BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NÔNG LÂM TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ SINH HỌC

****

KHÓA LUẬN TỐT NGHIỆP

PHÁT HIỆN MARKER MICROSATELLITE TỪ CƠ SỞ DỮ LIỆU TRÌNH TỰ EST (Expressed Sequence Tags) CỦA CÂY XOÀI (*Mangifera indica*)

Ngành học: CÔNG NGHỆ SINH HỌC

Niên khóa: 2002-2006

Sinh viên thực hiện: NGUYỄN MINH HIỀN

Thành phố Hồ Chí Minh

Tháng 8/2006

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM THÀNH PHỐ HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ SINH HỌC

****

PHÁT HIỆN MARKER MICROSATELLITE TỪ CƠ SỞ DỮ LIỆU TRÌNH TỰ EST (Expressed Sequence Tags) CỦA CÂY XOÀI (*Mangifera indica*)

Giáo viên hướng dẫn:
TS. BÙI MINH TRÍ

Sinh viên thực hiện:
NGUYỄN MINH HIỀN

Thành phố Hồ Chí Minh
Tháng 8/2006

LỜI CẢM TẠ

Xin gửi lòng biết ơn sâu sắc đến ba mẹ và gia đình đã hết lòng hỗ trợ, động viên về mọi mặt để tôi hoàn thành đề tài.

Tôi xin cảm ơn

- Ban Giám hiệu trường Đại học Nông Lâm Thành phố Hồ Chí Minh
- Ban Giám đốc Trung tâm Phân tích Thí nghiệm Trường Đại học Nông Lâm Thành phố Hồ Chí Minh
- Ban chủ nhiệm Bộ Môn Công nghệ Sinh học cùng toàn thể Quý Thầy Cô đã truyền đạt kiến thức cho tôi trong suốt quá trình học tập tại trường.

Tôi xin gửi lòng biết ơn sâu sắc đến

TS. Bùi Minh Trí

Đã tận tình hướng dẫn tạo điều kiện tốt nhất cho tôi trong suốt quá trình thực hiện đề tài và hoàn thành luận văn tốt nghiệp này.

Tôi chân thành cảm ơn đến:

- Thầy Lưu Phúc Lợi
- Các anh chị đang làm việc tại Trung tâm Phân tích Hóa Sinh
- Các bạn trong lớp CNSH28

Đã giúp đỡ, hỗ trợ, động viên, chia sẻ những buồn vui trong suốt thời gian tôi thực tập và thực hiện đề tài.

Tp. Hồ Chí Minh tháng 08 năm 2006

Sinh viên thực hiện

Nguyễn Minh Hiền

TÓM TẮT

NGUYỄN MINH HIỀN, Đại học Nông Lâm Thành phố Hồ Chí Minh. Tháng 8/2006. "PHÁT HIỆN MARKER MICROSATELLITE TỪ CƠ SỞ DỮ LIỆU TRÌNH TỰ EST (Expressed Sequence Tags) CỦA CÂY XOÀI (*Mangifera indica*)".
Giảng viên hướng dẫn:
TS. BÙI MINH TRÍ

Thời gian nghiên cứu: từ tháng 2 đến tháng 7 năm 2006

Địa điểm nghiên cứu: Trung tâm Phân tích Thí Nghiệm - trường Đại học Nông Lâm TP. Hồ Chí Minh

Hiện nay với sự phát triển của khoa học kỹ thuật cùng với sự kết hợp liên thông giữa các ngành khoa học đã mở ra những thuận lợi to lớn cho việc nghiên cứu và phát triển. Tin sinh học – một ngành khoa học mới ra đời với mục đích hỗ trợ, cung cấp thông tin dữ liệu sẽ là một công cụ hữu ích giúp giải quyết những vấn đề khó khăn trong nghiên cứu sinh học trên thực tế.

Cây xoài là loại cây ăn quả nhiệt đới quan trọng ở Việt Nam có giá trị kinh tế cao. Chính vì thế việc xác định các giống xoài, phân tích sự đa dạng di truyền, lập bản đồ các gen trong bộ gen là mục tiêu hiện nay. Với các ưu điểm của một marker rất hữu dụng trong nghiên cứu di truyền, chúng tôi đã tiến hành xây dựng phương pháp phát hiện marker microsatellite từ nguồn cơ sở dữ liệu EST hiện có.

Phương pháp: chúng tôi đã sử dụng các chương trình Perl est_trimmer.pl, misa.pl, phần mềm BioEdit với công cụ CAP contig assembly program, phần mềm Primer3 và gói công cụ ssrfinder_1_0.

Kết quả đạt được:

Tải được các trình tự EST của cây xoài có trong nguồn cơ sở dữ liệu của NCBI

Xác định được 267 microsatellite bao gồm các dạng dinucleotide (4.12%), trinucleotide (95.51%) và tetranucleotide (0.37%)

Xác định vùng bảo tồn và thiết kế primer cho 6 loại microsatellite là các loại microsatellite sau CAA, CCA, CAT, TCA, TCT, TGA

SUMMARY

HIEN NGUYEN MINH, Nong Lam University, Ho Chi Minh City. August, 2006. "DEVELOPMENT OF MICROSATELLITE MARKER FROM EST (Expressed Sequence Tags) SEQUENCE DATABASE OF MANGO TREE (*Mangifera indica*)".

Supervisor:
Dr. TRI BUI MINH

The research was carried out at the Chemical and Biological Analysis and Experiment Center at Nong Lam University.

Nowadays the development of science and technology together with the combination of different research field have created great advantages for research. Bioinformatics – a new field that support speed up information processing will be an useful tool to deal with problems in biology research.

Mango tree is an important tropical fruit tree in Vietnam, it has high economic value. Therefore the identification of mango genus, the analysis of genetic diversity, gene mapping are the current goal. Because of useful marker, our objective is to develop an in-silico method in order to identify microsatellite marker from EST database.

Methodology: we used Perl scripts such as est_trimmer.pl, misa.pl, BioEdit software with CAP contig assembly program, Primer3 software and the package tool – ssrfinder_1_0.

Result:

Download EST sequences from NCBI database

Identify 267 microsatllite include dinucleotide (4.12%), trinucleotide (95.51%) and tetranucleotide (0.37%)

Identify consensus region and design primer for 6 sorts: CAA, CCA, CAT, TCA, TCT, TGA.

MỤC LỤC

DANH SÁCH CÁC CHỮ VIẾT TẮT

- AFLP Amplified Fragment Length Polymorphism
- BAC Bacterial Aritificial Chromosome
- bp base pair
- cDNA complementary DNA
- CIB Center Information Biology
- DDBJ DNA Data Bank Japan
- DNA Deoxyribonucleic acid
- EBI European Bioinformatics Institute
- EMBL European Molecular Biology Laboratory
- EST Expressed Sequence Tag
- IAM Infinite Alleles Model
- kb kilo base
- Mb mega base
- MISA Microsatellite identification tool
- NIG National Institute of Genetics
- NIH National Institute of Health
- NCBI National Center for Biotechnology Information
- PCR Polymerase Chain Reaction
- PDBj Protein Database Japan
- PIR Protein Information Resource
- RAPD Random Amplified Polymorphic DNA
- SMM Stepwise Mutation Model
- SSR Simple Sequence Repeat
- SSRIT Simple Sequence Repeat Identification Tool
- UTR unstranlated region
- YAC Yeast Artificial Chromosome

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC HÌNH

HÌNH TRANG

Phần 1

MỞ ĐẦU

1.1. Đặt vấn đề

Hiện nay với sự phát triển của khoa học kỹ thuật cùng với sự kết hợp liên thông giữa các ngành khoa học đã mở ra những thuận lợi to lớn cho việc nghiên cứu và phát triển. Tin sinh học – một ngành khoa học mới ra đời với mục đích hỗ trợ, cung cấp thông tin dữ liệu sẽ là một công cụ hữu ích giúp giải quyết những vấn đề khó khăn trong nghiên cứu sinh học trên thực tế.

Xoài là cây ăn quả nhiệt đới quan trọng ở nước ta chúng được trồng phổ biến ở nhiều vùng miền trong cả nước. Cây xoài vừa có giá trị dinh dưỡng vừa có giá trị kinh tế cao, từ quả xoài, rễ xoài,… đến lá xoài đều là nguồn thu lợi ích cho người trồng. Chính vì thế việc xác định các giống xoài, phân tích sự đa dạng di truyền, lập bản đồ các gen trong bộ gen là mục tiêu hiện nay

Hiện này microsatellite là một marker rất hữu dụng trong việc lập bản đồ phân tử, xác định các giống cây trồng, đánh giá nguồn gốc tổ tiên của cây trồng cho mục đích nghiên cứu quần thể cây trồng và nghiên cứu quá trình tiến hóa. Nguyên nhân là do microsatellite có những ưu điểm vượt trội so với những marker khác như biểu hiện số lượng lớn sự đa hình, là marker đồng trội nên có thể phân biệt được dị hợp tử. Một thuận lợi to lớn nữa của marker microsatellite là có thể phát triển in silico (trên máy tính) dựa vào các phần mềm tin sinh học. Vì vậy có thể giảm chi phí và thời gian cho việc phát hiện microsatellite so với cách thực hiện bằng thực nghiệm.

Dựa trên những cơ sở đó, chúng tôi thực hiện đề tài "**Phát hiện marker microsatellite từ cơ sở dữ liệu trình tự EST (Expressed Sequence Tags) của cây xoài (*Mangifera indica*).**"

1.2. Mục đích và yêu cầu

1.2.1. Mục đích

Xây dựng phương pháp phát hiện microsatellite đối với cây xoài từ nguồn cơ sở dữ liệu EST hiện có, cho phép tạo ra công cụ phân tích, nhận diện, so sánh các giống xoài.

1.2.2. Yêu cầu

Tìm kiếm và tải được hầu hết các trình tự EST của cây xoài hiện có trên các cơ sở dữ liệu.

Phát hiện các kiểu SSR phổ biến từ EST có được.

Thiết kế các primer phù hợp cho phép phát hiện ra các SSR kể trên bằng công cụ PCR.

1.3. Giới hạn

Cơ sở dữ liệu trình tự sinh học giới hạn ở NCBI.

Quy trình thực hiện chỉ tiến hành trên đối tượng là cây xoài.

Phần 2

TỔNG QUAN TÀI LIỆU

2.1. Giới thiệu về tin sinh học (bioinformatics)

2.1.1. Định nghĩa

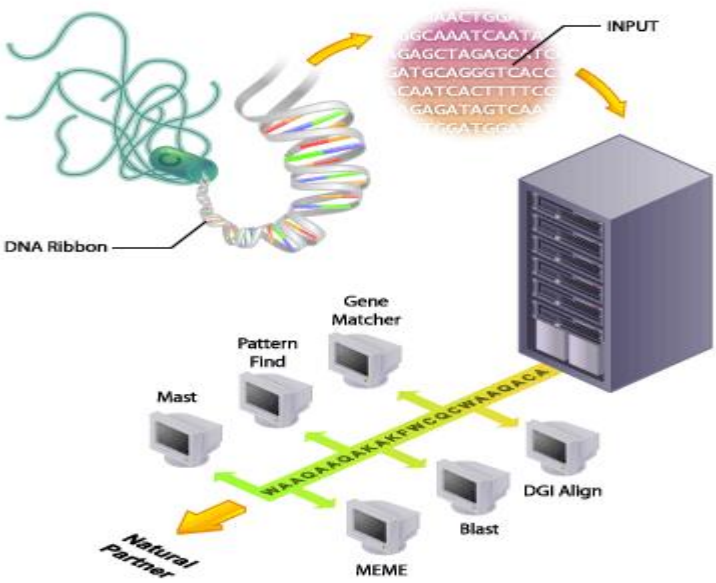
Sự kết hợp, liên thông giữa các ngành giúp cho khoa học có những bước phát triển mới. Trong thời đại khoa học kỹ thuật ngày nay, sự kết hợp giữa các ngành lại với nhau là rất cần thiết. Không một ngành khoa học nào có thể phát triển mà không cần sự hỗ trợ của ngành khác. Bioinformatics hay tin sinh học là một ví dụ rất điển hình của sự liên kết này và kết quả đạt được từ ngành khoa học này là rất khả quan.

Theo NCBI (National Center for Biotechnology Information – Trung Tâm Thông Tin Quốc gia về Công Nghệ Sinh Học) tin sinh học là sự kết hợp giữa công nghệ sinh học và công nghệ thông tin với mục tiêu giúp hiểu biết và khám phá những nguyên lý trong sinh học.

2.1.2. Mối quan hệ giữa sinh học và tin học

Tin học có ảnh hưởng sâu sắc đến sinh học, thông thường, những người làm tin sinh học sử dụng những kiến thức hay/và công cụ trong tin học để giải quyết những vấn đề trong sinh học. Ví dụ, người ta tiến hành xây dựng những cơ sở dữ liệu nhằm quản lý và khai thác một lượng lớn các dữ liệu sinh học phân tử (nucleotide, amino acid).

Mặt khác, sinh học cũng có những tác động ngược lại đến tin học. Ví dụ xây dựng mạng nơron (neural network) bằng cách mô phỏng bộ não của con người, hay thiết kế các thuật toán di truyền (genetic algorithms) dựa vào mô phỏng quá trình tiến hóa của các loài sinh vật.



Hình 2.1. Sử dụng máy tính để xử lý các thông tin sinh học

2.1.3. Tầm quan trọng của tin sinh học

Với sự phát triển mạnh trong cả hai lĩnh vực là công nghệ sinh học và công nghệ thông tin, ngày nay một khối lượng khổng lồ dữ liệu sinh học phân tử được thu thập và phục vụ cho quá trình nghiên cứu. Một trong những ví dụ tiêu biểu nhất là sự hoàn thành việc giải mã bản đồ gen của người (human genome) vào năm 2003. Bộ gen của người bao gồm khoảng 3 tỷ nucleotide và được lưu trữ dưới dạng số hóa.

Tuy nhiên, việc giải mã thành công bộ gen của người hay các sinh vật khác như chuột hay lúa mới chỉ là bước đầu tiên trong quá trình tìm hiểu về bản chất phức tạp của sự sống. Việc giải mã thành công bộ gene người được so sánh như việc chúng ta tìm ra bức thư của tạo hóa nói về cấu tạo cũng như chức năng của các bộ phận trong cơ thể con người, tuy nhiên nội dung của bức thư trên lại được viết bởi ngôn ngữ tự nhiên (natural language) mà chúng ta chưa hiểu được. Mục tiêu và thách thức của chúng ta hiện tại cũng như trong tương lai là từng bước tìm hiểu và dịch nội dung của bức thư trên sang dạng ngôn ngữ mà con người có thể hiểu được.

Ngôn ngữ tự nhiên như mọi ngôn ngữ khác, ngôn ngữ này bắt đầu từ các ký tự chữ cái (amino acid), đến các từ (motif), các câu (protein) và ngữ pháp (cấu trúc protein).

Bằng cách sử dụng các phương pháp sinh học tính toán chúng ta đã có thể nhận diện được các từ của ngôn ngữ - các amino acid. Tuy nhiên, bằng cách này chúng ta

vẫn chưa có khả năng để nhận diện được các quy tắc ngữ pháp phức tạp và chặt chẽ của nó - cấu trúc protein.

