron hits. In the future, it would be important to try to create a better

model for our gene search or find alternative strategies so that we can identify those smaller gene segment hits (mini-exons).

5.3 Future Work

For future work, I hope to include an augmented model for the subtype IA2. Throughout the process of collecting new data for group I introns, I was not able to get any new intron data for the subtype IA2. I hope in the future to include an augmented covariance model for IA2, along side with subtypes IC2 and ID.

Another future work would be to include a function where it can identify Open Reading Frames, by including the InterPro software into the pipeline. We can use this to help the user to identify any possible Open Reading Frames that might be found within the identified group I introns.

One other future work would be to use this program to identify possible group II introns in addition to group I introns. I would assume the overall framework would be similar to the current proposed pipeline. As far as we know, there is no available group II intron covariance model that has been published, so we would need to create one from scratch. Moreover, the published fungal mitogenomes do not seem to contain a lot of them, so we would not have a lot of data specific to fungi to build our models.

Bibliography

[1] Anaconda Software Distribution. Anaconda. `https://www.anaconda.com/`. Accessed: 2024-10-24.

[2] B. Barnes and J. Dupré. *Genomes and what to make of them*. University of Chicago press, 2009.

[3] L. Barquist, S. W. Burge, and P. P. Gardner. Studying RNA homology and conservation with infernal: from single sequences to RNA families. *Current Protocols in Bioinformatics*, 54(1):12–13, 2016.

[4] A. D. Baxevanis, G. D. Bader, and D. S. Wishart. *Bioinformatics*. John Wiley & Sons, 2020.

[5] A. J. Berk and P. A. Sharp. Sizing and mapping of early adenovirus mrnas by gel electrophoresis of s1 endonuclease-digested hybrids. *Cell*, 12(3):721–732, 1977.

[6] T. R. Cech. Self-splicing and enzymatic activity of an intervening sequence rna from tetrahymena (nobel lecture). *Angewandte Chemie International Edition in English*, 29(7):759–768, 1990.

[7] C. Chen, Q. Li, R. Fu, J. Wang, C. Xiong, Z. Fan, R. Hu, H. Zhang, and

D. Lu. Characterization of the mitochondrial genome of the pathogenic fungus scytalidium auriculariicola (leotiomycetes) and insights into its phylogenetics. *Scientific Reports*, 9(1):1–12, 2019.

[8] B. Cinget and R. R. Bélanger. Discovery of new group id introns leads to creation of subtypes and link to an adaptive response of the mitochondrial genome in fungi. *RNA biology*, 17(9):1252–1260, 2020.

[9] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 03 2009.

[10] D. Copertino and R. B. Hallick. Group ii twintron: an intron within an intron in a chloroplast cytochrome b-559 gene. *The EMBO journal*, 10(2):433–442, 1991.

[11] W. J. d. S. Diniz and F. Canduri. Bioinformatics: an overview and its applications. *Genet Mol Res*, 16(1):10–4238, 2017.

[12] R. G. Drager and R. B. Hallick. A complex twintron is excised as four individual introns. *Nucleic acids research*, 21(10):2389–2394, 1993.

[13] N. M. Druzhyna, G. L. Wilson, and S. P. LeDoux. Mitochondrial dna repair in aging and disease. *Mechanisms of ageing and development*, 129(7-8):383–390, 2008.

[14] S. R. Eddy and the HMMER development team. Hmmer: biosequence analysis using profile hidden markov models. `http://hmmer.org/`. Accessed: 2024-10-24.

[15] J.-L. Ferat and F. Michel. Group ii self-splicing introns in bacteria. *Nature*, 364(6435):358–361, 1993.

[16] R. D. Finn, J. Clements, and S. R. Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[17] R. M. O. d. S. Gomes, K. J. G. d. Silva, and R. C. Theodoro. Group i introns: Structure, splicing and their applications in medical mycology. *Genetics and Molecular Biology*, 47:e20230228, 2024.

[18] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an rna family database. *Nucleic acids research*, 31(1):439–441, 2003.