Vì vậy việc nhận diện các quy tắc ngữ pháp vẫn phải dựa vào các thực nghiệm hóa lý. Hạn chế của cách tiếp cận thực nghiệm là đắt tiền và mất nhiều thời gian.Từ đó thúc đẩy các nhà nghiên cứu tiếp tục tìm ra các quy tắc ngữ pháp để có thể hiểu được nội dung các câu đã có - hiểu được protein và tự viết ra một câu mới - tự thiết kế một protein.

2.1.4. Mục tiêu của tin sinh học

- Tổ chức dữ liệu để quản lý và truy cập thông tin
- Phát triển các công cụ và tài nguyên hỗ trợ phân tích dữ liệu sinh học, ví dụ như so sánh trình tự protein đặc thù với các trình tự đã biết rõ chức năng
- Dùng những công cụ này để phân tích dữ liệu và diễn giải kết quả theo ý nghĩa trong sinh học.

2.1.5. Vai trò của tin sinh học

Sự phát triển của tin sinh học cho phép mở rộng những phân tích sinh học theo 2 chiều, sâu và rộng.

Theo bề sâu sẽ bao gồm các nghiên cứu nhằm hiểu biết ngày càng nhiều các protein. Bắt đầu với một gen, xác định chuỗi protein, từ đó dự đoán cấu trúc của protein. Dựa vào các tính toán hình học có thể dự đoán hình dạng và bề mặt protein, mô phỏng phân tử, nhận diện liên kết, và suy đoán chức năng protein. Thực tế, những bước trung gian vẫn khó thực hiện chính xác, và cần kết hợp với những phương pháp khác để đạt kết quả mong muốn.

Theo chiều rộng sẽ bao gồm các phương pháp so sánh gen này với gen khác, protein này với protein khác. Ban đầu là những thuật giải đơn giản được dùng để so sánh chuỗi và cấu trúc của cặp protein liên quan. Khi dữ liệu sinh học gia tăng mạnh mẽ sẽ phát sinh nhu cầu cải tiến các thuật giải có hiệu suất cao để sắp gióng cột nhiều trình tự, phân lập mẫu chuỗi hay mẫu cấu trúc xác định họ protein, tạo cây phát sinh loài để khảo sát quá trình tiến hoá của protein. Cuối cùng, do thông tin được lưu trong cơ sở dữ liệu lớn, công việc so sánh trở nên phức tạp hơn, đòi hỏi nhiều cải tiến trong cơ chế tổ chức và quản lý cơ sở dữ liệu.

2.1.6. Một số bài toán lớn trong tin sinh học

Bài toán đầu tiên và hết sức quan trọng mà chúng ta phải giải quyết là xây dựng các cơ sở dữ liệu (database) để quản lý và khai thác một cách hiệu quả các dữ liệu về sinh học phân tử mà chúng ta đã thu thập được. Hai cơ sở dữ liệu nổi tiếng và được nhiều người dùng là cơ sở dữ liệu sinh học Châu Âu (EBI) và cơ sở dữ liệu sinh học quốc gia Mỹ (NCBI). Bên cạnh hai cơ sở dữ liệu sinh học trên, nhiều cơ sở dữ liệu sinh học khác đã, đang và sẽ được xây dựng nhằm phục vụ cho nhiều mục đích khác nhau và riêng biệt.

Một câu hỏi mà tất cả chúng ta đều muốn tìm hiểu và trả lời đó là nguồn gốc và quá trình tiến hóa của các loài sinh vật nói chung và con người nói riêng (evolution process). Ngày nay, việc nghiên cứu quá trình tiến hóa của các loài sinh vật chủ yếu dựa vào các dữ liệu sinh học phân tử bởi chúng thường cho kết quả chính xác cao hơn các loại dữ liệu khác. Ví dụ, xây dựng cây tiến hóa để tìm hiểu mối quan hệ tiến hóa giữa các loài sinh vật (phylogenetic tree reconstruction) là một bài toán hết sức thú vị và đang được sự quan tâm của nhiều nhà nghiên cứu trên thế giới.

Tìm hiểu mối quan hệ giữa các chuỗi sinh học phân tử (pairwise alignment, multiple alignment) là một trong những mục tiêu cơ bản và quan trọng trong tin sinh học. Dựa vào mối quan hệ giữa các chuỗi sinh học phân tử (gene hay protein) chúng ta có thể chẩn đoán được chức năng hay cấu trúc cho các chuỗi phân tử mới phát hiện (gene/protein function prediction).

Chẩn đoán cấu trúc bậc cao của các chuỗi sinh học phân tử (RNA/protein high structure prediction) là một bài toán hết sức quan trọng (tuy nhiên rất khó) trong tin sinh học bởi vì chức năng của các chuỗi phân tử được quyết định bởi cấu trúc không gian của chúng (tertiary structure). Với các công nghệ sinh học ngày nay, cấu trúc bậc một của RNA hay protein (RNA/protein primary structure) được xác định một cách đơn giản và hiệu quả, tuy nhiên, để tìm được cấu trúc bậc cao của RNA hay protein cần tốn nhiều thời gian và chi phí cao. Để giúp đỡ giải quyết vấn đề trên, người ta xây dựng các thuật toán để chẩn đoán cấu trúc không gian dựa vào thông tin về cấu trúc bậc một của chúng.

2.2 Khái quát về dữ liệu trình tự

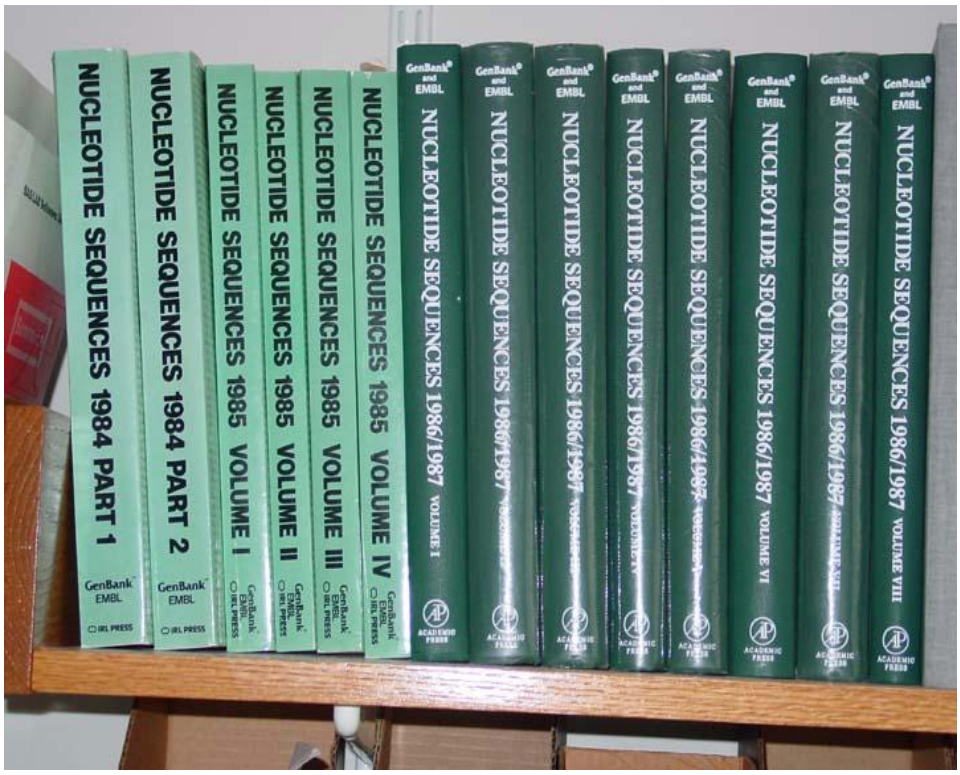
2.2.1 Lịch sử

Hơn ba thập kỷ trước của thế kỷ 20, có một sự thúc đẩy các nhà sinh học (hay nhà khoa học nói chung) tìm hiểu bằng cách nào hàng triệu hay hàng tỉ những đơn vị trong bộ gen của sinh vật chứa đựng tất cả các thông tin. Mà các thông tin này cần cho tế bào để tạo nên vô số tiến trình trao đổi chất thiết yếu cho sự sống của sinh vật, và được truyền từ thế hệ này sang thế hệ khác. Để có một sự hiểu biết cơ bản làm sao sự tập hợp các đơn vị nucleotide riêng biệt điều khiển sự sống, một số lượng lớn các dữ liệu trình tự phải được thu thập và lưu giữ theo một cách mà những dữ liệu này có thể được tìm kiếm và phân tích dễ dàng.

Lịch sử của dữ liệu trình tự bắt đầu từ những năm 1960, khi Margaret Dayhoff và cộng sự ở PIR (Protein Information Resource) thu thập tất cả trình tự protein đã biết lúc bấy giờ; nhóm của bà đã xuất bản sự thu thập này dưới dạng một cuốn sách có tên là “Atlas of Protein Sequence and Structure”. Khi số lượng đáng kể của những trình tự nucleotide đã có sẵn, những dữ liệu này được liệt kê trong Atlas. (Cần phải nhớ rằng vào thời điểm lịch sử của sinh học này, những trình tự protein được chú trọng hơn là những trình tự DNA.) Khi Atlas được mở rộng, nó bao gồm sự miêu tả ở dạng văn bản để cung cấp những trình tự protein cũng như những thông tin liên quan đến sự tiến hóa của nhiều họ protein.

Khoảng năm 1972 số lượng dữ liệu chứa trong Atlas không còn rộng khắp, và nhu cầu nó ở định dạng điện tử là điều hiển nhiên. Nội dung của Atlas được sắp xếp bằng điện tử bởi PIR trên các băng từ, và sự sắp xếp này bao gồm một vài chương trình cơ bản mà có thể được sử dụng để tìm và đánh giá mối quan hệ tiến hóa xa.

Sự tiến bộ của dữ liệu trình tự DNA vào năm 1982, mở đầu bởi EMBL (the European Molecular Biology Laboratory) và sau đó không lâu kết hợp với GenBank, dẫn đến một thời kỳ tiếp theo trong lịch sử của dữ liệu trình tự: sự bùng nổ thực sự của số lượng dữ liệu trình tự nucleotide đã trở nên sẵn sàng cho các nhà nghiên cứu.



Hình 2.2. Dữ liệu trình tự theo cách cũ

2.2.2 Một số cơ sở dữ liệu trên thế giới

2.2.2.1. NCBI (National Center for Biotechnology Information)

NCBI là trung tâm thông tin quốc gia về công nghệ sinh học thuộc viện sức khỏe quốc gia của Hoa Kỳ (NIH). NCBI chính thức được thành lập vào ngày 4 tháng 11 năm 1988. Đến năm 1991, NCBI đảm nhiệm việc quản lý cơ sở dữ liệu trình tự DNA và từ đó NCBI còn được gọi là GenBank.

NCBI là nơi cung cấp, trao đổi thông tin về sinh học phân tử của Mỹ, thông qua những cơ sở dữ liệu trực tuyến. Ngoài ra, NCBI còn tham gia những nghiên cứu về sinh học tính toán (computational biology), phát triển những công cụ phân tích dữ liệu bộ gen, protein…

2.2.2.2 EBI (European Bioinformatics Institute)

EBI là viện tin sinh học của cộng đồng chung Châu Âu. EBI đặt tại Wellcome Trust Genome Campus nước Anh, thành lập năm 1992. EBI bắt nguồn từ EMBL (European Molecular Biology Laboratory). EMBL được thành

lập năm 1980 tại phòng thí nghiệm sinh học phân tử Heidelberg của Đức và đây là cơ sở dữ liệu trình tự nucleotide đầu tiên trên thế giới.

EBI phục vụ cho việc nghiên cứu trong các lĩnh vực như sinh học phân tử, di truyền, y học, nông nghiệp… bằng cách xây dựng, duy trì những cơ sở dữ liệu chia sẻ trực tuyến thông tin cần thiết. Bên cạnh đó, EBI còn thực hiện những nghiên cứu trong lĩnh vực tin sinh học và sinh học phân tử tính toán.

2.2.2.3. DDBJ (DNA Data Bank Japan) và PDBj (Protein Database Japan)

DDBJ là cơ sở dữ liệu về trình tự DNA của Nhật Bản, chính thức đi vào hoạt động năm 1986, đặt tại viện di truyền quốc gia (NIG). Đến năm 2001, trung tâm thông tin về sinh học ở NIG được tổ chức lại với cái tên là CIB (Center Information Biology) kết hợp với DDBJ, viết tắt là CIB/DDBJ.

PDBj là cơ sở dữ liệu của Nhật Bản, tích trữ dữ liệu về cấu trúc, chức năng protein.

- DDBJ của Nhật Bản, EMBL của Châu Âu, NCBI của Hoa Kỳ là ba cơ sở dữ liệu về trình tự nucleotide lớn, mang tính toàn cầu và ba cơ sở dữ liệu này có hợp tác, trao đổi qua lại dữ liệu. Từ đó, càng làm cho dữ liệu về trình tự nucleotide trở nên phong phú hơn.

2.3. Ngôn ngữ lập trình Perl (Practical Extraction Reporting Language)

2.3.1. Giới thiệu về Perl và lịch sử phát triển

Vào ngày 18 tháng 10 năm 1987, Larry Wall – tác giả của ngôn ngữ này, lần đầu tiên đưa Perl (Perl 1.0) vào sử dụng. Ngôn ngữ này phát sinh từ ngôn ngữ lập trình C và bị ảnh hưởng bởi các ngôn ngữ khác như BASIC, awk, sed và UNIX shell. Perl là sự kết hợp các ưu điểm của những ngôn ngữ trên.

Sau Perl 1.0 là Perl 2.0 được giới thiệu vào ngày 5 tháng 6 năm 1988. Đến thời điểm này số lượng người lập trình với những mục đích khác nhau sử dụng Perl đã tăng lên rất nhiều.

Một năm rưỡi sau, ngày 18 tháng 10 năm 1989, Perl 3.0 ra đời. Hàng ngàn người sử dụng Perl và Web (lúc này chỉ mới phát triển) đã làm cho nó thực sự nổi tiếng.

Tháng 3 năm 1991 Perl 4.0 xuất hiện. Đến lúc này Perl đã là một ngôn ngữ tương đối hoàn chỉnh mặc dù vẫn còn một số khuyết điểm.

Tháng 10 năm 1994 Perl 5 ra đời. Phiên bản này có nhiều cải tiến và đưa ngôn ngữ này lên một cấp độ mới. Perl 5 là phiên bản đầu tiên làm cho ngôn ngữ lập trình này vượt xa hơn những công việc quản trị đơn giản và trở nên phổ biến hơn. Trình diễn dịch được viết lại hoàn toàn để gia tăng tốc độ, tính hiệu quả và chức năng.

Perl 5.6 xuất hiện vào tháng 3 năm 2000, bổ sung nhiều đặc tính cho việc lập trình.

Năm 2002, phiên bản Perl 5.8 ra đời cùng với nhiều cải tiến mới được bổ sung.

Hiện nay phiên bản Perl mới nhất được Larry Wall công bố là Perl 6.0.

Perl có thể cài được trên các hệ điều hành khác nhau. Mỗi hệ điều hành khác nhau sẽ có phiên bản Perl khác nhau. Trên hệ điều hành Windows ta dùng phiên bản ActivePerl 5.6 (hay 5.8) cho Win.

Để soạn thảo ngôn ngữ Perl, ta có thể dùng các phần mềm soạn thảo như: UltraEdit, Notepad, EditPlus, Perl Builder …

Để chạy chương trình Perl, ta dùng các dòng lệnh trên MS-DOS.

2.3.2. Ứng dụng

Perl được dùng để xử lý file, truy cập dữ liệu, và được dùng cho giao diện cổng chung (Common Gateway interface – CGI), tiến trình tạo script (chương trình) của Microsoft Windows, giao diện người dùng đồ họa (Graphical User interfaces – GUI).

2.3.3. Perl và tin sinh học

Ngày nay, việc sử dụng Perl trong sinh học đã trở thành sự thực hành tiêu chuẩn. Perl còn là ngôn ngữ phổ biến nhất giữa các nhà sinh học cho vô số các công việc lập trình. Perl cũng là ngôn ngữ chung của lập trình trong sinh học hay của tin sinh học.

Một trong những lý do tại sao Perl trở nên rất thích hợp để giải quyết các vấn đề như dữ liệu trình tự DNA và protein là vì Perl rất dễ khai báo và sử dụng chuỗi. Bạn chỉ cần sử dụng nó, không cần lo lắng về việc định vị bộ nhớ, hay quản lý bộ nhớ khi

chuỗi gia tăng hay giảm xuống. DNA và protein cũng như các dữ liệu sinh học khác hầu hết luôn hiện diện trong Perl dưới dạng các chuỗi, vì vậy điều kiện thuận lợi cho các chuỗi thì cũng thuận lợi cho DNA và protein.

2.3.4. Các thành phần cơ bản trong Perl

2.3.4.1. Dữ liệu vô hướng

a) Dữ liệu vô hướng (scalar data) là một kiểu dữ liệu duy nhất bao gồm số và chuỗi.

b) Kiểu số

Ví dụ: 1, 109, 1.5e5….

c) Kiểu chuỗi

Ví dụ: ‘Đây là chuỗi trình tự DNA’, hay ta có thể viết “Đây là chuỗi trình tự DNA”. Chuỗi có thể đặt trong dấu ‘’ hay “”.

d) Biến vô hướng

- Biến vô hướng dùng để lưu giá trị dữ liệu vô hướng trong quá trình tính toán, thực hiện chương trình.

- Biến vô hướng phải bắt đầu tên biến với ký tự “$”.

- Sau ký tự “$” phải có ít nhất một mẫu tự, và mẫu tự bắt đầu không được là ký tự số.

- Tên biến có sự phân biệt giữa chữ hoa và chữ thường.

e) Các toán tử

- Toán tử tính toán cơ bản

Toán tử	Ý nghĩa	Ví dụ
=	Gán	$DNA = ‘actggtaccatg’
+	Cộng	2+3
-	Trừ	8-5
*	Nhân	4*5
/	Chia	10/5
**	Lũy thừa	2**5

- Toán tử gán nhị phân

Toán tử	Ví dụ	Ý nghĩa
+=	$x += 5	$x = $x + 5
-=	$x -= 5	$x = $x - 5
*=	$x *= 5	$x = $x * 5
/=	$x /= 5	$x = $x / 5

- Toán tử tăng giảm tự động

Toán tử	Ví dụ	Ý nghĩa
++	$x++	Biến $x tự tăng một đơn vị
--	$x--	Biến $x tự giảm một đơn vị

- Các toán tử so sánh: kết quả trả về là true hay false

Áp dụng đối với số	Áp dụng đối với chuỗi	Ý nghĩa
<	lt	Nhỏ hơn
>	gt	Lớn hơn
==	eq	Bằng
<=	le	Nhỏ hơn hoặc bằng
>=	ge	Lớn hơn hoặc bằng
!=	ne	Không bằng

- Các toán tử luận lý

Toán tử	Cách dùng tương đương
&&	and
||	or
^	xor
!	not

- Một số toán tử thông dụng khác

Toán tử	Chức năng
<STDIN> hoặc <>	Nhập input từ bàn phím
chomp	Cắt bỏ ký tự newline ở cuối chuỗi
chop	Cắt bỏ ký tự bất kỳ ở cuối chuỗi
length	Tính chiều dài của chuỗi

2.3.4.2. Các cấu trúc điều khiển

a. Câu lệnh điều kiện

- If

```
If (biểu thức) {
Khối lệnh cần thực hiện;
}
```

* Nếu biểu thức là đúng thì khối lệnh được thực hiện, nếu không khối lệnh được bỏ qua.

- If – else

```
If (biểu thức) {
Khối lệnh 1 cần thực hiện;
} else {
Khối lệnh 2 cần thực hiện;
}
```

* Nếu biểu thức là đúng thì khối lệnh 1 được thực hiện, nếu không khối lệnh hai được thực hiện.

- If – elsif - else

```
If (biểu thức 1) {
Khối lệnh 1 cần thực hiện;
} elsif (biểu thức 2) {
Khối lệnh 2 cần thực hiện;
}.....
} else {
Khối lệnh cần thực hiện;
}
```

* Nếu biểu thức 1 là đúng thì khối lệnh 1 được thực hiện, nếu không sẽ kiểm tra biểu thức 2. Nếu biểu thức 2 đúng thì khối lệnh 2 được

thực hiện…Nếu không biểu thức nào được thỏa mãn, khối lệnh trong biểu thức else được thực hiện.

- Unless

```
unless (biểu thức) {
Khối lệnh cần thực hiện;
}
```

* Nếu biểu thức sai thì khối lệnh sẽ được thực hiện.

- Unless - else

```
unless (biểu thức) {
Khối lệnh 1 cần thực hiện;
} else {
Khối lệnh 2 cần thực hiện;
}
```

* Nếu biểu thức là sai thì khối lệnh thứ 1 sẽ được thực hiện, nếu không thì khối lệnh 2 được thực hiện.

b. Vòng lặp “while”

```
while (biểu thức) {
Khối lệnh cần thực hiện;
}
```

* Đầu tiên, biểu thức sẽ được kiểm tra. Nếu biểu thức là đúng thì khối lệnh sẽ được thực hiện. Việc thực hiện khối lệnh sẽ được lặp đi lặp lại và sẽ dừng lại khi biểu thức sai. Khối lệnh có thể sẽ không thực hiện lần nào nêu biểu thức sai ngay từ đầu.

c. Vòng lặp “for”

* Vòng lặp for thường dùng để xác định số lần mà khối lệnh muốn thực hiện

```
for (biểu thức 1; biểu thức điều kiện; biểu thức 2){
khối lệnh cần thực hiện;
}
```

* Vòng lặp sẽ dừng lại khi “biểu thức điều kiện” là sai.

2.3.4.3. Mảng (array)

a. Giới thiệu

Biến mảng giống như biến vô hướng, nó được tạo ra để lưu dữ liệu. Tuy nhiên dữ liệu là một danh sách (list) (danh sách là một nhóm dữ liệu vô hướng được sắp xếp theo thứ tự).

Mở đầu biến mảng là ký tự "@", và các quy tắc đặt tên cho biến mảng cũng tương tự như đặt tên cho biến vô hướng.

Ví dụ:

```
@a;
@a = (1, 2, 3, $x, $y);
```

Các phần tử của mảng được đánh số từ 0, như mảng trên 1 ở vị trí 0, 2 là vị trí 1…

Truy cập đến một phần tử trong mảng: `$a[0]` truy cập đến phần tử thứ 0, `$a[1]` truy cập đến phần tử thứ 1 của mảng.

Nhập phần tử vào mảng từ bàn phím: `@array = <STDIN>;`

b. Một số hàm thao tác trên mảng

- Tìm chiều dài mảng:

```
$chieudai = scalar (@a);
```

Hoặc `$chieudai = ($#a +1);`

- Tìm chỉ số phần tử cuối cùng của mảng:

```
$chisophantucuoi = $#a;
```

- Hàm sort, sắp xếp thứ tự:

```
@b = sort (@a);
```

* Hàm này sắp xếp thứ tự các phần tử trong mảng theo thứ tự bảng mã ASCII, không sắp xếp theo thứ tự số.

- Hàm push, thêm phần tử mới vào mảng:

```
push (@a, $new_element);
```

* Phần tử mới được thêm vào vị trí cuối cùng.

- Hàm pop, lấy đi phần tử cuối cùng:

```
$x = pop (@a);
```

* Sau dòng lệnh này, mảng `@a` sẽ mất đi phần tử cuối cùng sẽ được gán vào biến `$x`.

- Hàm unshift, thêm phần tử mới vào đầu mảng:

```
unshif (@a, 'new_element');
```

Phần tử new_element được thêm vào đầu mảng.

- Hàm shift, lấy đi phần tử đầu tiên của mảng

```
$x = shift (@a);
```

Sau dòng lệnh này, mảng @a sẽ mất đi phần tử đầu tiên của mảng và phần tử này được gán tới biến $x.

- Hàm reverse, đảo ngược các phần tử trong mảng:

```
@b = reverse (@a);
```

- Hàm join, nối các phần tử trong mảng thành một chuỗi:

```
$string = join ("separator", @a);
```

Separator là ký tự hay chuỗi ký tự phân cách giữa hai phần tử mảng.

- Hàm split, tách một chuỗi thành một bảng các phần tử:

```
@a = split ("separator", $string);
```

c. Mảng con

Mảng con chỉ chứa một số phần tử trong mảng cho trước

```
@a = (a, b, c, d, e, f);
```

`@b = @a[1..3];` mảng @b chứa các phần tử thứ 1, 2, 3 trong mảng @a, cụ thể là các phần tử b, c, d.

`@c = @a[1,4,5];` mảng @c chứa các phần tử thứ 1, 4, 5 trong mảng @a, cụ thể là các phần tử b, e, f.

d. Vòng lặp dành cho mảng

Vòng lặp foreach được áp dụng cho mảng.

```
foreach $a (@array) {
khối lệnh cần thực hiện;
}
```

* Các phần tử trong mảng lần lượt được gán cho biến $a qua mỗi vòng lặp. Biến $a chỉ có hiệu lực cục bộ trong vòng lặp foreach.

2.3.4.4 Bảng băm (Hash)

a. Giới thiệu

Hash là một loại biến dùng để lưu trữ danh sách dữ liệu vô hướng tương tự như mảng. Tuy nhiên, các phần tử trong mảng được chỉ mục (index) tự động còn trong Hash thì không được tạo chỉ mục một cách tự động. Các phần tử trong Hash đi thành từng cặp key/ value, trong đó phần tử key dùng làm chỉ mục cho phần tử value.

Mở đầu biến hash là ký tự "%" và qui tắc đặt tên cho hash tương tự như mảng.

Có hai cách khai báo:

- `%hash = (key1, value1, key2, value2, key3, value3);`
- ```
  %hash = ( key1 => value 1,
        key2 => value 2,
        key3 => value 3);
  ```

Truy cập một phần tử của hash:

```
$a = $hash {$key};
```

Thêm phần tử mới vào hash:

`$hash{$key} = $value`; cặp giá trị key/ value được thêm vào hash.

b. Một số hàm thao tác trên hash

- Hàm delete, xóa phần tử trong hash

  `delete $hash{$key}`; xóa cặp giá trị key/value tương ứng với nhau.

- Hàm keys, trích các keys và lưu các keys này vào mảng:

  ```
  @keys = keys (%hash);
  ```

- Hàm values, trích các values và lưu các values này vào mảng

  ```
  @values = values (%hash);
  ```

2.3.4.5. Thao tác với tập tin

a. Mở tập tin

Cú pháp
  ```

```
Open (Filehandle, "đường dẫn đến tập tin cần mở") or
die ("Không mở được tập tin");
```

FileHandle sẽ là tham chiếu đến tập tin cần mở suốt chương trình. Nếu không mở được tập tin với lý do nào đó, hàm die được thực thi và chương trình bị ngắt.

Khi mở một tập tin, chúng ta có thể mở ở ba chế độ khác nhau: đọc (read), viết (write), chèn (append). Một tập tin được mở thì mặc định trong chế độ đọc.

Mở tập tin trong chế độ viết (write), ta thêm dấu ">"trước đường dẫn. Chú ý khi mở tập tin trong chế độ Write thì nội dung của toàn bộ tập tin sẽ bị xóa và nội dung mới sẽ được ghi thêm vào, nếu không được thêm vào tập tin sẽ là rỗng.

Mở tập tin trong chế độ chèn (append) ta thêm dấu ">>" vào trước đường dẫn. Khi mở tập tin trong chế độ này ta có thể thêm nội dung vào tập tin.

Mở tập tin để đọc và thêm nội dung vào (read/write) ta thêm dấu "+<" vào trước đường dẫn.

Tạo một tập tin mới có thể đọc và viết vào ta thêm dấu +> vào trước đường dẫn.

b. Đóng tập tin

Cú pháp

```
close (FileHandle);
```

c. Đọc tập tin

Sau lệnh mở tập tin, nội dung của tập tin có thể được đọc như sau:

```
Open (THU, "D:/Perl/thu.txt") or die ("Không mở được
tập tin");
$thu = <THU>;
print "dòng đầu tiên của tập tin là: $thu";
```

Nếu tập tin thu.txt có nhiều dòng, mỗi dòng trong tập tin thu.txt tương ứng với một phần tử trong mảng. Do đó khi gán `$thu = <THU>`, `$thu` chỉ chứa dòng đầu tiên của tập tin. Để in hết nội dung của tập tin thu.txt, ta phải dùng vòng lặp

```
open (THU, "D:/Perl/thu.txt") or die ("Không mở được
tập tin");
$thu = <THU>;
while ($thu) {
print "$thu \n";
$thu = <THU>;
}
exit;
```

Ngoài ra ta có thể dùng mảng chứa nội dung tập tin, trong đó mỗi dòng trong tập tin ứng với mỗi phần tử trong mảng. Ta thực hiện như sau

```
open (THU, "D:/Perl/thu.txt") or die ("Không mở được
tập tin");
@thu = <THU>;
print "@thu";
exit;
```

d. Viết nội dung vào tập tin

Cú pháp

```
print FileHandle "nội dung cần ghi vào";
```

Có thể viết nội dung cho tập tin từ bàn phím:

```
$thu = <STDIN>;
print FileHandle "$text";
```

2.3.4.6. Chương trình con

a. Giới thiệu

Chương trình con là các đoạn mã thể hiện các chức năng khác nhau trong chương trình chính. Khi viết các chương trình con chúng ta có thể tái sử dụng thay vì viết lại tất cả. Và việc dùng chương trình con làm cho việc tổ chức chương trình tốt hơn, làm cho chương trình dễ đọc và dễ kiểm soát hơn.

Khai báo:

```
Sub TenChuongTrinhCon {
Đoạn mã cần thực hiện;
}
```

b. Sử dụng chương trình con

Gọi chương trình con

```
&TenChuongTrinhCon ( );
```

Ta có thể bỏ đi dấu "&".

c. Ví dụ cách dùng chương trình con