[19] M. Hafez and G. Hausner. Convergent evolution of twintron-like configurations: one is never enough. *RNA biology*, 12(12):1275–1288, 2015.

[20] G. Hausner, C. Davis, E. Gibb, and J. Reid. Blue-stain fungi associated with the european pine shoot beetle, tomicus piniperda, in red pine in ontario. *Canadian Journal of Plant Pathology*, 25:425, 2003.

[21] G. Hausner, M. Hafez, and D. R. Edgell. Bacterial group i introns: mobile rna catalysts. *Mobile DNA*, 5(1):1–12, 2014.

[22] D. Huchon, A. Szitenberg, S. Shefer, M. Ilan, and T. Feldstein. Mitochondrial group i and group ii introns in the sponge orders agelasida and axinellida. *BMC Evolutionary Biology*, 15:1–14, 2015.

[23] M. Irimia and S. W. Roy. Origin of spliceosomal introns and alternative splicing. *Cold Spring Harbor perspectives in biology*, 6(6):a016071, 2014.

[24] B.-S. Jo and S. S. Choi. Introns: the functional benefits of introns in genomes. *Genomics & informatics*, 13(4):112, 2015.

[25] I. Kalvari, E. P. Nawrocki, N. Ontiveros-Palacios, J. Argasinska, K. Lamkiewicz, M. Marz, S. Griffiths-Jones, C. Toffano-Nioche, D. Gautheret, Z. Weinberg, et al. Rfam 14: expanded coverage of metagenomic, viral and microrna families. *Nucleic Acids Research*, 49(D1):D192–D200, 2021.

[26] N. Kavalecz, N. Ág, L. Karaffa, C. Scazzocchio, M. Flipphi, and E. Fekete. A spliceosomal twin intron (stwintron) participates in both exon skipping and evolutionary exon loss. *Scientific reports*, 9(1):9940, 2019.

[27] D. M. Kupfer, S. D. Drabenstot, K. L. Buchanan, H. Lai, H. Zhu, D. W. Dyer, B. A. Roe, and J. W. Murphy. Introns and splicing elements of five diverse fungi. *Eukaryotic cell*, 3(5):1088–1100, 2004.

[28] B. F. Lang, N. Beck, S. Prince, M. Sarrasin, P. Rioux, and G. Burger. Mitochondrial genome annotation with mfannot: a critical analysis of gene identification and gene model prediction. *Frontiers in Plant Science*, 14:1222186, 2023.

[29] B. F. Lang, M.-J. Laforest, and G. Burger. Mitochondrial introns: a critical view. *Trends in Genetics*, 23(3):119–125, 2007.

[30] F. Lisacek, Y. Diaz, and F. Michel. Automatic identification of group i intron cores in genomic dna sequences. *Journal of molecular biology*, 235(4):1206–1217, 1994.

[31] T. Liu and A. M. Pyle. Discovery of highly reactive self-splicing group ii introns

within the mitochondrial genomes of human pathogenic fungi. *Nucleic Acids Research*, 49(21):12422–12432, 2021.

[32] T. Liu and A. M. Pyle. Highly reactive group i introns ubiquitous in pathogenic fungi. *Journal of Molecular Biology*, 436(8):168513, 2024.

[33] F. Madeira, N. Madhusoodanan, J. Lee, A. Eusebi, A. Niewielska, A. R. N. Tivey, R. Lopez, and S. Butcher. The embl-ebi job dispatcher sequence analysis tools framework in 2024. *Nucleic acids research*, 52(W1):W521—W525, July 2024.

[34] A. H. Megarioti and V. N. Kouvelis. The coevolution of fungal mitochondrial introns and their homing endonucleases (giy-yig and laglidadg). *Genome biology and evolution*, 12(8):1337–1354, 2020.

[35] C. S. Mukhopadhyay, R. K. Choudhary, and M. A. Iquebal. *Basic Applied Bioinformatics*. John Wiley & Sons, 2017.

[36] J. Mukhopadhyay and G. Hausner. Organellar introns in fungi, algae, and plants. *Cells*, 10(8):2001, 2021.