```
# !/usr/bin/perl –w
print "Nhap vao trinh tu DNA thứ 1: " ;
my $dna1 = <STDIN> ;
Chomp $dna1 ;
Print "Nhap vao trinh tu DNA thứ 2: ";
my $dna2 = <STDIN>;
chomp $dna2;
my $dna3 = &noiDNA ($dna1, $dna2);
print "Đây là chuỗi DNA nối: $dna3 \n";
exit;
##################################

sub noiDNA {
my ($dna1,$dna2) = @_;
my $dna3 = $dna1. $dna2;
return $dna3;
}
```

* Đầu tiên chương trình nhận vào hai trình tự DNA nhập từ bàn phím và lưu chúng lần lượt vào hai biến vô hướng `$dna1, $dna2`. Biến được khai báo với `my` qui định phạm vi hoạt động của biến và đảm bảo không có hiện tượng trùng tên biến xảy ra. Hai biến `$dna1` và `$dna2` được xem như tham số và được truyền vào chương trình con để xử lý. Lúc này mọi hoạt động sẽ diễn ra trong chương trình con. Chương trình con nhận vào hai biến `$dna1, $dna2` thông qua biến đặc biệt `@_` và gán cho hai biến `$dna1` và `$dna2` trong chương trình con. Chương trình con thực hiện nối nội dung hai biến lại, gán cho biến `$dna3` cuối cùng trả giá trị lại cho chương trình chính qua chức năng `return`. Biến `$dna3` trong chương trình chính sẽ nhận giá trị trả về này, sau đó được xuất ra màn hình bởi dòng lệnh print.

2.3.4.7. Regular Expression

a. Giới thiệu

Regular expression là một đặc tả cho một nhóm ký tự ta muốn tìm trong một chuỗi.

Pattern là một chuỗi ký tự nhất định mà ta có thể tìm kiếm trong một chuỗi.

Vậy, regular expression sẽ đặc tả một pattern và patern này sẽ là khuôn mẫu có thể so khớp với chuỗi ký tự đã cho.

b. Ví dụ cách dùng regular expression

Ta viết chương trình tìm đoạn nhỏ DNA trong một chuỗi trình tự DNA cho trước

```
#!/usr/bin/perl –w
my $dna = 'ACTGTGATGCGTACGTTTAC';
my $subdna = 'ATGC';
if ($dna =~ /$subdna/){
print "Tìm thấy $subdna trong chuỗi DNA $dna \n";
}else {
print "Không tìm thấy $subdna trong chuỗi DNA $dna
\n";
}
exit;
```

Trong chương trình này, ta dùng regular expression ở dòng lệnh `$dna =~ /$subdna/`. Mục đích chương trình là kiểm tra 'ATGC' có trong chuỗi DNA ban đầu hay không. Pattern ở đây là `$subdna`, mang nội dung là đoạn trình tự gắn 'ACTG' được thể hiện thành regular expression khi đặt pattern này vào giữa hai dấu "//". Regular expression này sẽ tìm `$subdna` trong chuỗi DNA ban đầu thông qua toán tử kết nối =~ (binding operator). Kết quả tìm kiếm sẽ trả lại giá trị true, nếu tìm thấy và false nếu không tìm thấy.

2.4. Giới thiệu về cây xoài

2.4.1. Vị trí phân loại

Cây xoài (2n=40) thuộc

Giới	Plantaeia
Ngành	Magnoliophyta
Lớp	Magnoliopsida
Phân lớp	Rosidae
Bộ	Sapindales
Họ	Anacardiaceae
Giống	Mangifera
Loài	Mangifera indica L.

2.4.2. Nguồn gốc

Cây xoài là một trong những cây ăn quả được người Ấn Độ trồng từ rất lâu đời. Theo De Candolle (1886) người Ấn Độ đã biết trồng xoài cách đây khoảng 4000 năm, còn theo Hill (1952) thì khoảng 6000 năm, vì vậy rất khó xác định rõ nguồn gốc phát sinh của cây xoài.

Dựa vào sự xuất hiện của các loài hoang dại, các bằng chứng khảo cổ học, sự phân bố địa lý cũng như lịch sử trồng trọt lâu đời đã gắn liền với các phong tục tập quán của người dân trong vùng, nhiều nhà nghiên cứu (De Candolle, 1904; Popenoe, 1920; Vavilov, 1949-1950; Mukherjee, 1951; và Singh, 1959) cho rằng cây xoài có nguồn gốc ở vùng biên giới giữa Ấn Độ và Myanma. Theo Bondad (1989) có ba vùng có thể được coi là nơi phát sinh của cây xoài, đó là khu vực Ấn Độ và Đông Dương, vùng biên giới giữa Ấn Độ và Myanma, khu vực Đông Nam Á.

Do đó, cây xoài là cây của vùng nhiệt đới và có nguồn gốc từ các nước trong khu vực Châu Á mà trong đó Ấn Độ và các nước trong khu vực Đông Nam Á được coi là trung tâm phát sinh của cây xoài.

2.4.3. Giá trị dinh dưỡng và lợi ích

Khi phân tích thịt quả xoài có

Bảng 2.1. Giá trị dinh dưỡng của quả xoài

Đơn vị	**Năng lượng (calo)**	**Carbohydrate (gram)**	**Protein (gram)**	**Cholesterol (milligram)**	**Trọng lượng (gram)**	**Chất béo (gram)**	**Chất béo bão hòa (gram)**
1 quả	135	35	1	0	207	1	0.1

Tỷ lệ phần ăn được của quả xoài là 70%. Xoài giàu vitamin A, B2, và C đặc biệt là vitamin A, trong 100g ăn được có đến 4,8mg. Ngoài ra còn có các loại muối khoáng K, Ca, P, Cl.

Quả xoài ngoài ăn tươi còn dùng làm đồ hộp, làm mứt, nước giải khát, cho lên men rượu, làm dấm. Vỏ quả chữa kiết lị, hoại huyết. Vỏ cây xoài già chữa sốt, đau răng. Lá chữa ho, sưng họng. Rễ cây xoài cũng có thể nhuộm vải. Hoa xoài là nguồn mật cho ong.

2.4.4. Đặc điểm hình thái

2.4.4.1. Rễ

Cây xoài có bộ rễ rất sâu và khỏe, nhất là hệ thống rễ cọc. Rễ có thể mọc sâu 5-6m nhưng phần lớn phân bố tập trung ở tầng đất 0-50cm. Về bề rộng, rễ có thể ăn xa 9m, nhưng tập trung ở vùng bán kính 2m. Nhờ có bộ rễ ăn sâu và phân bố rộng mà cây xoài được coi là cây có khả năng chịu hạn rất tốt.

2.4.4.2. Thân và tán cây

Cây xoài thuộc loại đại mộc, sinh trưởng khỏe nên cây to và tán lớn, xanh quanh năm. Thân gỗ cao 10-15m với độ lớn tán tương tự. Tán có hình bầu dục, hình tháp hoặc hình cầu tùy theo giống.

2.4.4.3. Lá

Lá đơn, mọc vòng, có kích thước lớn: rộng 6-10cm, dài 35cm. Mỗi năm cây ra 3-4 đợt lộc, lá non ra trên các chồi mới, mọc theo chùm, mỗi chùm có 7-12 lá. Lá non có màu tím hồng hoặc phớt nâu. Lá già có màu xanh đậm. Lá non đạt kích thước ổn định hai tuần sau khi mọc và lá chuyển lục hoàn toàn sau 35 ngày. Thời gian tồn tại của lá xoài là ba năm.

2.4.4.4. Hoa

Hoa mọc thành chùm ở ngọn cành. Chùm hoa to và dài 20-40cm. Mỗi chùm có 200-4000 hoa. Trên một chùm hoa thường có cả hai loại hoa: hoa lưỡng tính và hoa đực. Hoa có kích thước nhỏ 6-8mm. Hoa lưỡng tính có tiểu

nhụy hữu thụ, có vòi nhụy, có bầu noãn phát triển. Hoa đực thì tiểu nhụy bất thụ và có bao phấn phát triển.

Hình 2.3. Hoa xoài

2.4.4.5. Quả

Quả xoài có thịt quả, vỏ quả và hạt. Hình dạng, độ lớn và màu sắc của quả có thể nhận biết tùy theo giống. Thời gian từ khi ra hoa đến khi quả chín tùy giống, giống chín sớm thì 2 tháng, giống chính vụ thì 3-3,5 tháng, giống chín muộn thì 4 tháng. Xoài Việt Nam thuộc nhóm chính vụ.

Hình 2.4. Quả xoài

2.4.4.6. Hạt

Cấu tạo hạt xoài bao gồm

- Gân là các sọc dọc theo chiều dài hạt
- Xơ có ở khắp hạt, dài nhất ở bụng và lưng của hạt

- Lớp vỏ cứng (nội quả bì) dày màu nâu
- Lớp vỏ màu vàng trong suốt nằm sát lớp vỏ cứng
- Lớp vỏ bao màu nâu mềm bao quanh là mầm nối liền với cuống bằng một sợi nhỏ
- Lá mầm có nhiệm vụ cung cấp dinh dưỡng cho cây con như phôi nhũ của các hạt khác
- Phôi

2.4.4.7. Phôi

Xoài có nguồn gốc từ các nước Đông Dương, Malaysia, Indonesia, Philipine thường thuộc nhóm đa phôi, còn xoài ở Ấn Độ, Banglades, Pakistan có hiện tượng đơn phôi nhiều hơn. Xoài đa phôi là trong 1 hạt có nhiều phôi và khi gieo hạt đó có thể mọc lên nhiều cây con. Trong các phôi đó có 1 phôi hữu tính, còn lại là phôi vô tính do các tế bào của phôi tâm hình thành. Cây mọc từ phôi vô tính thì giống cây mẹ, còn cây mọc từ phôi hữu tính thì cây mẹ. Ở các giống đơn phôi, cây mọc khác cây mẹ vì đó là phôi hữu tính.

2.4.5. Yêu cầu sinh thái

2.4.5.1. Nhiệt độ

Nhiệt độ thấp nhất là 2-4 ^{o}C, thích hợp nhất là 24-26 ^{o}C, nhiệt độ cao nhất xoài chịu được là 44-45 ^{o}C nhưng ở nhiệt độ này yêu cầu đủ nước. Nhiệt độ ảnh hưởng rõ rệt đến sinh trưởng và thời gian chín của quả xoài. Đối với sinh trưởng quả, cây xoài cần nhiệt độ cao hơn so với thời gian ra hoa và nhiệt độ cao trong thời gian quả phát triển là yếu tố quan trọng để có thể thu hoạch xoài sớm.

2.4.5.2. Đất

Xoài không kén đất, thích hợp trồng trên nhiều loại đất, đất vàng, đỏ, Ferelit, phù sa cổ, phù sa mới ven sông… nhưng phải có tầng canh tác dày ít nhất là 1,5-2m. Độ pH tốt nhất là 5,5-6,5. Mực nước ngầm thích hợp là 2,5m, nếu mực nước nước ngầm không ổn định thì ảnh hưởng xấu đến bộ rễ.

2.4.5.3. Lượng mưa

Xoài có thể sinh trưởng, phát triển tốt mà không cần tưới ở những vùng có lượng mưa trung bình năm từ 1200-1500mm. Trong 1 năm cây xoài cần phải có một khoảng thời gian khô hạn vào thời điểm cuối năm để tạo điều kiện cho quá trình phân hóa mầm hoa. Trong thời gian xoài nở hoa yêu cầu thời tiết khô ráo để tạo thuận lợi cho quá trình thụ phấn hình thành quả.

2.4.6. Một số giống xoài trồng phổ biến ở Việt Nam

2.4.6.1. Xoài cát Hòa Lộc

Xuất xứ từ Cái Bè (Tiền Giang) và Cái Mơn (Bến Tre), được người nông dân ở nhiều tỉnh vùng đồng bằng sông Cửu Long tuyển chọn, nhân giống và trồng qua nhiều thế hệ do có phẩm chất tốt. Trái to trọng lượng trung bình 300-500g. Hình dáng quả bầu dài, vỏ mỏng, hột nhỏ. Thịt vàng, cơm dày, dẻ, không có xơ, hương vị thơm ngon và ngọt. Thời gian từ khi ra hoa đến chín là 3,5 tháng. Giống quý, nhưng hơi khó vận chuyển và xuất khẩu do có vỏ mỏng nên dễ bị dập nếu chuyên chở không cẩn thận.

2.4.6.2. Xoài cát Cần Thơ

Quả nhỏ hơn xoài cát Hòa Lộc, có cơm dày, ngọt, hương vị thơm ngon và cho năng suất khá cao. Thời gian từ khi ra hoa đến chín là 3,5 tháng.

2.4.6.3. Xoài thơm

Xoài này được trồng nhiều ở Tiền Giang, Đồng Tháp, Cần Thơ. Trọng lượng trái trung bình 250-300g, vỏ trái xanh sậm (thơm đen) hay xanh nhạt (thơm trắng), thịt quả ngọt thơm. Thời gian từ khi trổ đến khi chín khá sớm 2,5 tháng.

2.4.6.4. Xoài bưởi (xoài ghép)

Cây trồng bằng hạt chỉ 2-3,5 năm là có quả. Trọng lượng trái trung bình là 250-350g. Vỏ dày, thịt nhão, ít ngọt. Mùi hôi của trái giảm dần khi tuổi cây càng già.

2.4.6.5. Xoài tượng

Trọng lượng trung bình của trái là 700-800g. Thịt quả màu vàng nhạt, ít xơ, ít nước, không ngọt, hơi chua thường ăn sượng.

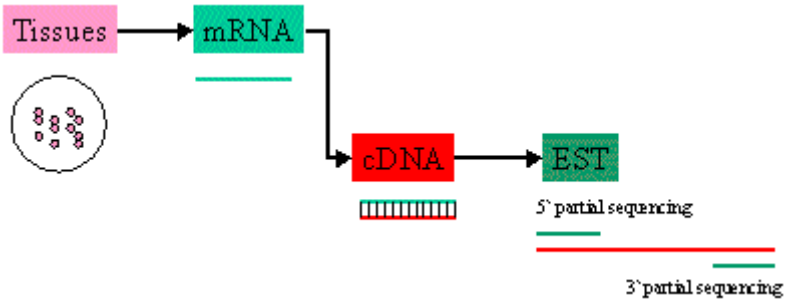
2.4.6.6. Xoài Thanh Ca

Xoài này được trồng phổ biến ở các tỉnh duyên hải miền Trung, 1 phần ở Thành phố Hồ Chí Minh, 1 số tỉnh miền Đông Nam Bộ và đồng bằng sông Cửu Long. Trọng lượng trung bình 350-580g, quả hình trứng dài, vỏ vàng tươi và bóng, thịt vàng tươi, ít xơ, nhiều nước, ngọt và thơm. Cây có nhiều đợt quả trái vụ trong năm.

2.5. Khái quát về EST (Expressed Sequence Tag)

2.5.1. Định nghĩa

EST là những đoạn nhỏ trong trình tự DNA (thường dài từ 200 đến 500 nucleotide) được tạo ra bằng cách giải trình tự một đầu hay cả hai đầu của một gen biểu hiện.



Hình 2.5. Sơ đồ hình thành EST

2.5.2. Nguyên nhân hình thành và ứng dụng của EST

Các nhà nghiên cứu đang lao động một cách cần mẫn để giải trình tự và thu thập bộ gen của rất nhiều loại sinh vật, bao gồm chuột và người, với một số lượng lớn vì những lý do quan trọng.

Mặc dù những mục tiêu quan trọng của bất kì dự án giải trình tự nào đều có thể có được trình tự gen và xác định được một tập hợp hoàn chỉnh của gen, nhưng mục tiêu cuối cùng là đạt đến tầm hiểu biết về việc khi nào, vị trí nào, và bằng cách nào mà một gen được hoạt hóa, một tiến trình mà thường được xem là sự biểu hiện gen.

Một khi chúng ta bắt đầu tìm hiểu vị trí nào và bằng cách nào 1 gen được biểu hiện dưới những điều kiện thông thường, sau đó chúng ta có thể nghiên cứu điều gì xảy ra trong một trạng thái đã thay đổi, ví dụ như trường hợp bị nhiễm bệnh. Tuy nhiên, để thực hiện được mục tiêu sau cùng, các nhà nghiên cứu phải xác định và nghiên cứu về protein, hay những protein mà nó được mã hóa bởi một gen nào đó.

Việc tìm thấy một gen mà mã hóa cho một protein hoặc nhiều protein là điều không dễ dàng. Như trước đây, các nhà nghiên cứu sẽ bắt đầu cuộc tìm kiếm bằng cách định rõ một vấn đề sinh học và phát triển thành một chiến lược cho việc nghiên cứu vấn đề đó. Thông thường, việc tìm những tài liệu khoa học thường cung cấp nhiều dẫn chứng cho việc tiến hành như thế nào. Ví dụ, các phòng thí nghiệm khác có thể công bố dữ liệu mà đã thiết lập sự liên kết giữa một protein đặc biệt và một căn bệnh được quan tâm. Các nhà nghiên cứu sau đó sẽ làm việc để phân lập protein, xác định chức năng của nó, và định vị gen mà nó mã hóa cho protein.

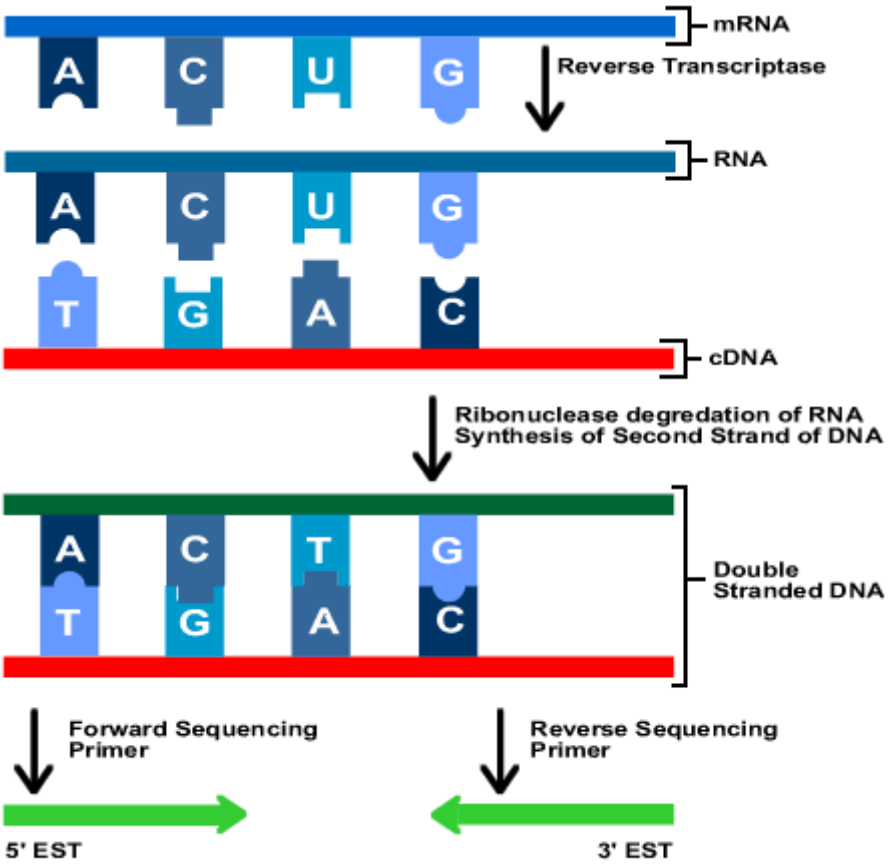