[37] J. Mukhopadhyay and G. Hausner. Interconnected roles of fungal nuclear-and intron-encoded maturases: at the crossroads of mitochondrial intron splicing. *Biochemistry and Cell Biology*, 102(5):351–372, 2024.

[38] J. Mukhopadhyay, A. Wai, and G. Hausner. The mitogenomes of leptographium aureum, leptographium sp., and grosmannia fruticeta: expansion by introns. *Frontiers in Microbiology*, 14:1240407, 2023.

[39] E. Nawrocki and S. Eddy for the INFERNAL development team. Infernal: inference of rna alignments. `http://eddylab.org/infernal/`. Accessed: 2024-10-24.

[40] E. P. Nawrocki. Annotating functional rnas in genomes using infernal. *RNA sequence, structure, and function: computational and bioinformatic methods*, pages 163–197, 2014.

[41] E. P. Nawrocki, T. A. Jones, and S. R. Eddy. Group i introns are widespread in archaea. *Nucleic acids research*, 46(15):7970–7976, 2018.

[42] H. Nielsen and S. D. Johansen. Group i introns: Moving in new directions. *RNA Biology*, 6(4):375–383, 2009.

[43] S.-H. Park, J. A. Kyndt, and J. K. Brown. Comparison of auxenochlorella protothecoides and chlorella spp. chloroplast genomes: Evidence for endosymbiosis and horizontal virus-like gene transfer. *Life*, 12(3):458, 2022.

[44] I. Pedroso, G. Rivera, F. Lazo, M. Chacon, F. Ossandon, F. A. Veloso, and D. S. Holmes. Alterorf: a database of alternate open reading frames. *Nucleic acids research*, 36(suppl_1):D517–D518, 2007.

[45] S. Prince, C. Munoz, F. Filion-Bienvenue, P. Rioux, M. Sarrasin, and B. F. Lang. Refining mitochondrial intron classification with erpin: Identification based on conservation of sequence plus secondary structure motifs. *Frontiers in Microbiology*, 13, 2022.

[46] I. T. Rombel, K. F. Sykes, S. Rayner, and S. A. Johnston. Orf-finder: a vector for high-throughput gene identification. *Gene*, 282(1-2):33–41, 2002.

[47] C. L. Schoch, S. Ciufo, M. Domrachev, C. L. Hotton, S. Kannan, R. Khovanskaya, D. Leipe, R. Mcveigh, K. O'Neill, B. Robbertse, et al. Ncbi taxonomy: a comprehensive update on curation, resources and tools. *Database*, 2020, 2020.

[48] C. H. Sellem, G. e. l. Lecellier, and L. e. o. Belcour. Transposition of a group ii intron. *Nature*, 366(6451):176–178, 1993.

[49] O. Shaul. How introns enhance gene expression. *The international journal of biochemistry & cell biology*, 91:145–155, 2017.

[50] P. Sieber, M. Platzer, and S. Schuster. The definition of open reading frame revisited. *Trends in Genetics*, 34(3):167–170, 2018.

[51] U. Singh and E. S. Wurtele. orfipy: a fast and flexible tool for extracting orfs. *Bioinformatics*, 37(18):3019–3020, 2021.

[52] J. E. Stajich, F. S. Dietrich, and S. W. Roy. Comparative genomic analysis of fungal genomes reveals intron-rich ancestors. *Genome biology*, 8(10):1–13, 2007.

[53] D. L. Wheeler, D. M. Church, S. Federhen, A. E. Lash, T. L. Madden, J. U. Pontius, G. D. Schuler, L. M. Schriml, E. Sequeira, T. A. Tatusova, et al. Database resources of the national center for biotechnology. *Nucleic acids research*, 31(1):28–33, 2003.

[54] Y. Zhou, C. Lu, Q.-J. Wu, Y. Wang, Z.-T. Sun, J.-C. Deng, and Y. Zhang. GISSD: group I intron sequence and structure database. *Nucleic acids research*, 36(suppl_1):D31–D37, 2008.

[55] M. Zvelebil and J. O. Baum. *Understanding bioinformatics*. Garland Science, 2007.