Một cách khác, các nhà khoa học sẽ tiến hành những nghiên cứu di truyền để xác định vị trí nhiễm sắc thể của một gen đặc biệt. Một khi vị trí nhiễm sắc thể đã được xác định, các nhà khoa học sẽ sử dụng những phương pháp hóa sinh để phân lập gen và protein tương ứng. Dù bằng cách nào thì những phương pháp này đều tốn nhiều thời gian, có trường hợp nhiều năm, và kết quả là chỉ có vị trí và sự miêu tả của một số lượng phần trăm nhỏ của gen được tìm thấy.

Tuy nhiên, thời gian đòi hỏi cho việc định vị và mô tả hoàn toàn một gen đã giảm xuống đáng kể nhờ sự phát triển và hướng tiếp cận của một kĩ thuật được dùng để tạo ra Expressed Sequence Tag hay EST. EST cung cấp cho nhà nghiên cứu một phương pháp nhanh chóng và không tốn kém cho việc khám phá các gen mới, tìm được dữ liệu về sự biểu hiện và điều hòa gen, và cho việc thành lập bản đồ gen.

Ý tưởng là giải trình tự những mảnh DNA mà chúng đại diện cho những gen biểu hiện trong tế bào, mô hay cơ quan nào đó từ những sinh vật khác nhau và sử dụng những sự đánh dấu này để tìm ra được gen bằng sự bắt cặp giữa các nucleotide. Thách thức kết hợp việc xác định gen từ trình tự bộ gen biến đổi giữa các sinh vật và độc lập với kích thước bộ gen cũng như sự hiện diện hay vắng mặt của intron, nó là những trình tự DNA xen vao làm gián đoạn trình tự mã hóa protein của một gen.

2.5.3. Sự hình thành EST

cDNA đại diện cho 1 gen biểu hiện đã được phân lập, các nhà khoa học sau đó có thể giải trình tự vài trăm nucleotide từ đầu này hay đầu kia của phân tử để tạo ra hai loại EST khác nhau.



Hình 2.6. Sự hình thành EST

- 5’EST

Chỉ giải trình tự phần bắt đầu của cDNA tạo ra 5’EST. 5’EST có được từ đầu 5’ của một bản sao (transcript) mà bản sao này thường mã hóa cho một protein. Những vùng này có khuynh hướng bảo tồn giữa các loài và không thay đổi nhiều trong một họ gen.

- 3’EST

Giải trình tự phần cuối của phân tử cDNA tạo ra 3’EST. Bởi vì những EST này được tạo ra từ đầu 3’ của bản sao, chúng thường rơi vào những vùng không mã hóa cho protein, hay là những vùng không dịch mã (untranslated region – UTR), và do đó chúng có khả năng biểu lộ sự bảo tồn giữa các loài thấp hơn so với những trình tự mã hóa.

2.6. Giới thiệu về microsatellite

2.6.1. Khái niệm

Microsatellite là những trình tự đặc biệt của DNA mà có chứa sự lặp lại nối tiếp từ 2 đến 6 bp (Connel và ctv, 1998).

Ví dụ

GTGTGTGTGTGT hay $(GT)_6$

CTGCTGCTGCTGCTG hay $(CTG)_5$

ACTCACTCACTCACTC hay $(ACTC)_4$

Trong các tài liệu microsatellite còn được gọi là SSR (simple sequence repeats), STR (short tandem repeats), VNTR (variable number of tandem repeats).

2.6.2. Đặc điểm

Microsatellite là marker được lựa chọn trong việc lập bản đồ phân tử, sự xác định những giống cây trồng, đánh giá nguồn gốc tổ tiên của cây trồng cho mục đích nghiên cứu quần thể cây trồng và sự tiến hóa là vì

- ✓ Có tính đa alen và biến dị cao
- ✓ Là marker đồng trội
- ✓ Phân bố ngẫu nhiên khắp bộ gen sinh vật
- ✓ Dễ dàng xác định bằng PCR sử dụng các primer đặc biệt

Microsatellite có ở bộ gen thực vật thấp hơn năm lần so với động vật có vú (Lagercrantz và ctv, 1993). Ước tính tần số xuất hiện của microsatellite ở thực vật trong phạm vi từ mỗi một 3.3 kb ở lúa mạch (Becker và Heun, 1995) đến 1.2 Mb cho sự lặp lại GA/CT và GT/CA ở cà chua (Broun và Tanksley, 1996). Trung bình sự xuất hiện của microsatellite là mỗi một 21.2 kb ở thực vật hai lá mầm và mỗi một 64.6 kb ở thực vật một lá mầm (Wang et al., 1994).

Một cá thể có một locus đồng hợp sẽ có cùng số lần lặp lại trên cả hai nhiễm sắc thể, trong khi một cá thể dị hợp sẽ có số lần lặp lại khác nhau trên hai nhiễm sắc thể. Những vùng xung quanh locus của microsatellite, được gọi là vùng hai bên (flanking region) có thể có cùng trình tự. Điều này rất quan trọng bởi vì những vùng hai bên có thể được dùng như primer của phản ứng PCR khi nó sẽ khuếch đại

microsatellite, và vùng hai bên này sẽ bảo tồn giữa các giống hay thỉnh thoảng giữa các họ.

Hình dưới có hai dòng đại diện cho hai nhiễm sắc thể tương đồng trong cơ thể lưỡng bội. (Để rõ ràng, chỉ một sợi của mỗi nhiễm sắc thể được thể hiện)

Đồng hợp (cả hai sợi có 7 lần lặp lại CT)

…CGTAGCCTTGCATCCTT**CTCTCTCTCTCTCT**ATCGGTACTACGTGG…

…CGTAGCCTTGCATCCTT**CTCTCTCTCTCTCT**ATCGGTACTACGTGG

5’ vùng hai bên microsatellite 3’ vùng hai bên

Dị hợp: (một sợi có 7 lần lặp lại, và sợi kia có 8 lần lặp lại

…CGTAGCCTTGCATCCTT**CTCTCTCTCTCTCT**ATCGGTACTACGTGG…

…CGTAGCCTTGCATCCTT**CTCTCTCTCTCTCTCT**ATCGGTACTACGTGG…

2.6.3. Cơ chế hình thành microsatellite

Sự đa dạng của microsatellite là kết quả từ sự khác nhau trong số lượng các đơn vị lặp lại. Sự khác biệt này được tạo ra bởi những lỗi trong quá trình tái bản DNA (Jarne và Lagoda, 1996; Moxon và Willis, 1999); enzyme DNA polymerase bị lỗi khi nó sao chép vùng lặp lại, làm thay đổi số lần lặp lại (Jarne và Lagoda, 1996).

2.6.3.1. Sự trượt lỗi của polymerase (Polymerase slippage)

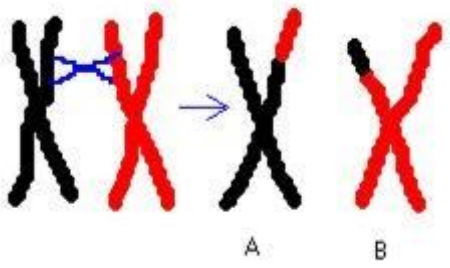
Khi DNA tái bản, enzyme polymerase không tìm thấy vị trí của nó và cắt bớt đơn vị lặp lại hay thêm vào quá nhiều đơn vị lặp lại. Kết quả là sợi mới có số lần lặp lại khác với sợi bố mẹ. Điều này giải thích cho những sự thay đổi nhỏ trong số lần lặp lại (thêm vào hoặc bớt đi một hay nhiều lần lặp lại).

Sự trượt lỗi có thể khuếch đại những trình tự lặp lại ngắn này thành nhiều lần lặp lại qua các thế hệ kế tiếp.

Bên cạnh đó, hiệu quả của hệ thống sửa chữa cho sự bắp cặp sai cũng đóng một vai trò quan trọng trong tốc độ biến đổi của microsatellite.

2.6.3.2 Sự bắt cặp không đồng đều trong giảm phân

Cơ chế này giải thích cho những thay đổi lớn hơn trong số lần lặp lại. Trong sơ đồ dưới, nhiễm sắc thể A có quá nhiều sự lặp lại, và nhiễm sắc thể B thì có quá ít sự lặp lại.



Hình 2.7. Sự bắt cặp không đồng đều trong giảm phân

2.6.4. Mô hình sự đột biến của microsatellite

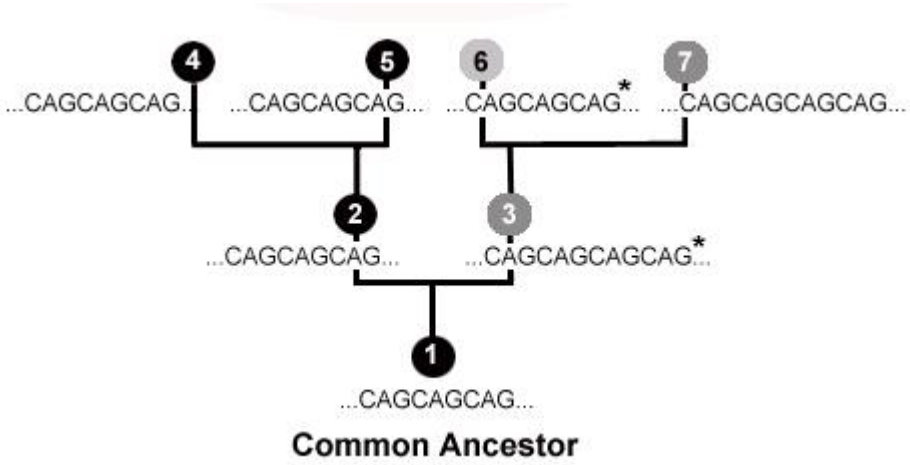
2.6.4.1 Mô hình đột biến bậc thang (SMM – Stepwise Mutation Model)

Mô hình này giữ cho các microsatellite chỉ tăng hoặc giảm một lần lặp lại. Nó gợi ý rằng hai alen khác nhau bởi 1 lần lặp lại thì có quan hệ họ hàng gần hơn (có tổ tiên chung gần hơn) so với những alen khác nhau nhiều lần lặp lại.

Nói cách khác, kích thước có ý nghĩa khi thực hiện những thí nghiệm thống kê trên quần thể. Việc sử dụng mô hình này để thống kê khoảng cách di truyền được gọi là Rst. SMM hầu như là mô hình ưu tiên khi tính toán mối quan hệ giữa các quần thể mặc dù sẽ xuất hiện vấn đề homoplasy (được giải thích ở phần sau)

Giả sử rằng bạn đang nghiên cứu một quần thể và bạn tìm thấy bốn cá thể. Ba cá thể trong số đó có cùng kiểu gen và một cá thể có sự khác biệt. điều này chỉ ra rằng ba cá thể đó có quan hệ họ hàng gần hơn so với cá thể còn lại.

Tuy nhiên đó không phải là trường hợp duy nhất. Để hiểu được tại sao, hãy xem sự phát sinh loài dưới đây. Dấu hoa thị chỉ ra sự đột biến của microsatellite.



Hình 2.8. Mô hình đột biến bậc thang

Trong hình này, quần thể 1 là nguồn gốc của hai quần thể, 2 và 3. Trong quần thể 3, có một đột biến bậc thang, nên nó có CAG lặp lại bốn lần thay vì ba lần. Quần thể 3 là nguồn gốc của hai quần thể, 6 và 7. Quần thể 6 mất đi một lần lặp lại, nên chỉ có CAG lặp lại ba lần. Vấn đề là các quần thể 4,5 và 6 còn cùng alen tại locus của microsatellite, tuy nhiên nó có lịch sử tiến hóa khác nhau. Chúng ta có thể nói rằng alen của chúng được xác định theo trạng thái nhưng không theo dòng dõi.

Nếu một nhà khoa học chỉ kiểm tra một locus này, người đó sẽ kết luận một cách nhầm lẫn rằng quần thể 6 có quan hệ họ hàng với quần thể 4 và 5 hơn so với quần thể 7.

Hiện tượng hai alen được xác định theo trạng thái nhưng không xác định bởi dòng dõi, được gọi là homoplasy. Trong nghiên cứu quần thể, homoplasy có thể dẫn đến việc đánh giá sai về sự phân hướng tiến hóa. Cách duy nhất để phát hiện homoplasy là thí nghiệm trên nhiều locus khác. Homoplasy được cho rằng có ảnh hưởng nhỏ trên quần thể trong một khoảng thời gian ngắn (hàng trăm thế hệ) và mô hình đột biến bậc thang vẫn là mô hình được ưu tiên (Goodman, 1998).

2.6.4.2. Mô hình "K" alen

Mô hình này giữ cho một microsatellite có thể đột biến một cách ngẫu nhiên thành bất cứ "K" alen nào. Do đó, nó không cho rằng một trình tự có 8

lần lặp lại nhất thiết đột biến thành trình tự có 7 hay 9 lần lặp lại. Trình tự thích hợp đột biến thành một trình tự có 15 lần lặp lại.

2.6.4.3. Mô hình alen vô hạn (infinite alleles model – IAM)

Mỗi đột biến có thể tạo ra một cách ngẫu nhiên bất cứ alen mới nào. Một alen có 15 lần lặp lại có thể có quan hệ gần với một alen có 10 lần lặp lại cũng như alen có 11 lần lặp lại. Nói cách khác, kích thước không quan trọng. Việc thống kê sử dụng mô hình này được gọi là Fst.

2.6.5. Nguyên nhân tồn tại của microsatellite

Microsatellite là DNA vô nghĩa, và sự biến đổi phần lớn không có tính chất rõ rệt.Chúng thường không có tác động có thể đo lường được trên kiểu hình, và khi chúng đột biến, thông thường là gây hại và không có lợi. Ở người, 90% những microsatellite đã biết được tìm thấy trong vùng không mã hóa của bộ gen. Khi tìm thấy ở vùng mã hóa ở người, microsatellite được biết là gây bệnh. Thú vị là khi tìm thấy trong vùng mã hóa, microsatellite thường là sự lặp lại ba nucleotide. Sự giải thích có thể là do những dạng nucleotide lặp lại khác sẽ gây hại nhiều cho vùng mã hóa, vì nó sẽ gây ra sự đột biến xê dịch khung.

Microsatellite cung cấp nguồn cần thiết cho sự đa dạng di truyền. Ở vi khuẩn, sự biến đổi alen của microsatellite trong vùng mã hóa được cho là để thích nghi với những môi trường khác nhau. Nghĩa là một alen ngắn có thể thích nghi với một môi trường, và một alen dài với nhiều lần lặp lại có thể thích nghi với một môi trường khác. Đặc biệt là, sợi nhỏ protein ngắn có thể làm cho vi khuẩn ít nhớt, và một sợi nhỏ protein dài hơn có thể làm nó dính hơn và gây bệnh hơn (Moxon và Wills, 1999). Do đó, có sự đa dạng trong quần thể sẽ đảm bảo sự sống sót của quần thể vi khuẩn trong những môi trường khác nhau. Tương tự, Kashi và Soller (1999) tin rằng sự đa dạng của microsatellite có thể là một cách để đền bù cho sự mất đi tính đa dạng di truyền do bởi sự chọn lọc di truyền.

Microsatellite có thể giúp điều hòa sự biểu hiện gen và chức năng protein. Kashi và Soller (1999) cũng đưa ra giả thuyết rằng microsatellite có thể có vai trò điều hòa trong biểu hiện gen. Chúng được tìm thấy một cách có hệ thống gần những vùng

mã hóa. Sự đa dạng của microsatellite cũng kết hợp với sự biến đổi về số lượng trong chức năng protein và hoạt động của gen.

2.6.6. Các cách phân lập

Microsatellite có thể được tìm thấy bằng nhiều phương pháp, bao gồm sự thu được từ những thư viện của bộ gen bằng cách sàng lọc thư viện của bộ gen, sàng lọc thư viện nhiễm sắc thể vi khuẩn, thư viện cDNA, từ những dữ liệu chung như ngân hàng gen (GenBank), từ các loại lân cận và từ dữ liệu sự đánh dấu trình tự biểu hiện (EST).

2.6.6.1 Microsatellite có nguồn gốc từ thư viện

Thư viện của bộ gen có thể là một nguồn của microsatellite. Để phát triển những microsatellite từ thư viện của bộ gen, những dòng thư viện được sàng lọc với các probe có đặc trưng lặp đi lặp lại. Những dòng dương tính sau đó được giải trình tự cho sự xác minh và thiết kế mồi. Những ví dụ về sự sử dụng thành công khuynh hướng này là ở lúa mì (Ma và ctv, 1996), cây thông (Kostia và ctv, 1995), cây lúa miến (Brown và ctv, 1996), nho (Bower và ctv, 1996), đậu nành (Akkaya và ctv, 1992). Trong sự so sánh với các phương pháp khác để có được microsatellite, phương pháp này có thể tốn nhiều công sức, đặc biệt là khi yêu cầu nhiều microsatellite. Trong một thí nghiệm ở cây thông, 6000 dòng được sàng lọc để có được 8 microsatellite hữu dụng (Kostia và ctv, 1995), và ở cây lúa miến chỉ có 0.2% dòng chứa microsatellite, trong đó số microsatellite hữu dụng là ít hơn (Brown và ctv, 1996). Một thuận lợi của hướng này là phương pháp có kỹ thuật đơn giản, phù hợp với tất cả các phòng thí nghiệm.

2.6.6.2 Microsatellite từ thư viện BAC/YAC

Microsatellite lấy từ thư viện BAC (nhiễm sắc thể nhân tạo từ vi khuẩn) hoặc YAC (nhiễm sắc thể nhân tạo của nấm men) là một phương pháp đầu tiên của sự phân lập những microsatellite mục tiêu đến những vùng của bộ gen mà không đầy đủ marker SSR. Những thư viện chèn vào lớn như BAC và YAC không được sử dụng thường xuyên ở thực vật cho sự phân lập microsatellite vì những thư viện chèn vào lớn chỉ thích hợp với một vài loài thực vật. BAC đã

được sử dụng thành công cho mục đích này ở đậu nành (Cregan và ctv, 1999) và có một ví dụ về việc sử dụng YAC ở nấm (Chen và ctv, 1995). Bất lợi của BAC và YAC là nguy cơ về sự lây nhiễm DNA eukaryote mà có chứa trình tự microsatellite (Cregan và ctv, 1999).

2.6.6.3 Microsatellite từ thư viện cDNA

Microsatellite thu được từ thư viện cDNA thì tương đương với microsatellite thu từ dữ liệu EST nếu EST là trình tự cDNA. Microsatellite từ thư viện cDNA có thể được sàng lọc từ trình tự trong một dữ liệu (ví dụ như EST) hoặc phân lập từ sự sàng lọc tự nhiên của những dòng thư viện qua việc lai với những đoạn chèn có chứa microsatellite. Microsatellite thu được từ sự sàng lọc những dòng thư viện cDNA với mồi oligo là một hướng thường được sử dụng trong nghiên cứu người và động vật (David và Maddox, 1997; Ruyter-Spira và ctv, 1998) và sử dụng ở mức độ thấp hơn ở thực vật. Ví dụ cho việc sử dụng nó ở thực vật là ở lúa (Panaud và ctv, 1995), khoai tây (Milbourne và ctv, 1998).

2.6.6.4 Microsatellite có nguồn gốc từ dữ liệu

a) GenBank và những dữ liệu trình tự công cộng

Một vài nghiên cứu đầu tiên về microsatellite dựa trên tính hữu ích của microsatellite trong những trình tự từ các dữ liệu công cộng như EMBL hay GenBank. Một số nghiên cứu từ những năm cuối thế kỷ 20 bằng việc sử dụng microsatellite từ nguồn dữ liệu là ở khoai tây (Milbourne et al., 1998), cây lúa miến (Brown et al., 1996), lúa mạch (Barker và Heun, 1995), cà chua (Smulders et al., 1997), đậu nành (Akkaya et al., 1992) và nhiều loài khác. Sự phân lập microsatellite từ nguồn dữ liệu này bao phủ tất cả trình tự có sẵn và thường gồm dữ liệu dạng cDNA hay EST. Microsatellite từ nguồn dữ liệu như EMBL và GenBank được xác định dễ dàng qua việc phân loại trên máy tính và chỉ yêu cầu thiết kế mồi cho những trình tự bên cạnh. Điều này làm cho microsatellite từ nguồn dữ liệu có chi phí thấp, ít tốn công và đáng tin cậy hơn so với những hướng trước đây. Tiêu chuẩn được thiết lập cho sự phân loại trên máy này có thể khác nhau, như một ví dụ theo Thiel và ctv (2003) tìm kiếm tất cả sự lặp lại

dinucleotide với n ≥ 6, sự lặp lại trinucleotide với n ≥ 5, sự lặp lại tetranucleotide với n ≥ 5, sự lặp lại pentanucleotide với n ≥ 5, sự lặp lại hexanucleotide với n ≥ 5. Bất lợi đầu tiên của việc thu được microsatellite từ nguồn dữ liệu công cộng là thường chỉ có một số lượng nhỏ microsatellite phù hợp với một số loài.

Một nghiên cứu về dữ liệu công cộng cho microsatellite trên cà chua (Smulders và ctv, 1997) nhận ra rằng 42% SSR nằm trên vùng upstream hoặc downstream của một gen, 26% ở intron, 22% ở cDNA và chỉ có 10% nằm trên DNA có mã hóa. Kết quả tương tự được nhận thấy trên khoai tây (Milbourne và ctv, 1998). Smulders và ctv (1997) cũng thấy rằng sự xuất hiện của dạng lặp lại phụ thuộc vào vị trí của microsatellite. Upstream hay downstream của gen và ở intron, 61% sự lặp lại là dinucleotide. Ở cDNA chỉ 37% sự lặp lại là dinucleotide, và ở exon là chỉ 13%. Với trinucleotide, Smulders thấy rằng hầu như có xu hướng ngược lại mặc dù nó thuộc vào dạng đặc biệt.

b) Dữ liệu EST

Microsatellite là một khuynh hướng đã được sử dụng ở người (Haddad và ctv, 1997) mà đã trở nên hữu ích ở thực vật khi dữ liệu EST đã trở nên phổ biến hơn. Ngày nay, khuynh hướng đặc biệt này đã được thực hiện ở lúa (Miyao và ctv, 1996; Cho và ctv, 2000). Microsatellite EST về mặt chức năng giống với microsatellite từ cDNA, điểm khác biệt rõ nhất là giữa sự tìm kiếm trình tự trên máy từ dữ liệu EST và việc lai những dòng cDNA. Microsatellite từ EST có những thuận lợi là nhanh chóng giải thích (bằng sự phân loại trên máy), phong phú, hiện diện ở những nơi nhiều gen, và có thể dời chuyển cao (Cho và ctv, 2000; Scott và ctv, 2000).

Bất lợi của microsatellite từ EST so với những phương pháp khác là nó phụ thuộc vào sự hiện diện trước đó của dữ liệu trình tự, và có thể ít đa hình hơn so với những microsatellite ngẫu nhiên. Microsatellite từ EST ít đa hình hơn so với microsatellite từ thư viện gen vì có một áp lực bảo tồn trình tự trong các vùng gen nên làm giảm sự đa hình. Tuy nhiên microsatellite từ EST vẫn có mức độ đa hình hữu ích cho việc lập bản đồ, và nghiên cứu sự tiến hóa (Meyer và ctv, 1995; Cho và ctv, 2000).

2.6.6.5 Kiểm tra microsatellite từ một loài có liên quan

Kiểm tra microsatellite từ một loài có liên quan với một loài khác là một phương pháp mong muốn của các nhà nghiên cứu, vì nó không đòi hỏi kỹ thuật cao, chi phí thấp. Sự hạn chế đầu tiên của việc sử dụng microsatellite từ những loài khác là chỉ có một phần microsatellite từ loài khác sẽ hữu dụng, số lượng và những loài mà microsatellite đã phát triển thì bị hạn chế. Thông thường việc sử dụng những mồi khác loại cũng yêu cầu sự tối ưu hóa hơn những mồi tương đồng. Mồi khác loại sẽ tạo ra những sản phẩm với kích thước không mong muốn, hoặc tạo ra những sản phẩm với kích thước mong muốn nhưng không phải SSR. Sản phẩm từ mồi khác loại dễ biến đổi qua các quá trình lai, giải trình tự (Westman và Kresovich, 1998) trước khi sử dụng trong những nghiên cứu có ý nghĩa.

2.6.7. Ưu điểm và hạn chế của phương pháp microsatellite

2.6.7.1. Ưu điểm

Thuận lợi to lớn của sự phân tích microsatellite là phương pháp này biểu hiện số lượng lớn sự đa hình. Một locus ở đậu nành (*Glycine max*) được báo cáo là có 26 alen (Cregan và ctv, 1994). Hơn nữa, khả năng phân biệt các cá thể khi có sự kết hợp các locus được kiểm tra làm cho phương pháp này rất hữu dụng trong các thí nghiệm dòng chảy gen, xác định cây trồng và phân tích mối quan hệ cha con (Hokanson và ctv, 1998).

Microsatellite là marker đồng trội, do đó dị hợp tử có thể dễ dàng được xác định. Tính đồng trội của microsatllite sẽ gia tăng sự hiệu quả và độ chính xác của những phép tính toán di truyền quần thể dựa trên những marker này so với những marker khác, như AFLP và RAPD. Hơn nữa, việc xác định dị hợp tử ở thế hệ F1 sẽ làm cho những phân tích phả hệ, sự lai giống, dòng chảy gen trở nên dễ dàng hơn (Schlotterer và Pemberton, 1994).

Khi các primer SSR đã được xác định, việc sàng lọc các vật liệu sử dụng kỹ thuật này hoàn toàn không đắt tiền. Hơn nữa, sự khuếch đại SSR giữa các loài nghĩa là sự xác định những primer SSR thích hợp không cần thiết trong những loài có quan hệ gần. Ví dụ, ba bộ primer microsatellite đã được thiết kế ở

Malus domestica (Rosaceae), các microsatellite này cung cấp 35 loci, trong số đó có những primer có thể khuếch đại các loài Malus khác (Guilford và ctv, 1997; Gianfranceschi và ctv, 1998; Hokanson và ctv, 1998).

2.6.7.2. Hạn chế

Hạn chế của phương pháp microsatellite là không thể áp dụng phân tích trên một hệ thống lớn bao gồm nhiều loài có quan hệ di truyền xa nhau, điều này là do microsatellite có tỉ lệ đột biến quá cao dẫn đến 2 trở ngại. Thứ nhất, trình tự vùng flanking ở 2 bên vùng microsatellite thường khác nhau giữa các loài do đột biến, vì vậy khó có thể áp dụng primer microsatellite của loài này cho loài khác. Thứ hai, do tỉ lệ đột biến cao nên khi 2 loài có cùng kết quả phân tích với 1 trình tự microsatellite, ví dụ như AC_{19}, chúng ta cũng không thể kết luận rằng 2 loài đó có cùng nguồn gốc tổ tiên ban đầu, vì có thể 1 loài phân ly từ tổ tiên của chúng là AC_{18} rồi đột biến thành AC_{19}, còn 1 loài phân ly từ tổ tiên của chúng là AC_{20} rồi đột biến thành AC_{19}.

Phần 3

PHƯƠNG TIỆN VÀ PHƯƠNG PHÁP TIẾN HÀNH

3.1. Thời gian và địa điểm

Khoá luận được thực hiện tại Trung Tâm Phân Tích Thí Nghiệm Trường Đại Học Nông Lâm Thành Phố Hồ Chí Minh từ tháng 2 đến tháng 7 năm 2006.

3.2. Phương tiện

Máy vi tính cài đặt hệ điều hành Microsoft Windows Server 2003

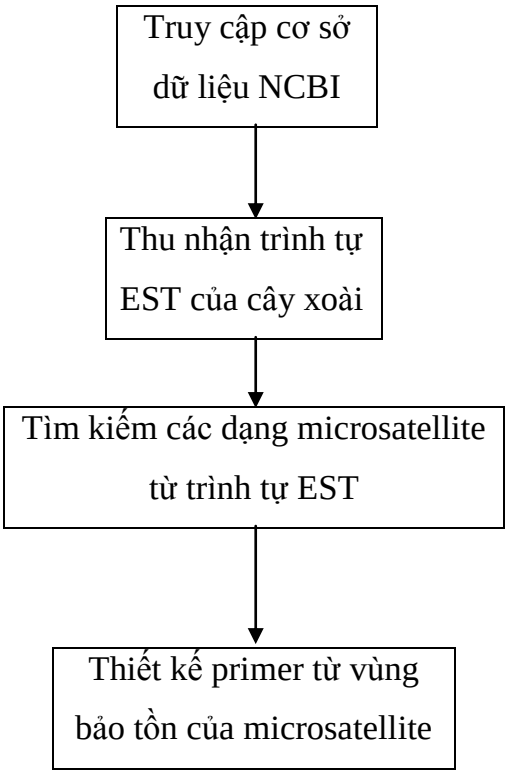
Đường truyền internet

Các phần mềm soạn thảo lập trình Perl như: UltraEdit, Notepad, Wordpad…

Trình biên dịch Active Perl 5.8

Các phần mềm sinh học như BioEdit, Primer3...

3.3 Phương pháp



Hình 3.1. Sơ đồ chung các bước tiến hành

3.3.1. Thu nhận trình tự EST của cây xoài từ NCBI

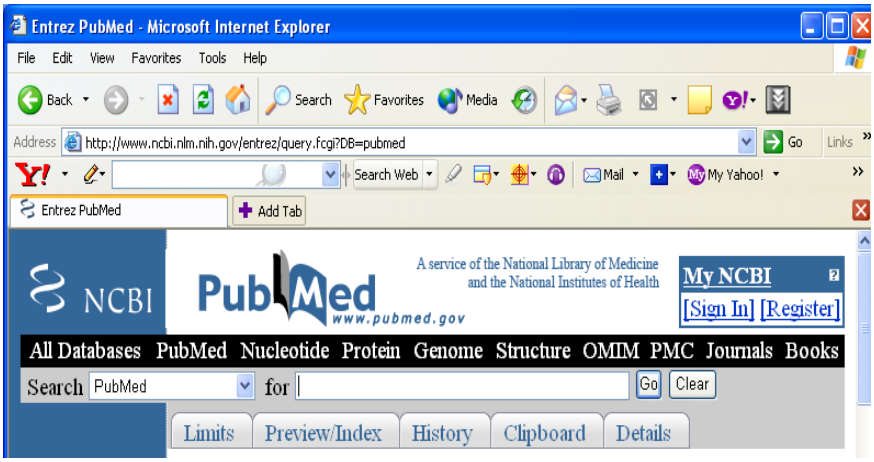
3.3.1.1. NCBI và EST

Hiện nay nhiều nhà khoa học cũng như các trung tâm giải trình tự bộ gen đã tạo ra hàng trăm hàng ngàn EST cho việc sử dụng công cộng. Khi EST đã được tạo ra, các nhà khoa học công bố chúng trong GenBank, được quản lý bởi NCBI. Có rất nhiều EST được đưa vào, do đó nó trở nên khó khăn cho việc xác định một trình tự mà trình tự đó đã được gửi vào cơ sở dữ liệu. EST trở nên dễ dàng truy cập và là một công cụ phát hiện gen hữu dụng, EST cần được tổ chức sắp xếp thành một cơ sở dữ liệu có thể tìm kiếm được và cũng có thể hỗ trợ sự tiếp cập với các dữ liệu gen khác. Do đó, vào năm 1992, các nhà khoa học ở NCBI đã phát triển một cơ sở dữ liệu mới được thiết kế như một tập hợp EST. Khi một EST đã được sàng lọc, chú thích và được đưa vào GenBank, sau đó nó được gửi vào cơ sở dữ liệu mới này, gọi là dbEST.

3.3.1.2 Truy cập cơ sở dữ liệu và thu nhận trình tự

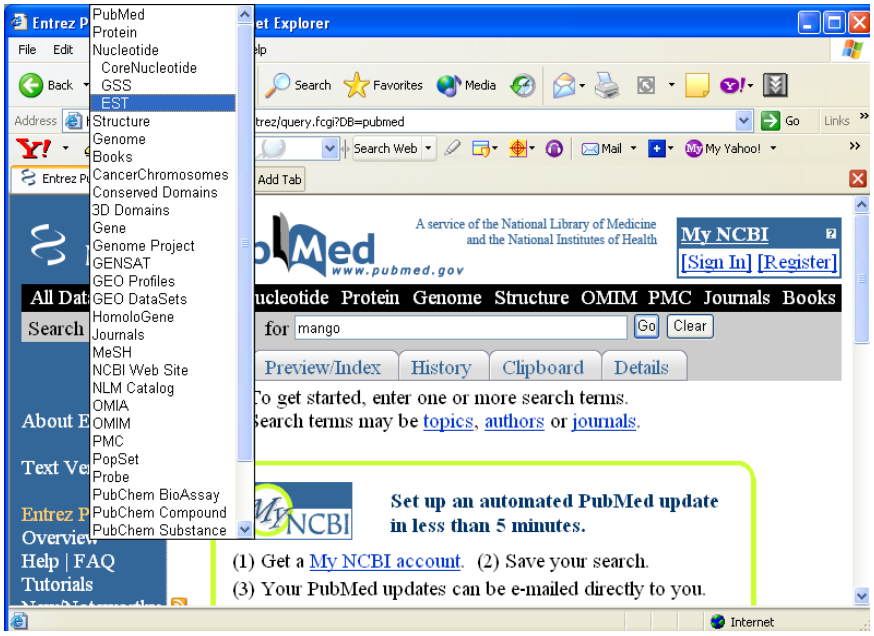
Để tìm các trình tự EST, chúng tôi sử dụng công cụ Entrez EST tìm trên toàn bộ các trình tự EST chứa trong hệ thống GenBank (NCBI), có liên kết với các cơ sở dữ liệu EMBL, DDBJ và một số hệ thống dữ liệu khác trên thế giới.

Truy cập vào trang web của NCBI tại địa chỉ http://www.ncbi.nlm.nih.gov/entrez



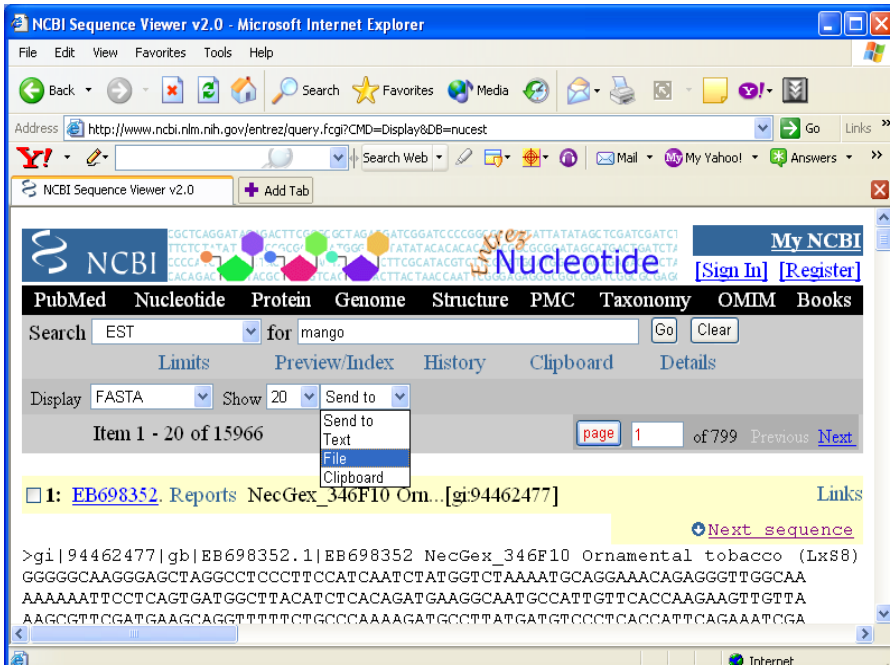
Hình 3.2. Trang entrez của NCBI

Chọn EST tại khung Search và "mango" tại khung for để truy cập trình tự EST của cây xoài.



Hình 3.3. Trang tìm kiếm trình tự

Lựa chọn định dạng "FASTA" ở khung Display và tải toàn bộ trình tự kiếm được bằng cách lựa chọn "File" tại khung Send to.



Hình 3.4. Tải toàn bộ trình tự

3.3.2. Sắp xếp các trình tự EST

Sử dụng một chương trình Perl có tên là “est_trimmer.pl” để sắp xếp các trình tự.

Cú pháp

est_trimmer.pl <FASTAfile> [-amb=n, win] [-tr5=N, n, win] [-tr3=N, n, win] [-cut=min, max] [-id=name]

Giải thích

<FASTAfile>	Là file đơn ở định dạng FASTA chứa các trình tự
[-amb=n, win]	Loại bỏ những đoạn ở ngoại biên chứa “n” nucleotide nhiều nghĩa hay mơ hồ
[-tr5=N, n, win]	Loại bỏ những đoạn có chứa các dạng N={A, C, G, T} từ đầu 5’. Giá trị “n” xác định số lần lặp lại thấp nhất của “N” trong mỗi đoạn ở đầu 5’ có kích thước “win”
[-tr3=N, n, win]	Tương tự nhưng xét trình tự ở đầu 3’
[-cut= min, max]	Xác định kích thước nhỏ nhất và lớn nhất của trình tự
[-id=name]	Kết quả cuối cùng sau khi chạy chương trình được lưu trong file “name”.results, và các bước của tiến trình được liệt kê trong file “name”.log. Nếu không nhập tên “id”, kết quả sẽ được gắn vào <FASTAfile>.

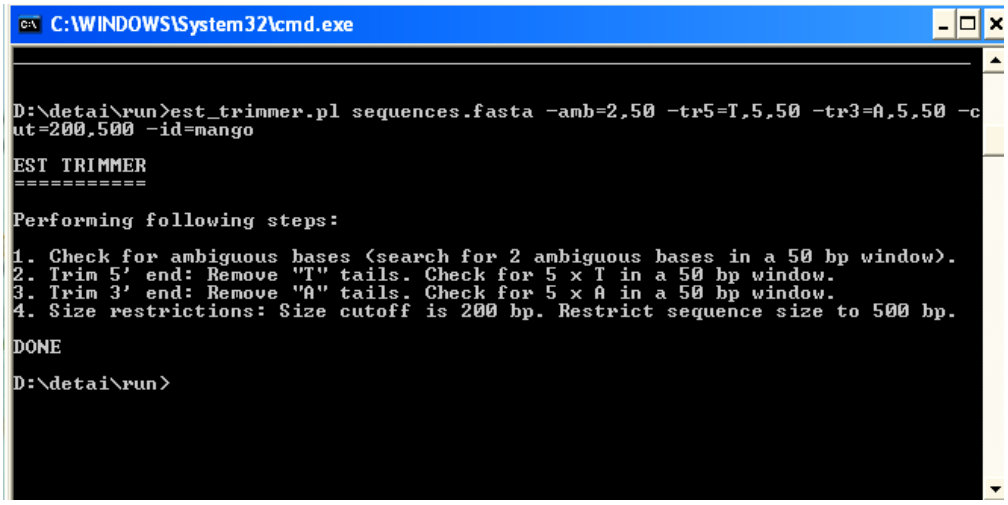
Với các tùy chọn trên chúng tôi xác định những yêu cầu chạy chương trình est_trimmer.pl như sau

-amb=2, 5, 50: kiểm tra các base nhiều nghĩa (tìm 2 base nhiều nghĩa trở lên trong mỗi đoạn 50 bp

-tr5=T, 5, 50: cắt ở đầu 5’, loại bỏ đuôi “T”, kiểm tra trong mỗi đoạn 50bp

-tr3=A, 5, 50: cắt ở đầu 3’, loại bỏ đuôi “A”, kiểm tra trong mỗi đoạn 50 bp

-cut=200, 500: loại bỏ những trình tự nhỏ hơn 200 bp, kích thước trình tự giới hạn là 500 bp

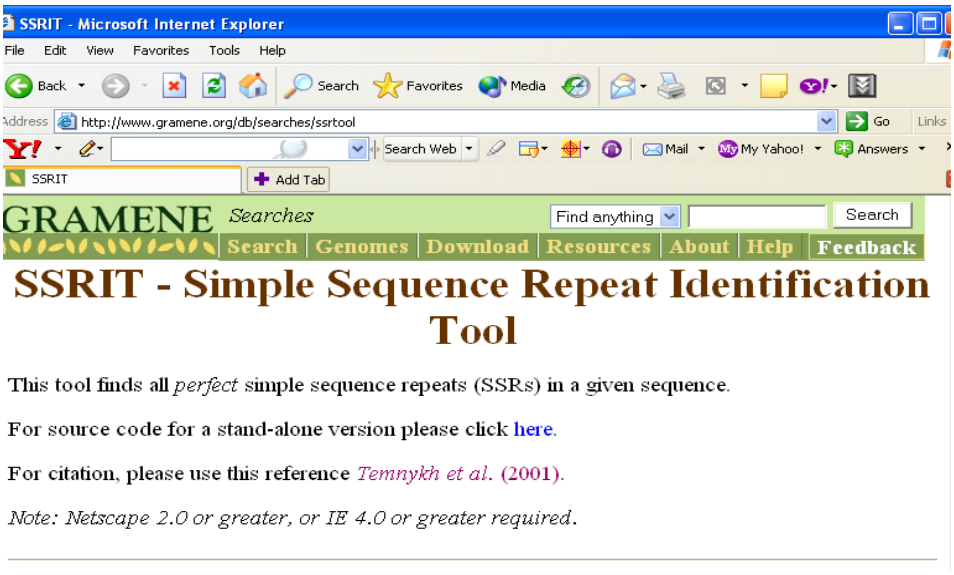


Hình 3.5. Chạy chương trình est_trimmer.pl

3.3.3. Tìm kiếm microsatellite

3.3.3.1. Công cụ SSRIT (Simple Sequence Repeat Identification Tool)

Đây là một chương trình tìm kiếm tất cả các microsatellite có trong các trình tự đưa vào. Chương trình này có thể sử dụng hoàn toàn miễn phí tại địa chỉ Internet http://www.gramene.org/db/searches/ssrtool



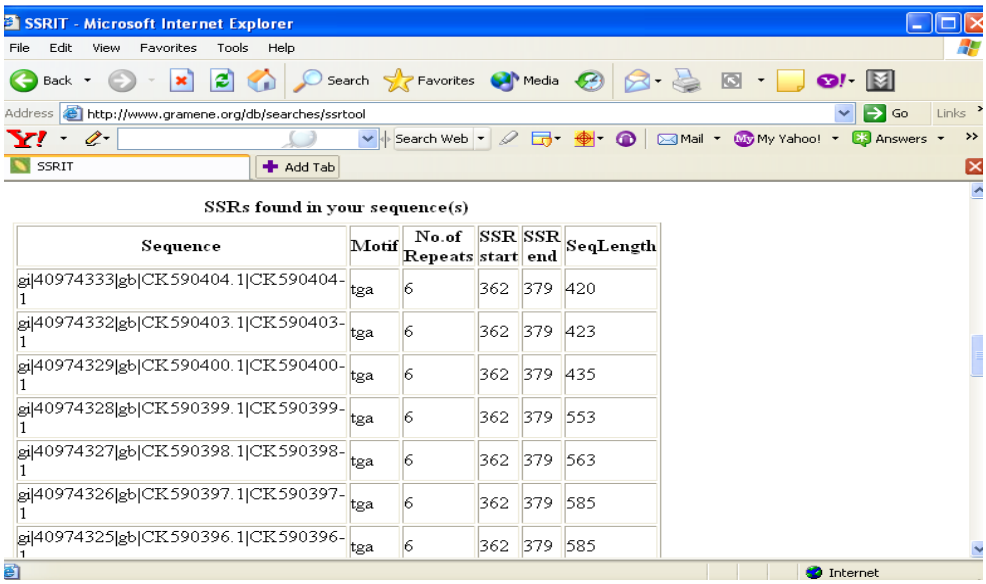
Hình 3.6. Công cụ SSRIT

Các thông số tìm kiếm được lựa chọn như sau

- Chọn lựa dạng của SSR, ví dụ nếu bạn muốn tìm tất cả các SSR đến hexamers (nghĩa là bạn muốn tìm dimers, trimers, tetramers, pentamers và hexamers bạn chỉ cần chọn lựa “hexamers”.

- Nhập vào số lần lặp lại tối thiểu của SSR
- Dán hoặc nhập vào trình tự cần tìm microsatellite
- Nhất nút "Find SSR" hoặc nhấn Enter

Kết quả sẽ xuất hiện dưới dạng một danh sách liệt kê bao gồm các mục tên trình tự, dạng SSR, số lần lặp lại, vị trí bắt đầu (có SSR), vị trí kết thúc, chiều dài trình tự



Sequence	Motif	No.of Repeats	SSR start	SSR end	SeqLength
gi|40974333|gb|CK590404.1|CK590404-1	tga	6	362	379	420
gi|40974332|gb|CK590403.1|CK590403-1	tga	6	362	379	423
gi|40974329|gb|CK590400.1|CK590400-1	tga	6	362	379	435
gi|40974328|gb|CK590399.1|CK590399-1	tga	6	362	379	553
gi|40974327|gb|CK590398.1|CK590398-1	tga	6	362	379	563
gi|40974326|gb|CK590397.1|CK590397-1	tga	6	362	379	585
gi|40974325|gb|CK590396.1|CK590396-1	tga	6	362	379	585

Hình 3.7. Kết quả tìm SSR của SSRIT

SSRIT có nhược điểm là chỉ tìm được một số lượng giới hạn trình tự (500 trình tự) trong một lần chạy chương trình và không có sự thống kê kết quả tìm kiếm

3.3.3.2. Công cụ MISA (Microsatellite Identification Tool)

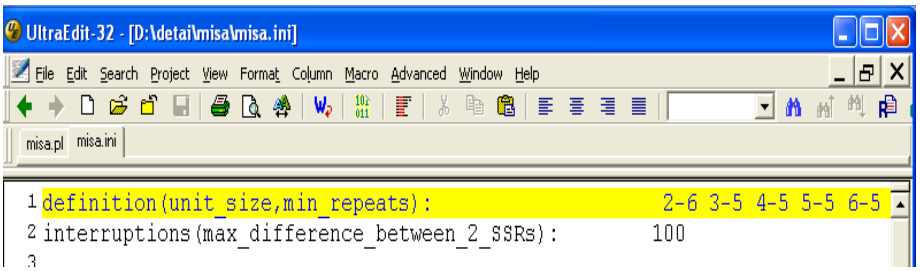
Công cụ này cho phép sự xác định và định vị microsatellite cũng như các microsatellite ghép (compound microsatellite) mà bị ngắt quãng bởi một số base nhất định. Công cụ này được viết từ ngôn ngữ lập trình Perl và có thể tải về máy sử dụng từ địa chỉ internet sau http://pgrc.ipk-gatersleben.de/misa/misa.html

Cú pháp

misa.pl <FASTAfile> với <FASTAfile> là file chứa trình tự ở định dạng FASTA

Công cụ này cần thêm một file chứa các thông số cho quá trình xác định microsatellite, file này có định dạng "misa.ini".

File misa.ini có cấu trúc như sau



Hình 3.8. File misa.ini

Trong đó hàng đầu tiên xác định dạng và số lần lặp lại tối thiểu của microsatellite. Hàng thứ hai là số nucleotide tối đa chèn vào giữa hai microsatellite.

Kết quả của việc tìm kiếm microsatellite sẽ được lưu trong hai file

File “<FASTAfile>.misa” lưu các giá trị sự định vị và sự xác định các microsatellite

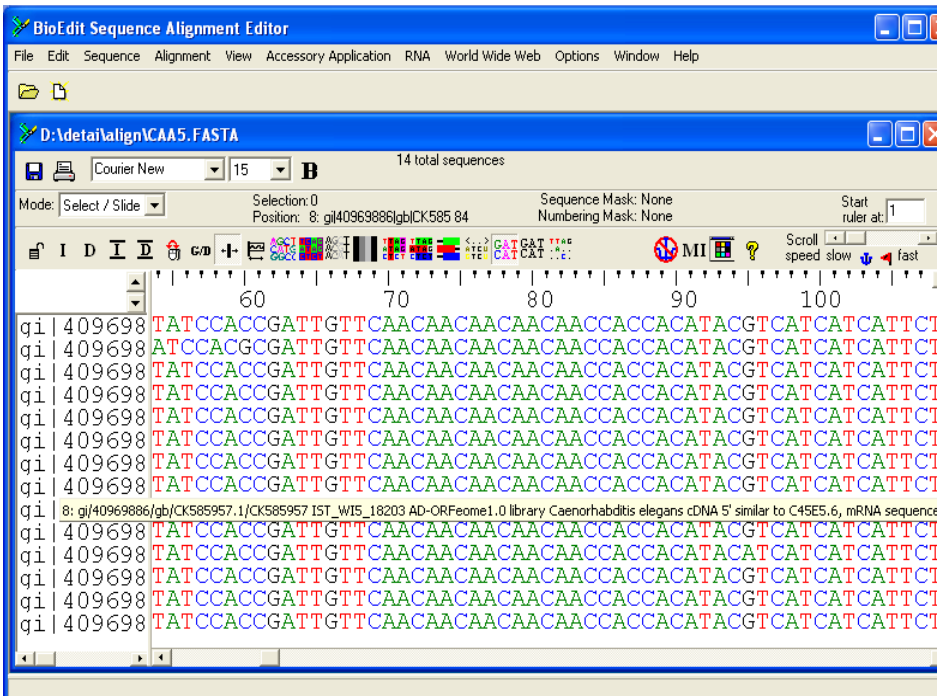
File “<FASTAfile>.statistics” thống kê kết quả của quá trình tìm kiếm

3.3.4. Xác định vùng bảo tồn

Vùng bảo tồn là vùng ở hai bên trình tự microsatellite, những trình tự này thường giống nhau đối với mỗi một dạng microsatellite. Vùng bảo tồn rất quan trọng trong phân tích microsatellite bởi đây chính là cơ sở cho việc thiết kế primer.

Để xác định được vùng bảo tồn, chúng tôi tiến hành việc sắp gióng cột (alignment) các trình tự.

Sắp gióng cột dựa vào microsatellite, chúng tôi lấy trình tự microsatellite làm điểm tập trung, sau đó tạo trình tự bảo tồn (consensus sequence) đối với mỗi dạng microsatellite bằng công cụ “CAP Contig Assembly Program” có trong phần mềm BioEdit.



Hình 3.9. Sắp gióng cột trình tự

3.3.5. Thiết kế primer

Primer là những đoạn nucleotide ngắn, bắt cặp bổ sung với đầu 5’ hay đầu 3’ của mạch DNA khuôn mẫu. Primer được thiết kế dựa vào vùng trình tự đã được biết, nằm ở hai đầu của đoạn gen cần khuếch đại.

Thông số quyết định sự thành công của phản ứng PCR là việc thiết kế primer. Một primer được thiết kế không tốt có thể sẽ cho kết quả ít hay không có sản phẩm do sự khuếch đại không chuyên biệt và/hay sự hình thành cấu trúc thứ cấp, các cấu trúc này sẽ cạnh tranh và ngăn chặn sự tạo thành sản phẩm mong muốn.

Việc thiết kế và chọn lựa primer phải thỏa mãn một số yêu cầu sau

- Chiều dài primer: chiều dài tốt là 18 đến 24 base. Chiều dài này đủ dài để đảm ảo tính chuyên biệt và đủ ngắn để primer bám vào mạch mẫu dễ dàng ở nhiệt độ bắt cặp.
- Nhiệt độ nóng chảy (T_m) là nhiệt độ mà một nửa sợi đôi DNA tách ra trở thành sợi đơn và cho biết tính ổn định của sợi đôi. Thành phần (G+C) trong DNA cao sẽ dẫn tới nhiệt độ T_m cao vì liên kết H trong DNA cao hơn. Có nhiều công thức tính T, hai trong những công thức được nhiều người sử dụng là

$$T_m = 59.9 + 0.41 * (\%GC) – 675 / \text{chiều dài}$$

$T_m = 2 (A+C) + 4 (G+C)$ (công thức Wallace)

Primer với nhiệt độ nóng chảy trong phạm vi 55 ^oC đến 72 ^oC thường cho kết quả tốt nhất.

- Tính chuyên biệt: primer phải được lựa chọn sao cho chỉ có một trình tự duy nhất trong DNA mẫu được khuếch đại. Vì Taq polymerase có hoạt tính trong một phạm vi nhiệt độ rộng, sự kéo dài primer sẽ xảy ra ở nhiệt độ thấp hơn nhiệt độ bắt cặp. Nếu sự thay đổi nhiệt độ quá chậm sự không chuyên biệt sẽ xảy ra và enzyme sẽ xúc tác sự kéo dài nếu có một sự tương đồng ngắn ở đầu 3'.
- Thành phần base: ảnh hưởng đến độ đặc hiệu của quá trình bắt cặp, nhiệt độ nóng chảy, nhiệt độ bắt cặp và sự ổn định của cấu trúc phân tử. Các base được sắp xếp ngẫu nhiên thì thích hợp hơn là những vùng (A+T) dài hay là những vùng giàu (G+C). Thành phần (G+C) trung bình khoảng từ 50% đến 60% sẽ cho nhiệt độ nóng chảy, nhiệt độ bắt cặp thích hợp trong một phản ứng PCR bình thường.
- Trình tự primer đầu 3' quyết định tính chuyên biệt và tính tương thích của phản ứng PCR. Các trình tự đầu 3' không nên có

  + G hay C không có nhiều hơn 3 tại vị trí này vì sẽ làm cho primer bắt cặp không chuyên biệt.

  + 3' thymidine, nó làm cho sự bắt cặp sai dễ xảy ra hơn so với những nucleotide khác.

  Các cặp primer nên được kiểm tra sự bổ sung ở đầu 3' vì nó thường dẫn đến sự hình thành cấu trúc thứ cấp.
- Trình tự primer đầu 5': các base ở đầu 5' ít quyết định đến sự bắt cặp của primer. Do đó có thể thêm các yếu tố trình tự như vị trí giới hạn (restriction site) - những vị trí này có thể giống nhau hoặc có cùng đầu dính (đầu bằng) với enzyme giới hạn trong MCS (Multiple Cloning Site) của vector chọn để dòng hóa gen quan tâm. NcoI (CCATGG) hay NdeI (CATATG) thường được sử dụng vì có thể tạo ra codon mở đầu ATG.
- Cấu trúc thứ cấp: nếu sự bắt cặp giữa forward primer với reverse primer (hình thành dimer, hetero-dimer), forward primer với forward

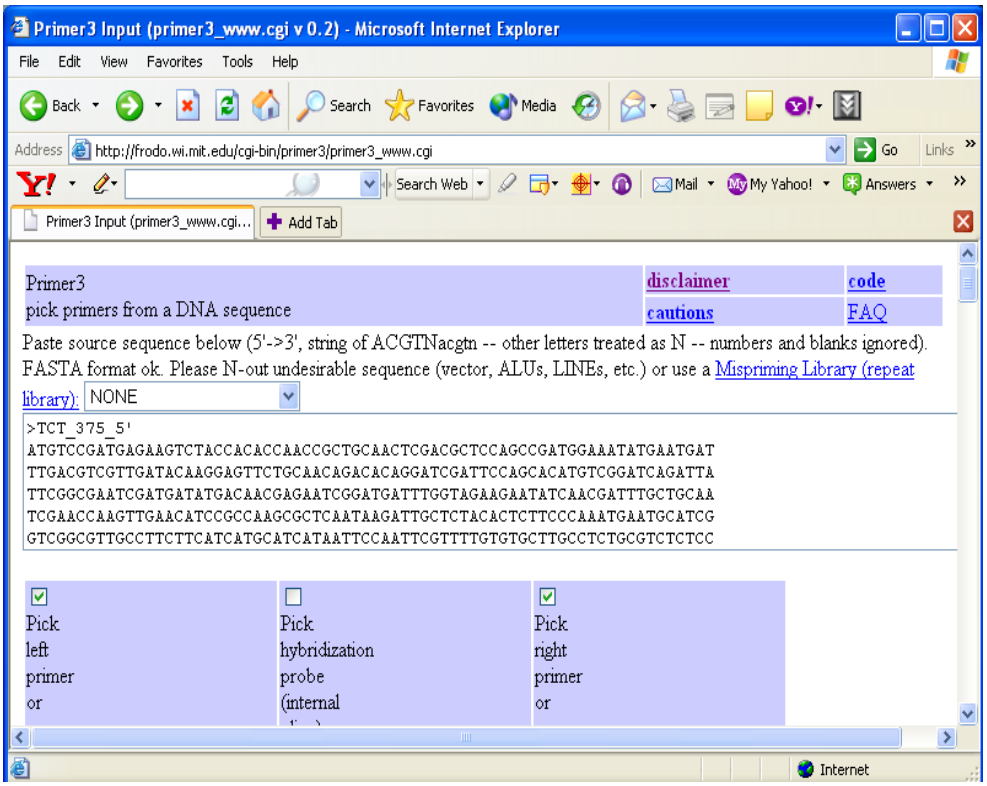
primer hay reverse primer với reverse primer (self-dimer, homo-dimer), hay primer tự tạo thành cấu trúc hairpin (kẹp tóc) xảy ra nhiều hơn so với sự bắt cặp của primer với DNA mẫu thì hiệu quả nhân bản của phản ứng PCR sẽ giảm một cách rõ rệt. Như vậy, nên tránh những trường hợp này.

3.3.5.1. Primer3

Chương trình Primer3 là một trong những chương trình thiết kế primer miễn phí được biết đến nhiều nhất. Chương trình này được tạo ra bởi các nhà khoa học thuộc Viện Nghiên cứu Sinh Y học Whitehead và Trung Tâm Nghiên cứu Genome của MIT (Whitehead Institute for Biomedical Research and MIT Center for Genome Research). Chương trình này có thể được sử dụng hoàn toàn miễn phí tại địa chỉ internet http://www-genome.wi.mit.edu/cgi-bin/primer/primer3_www.cgi

Chương trình Primer3 thiết kế primer cho một trình tự DNA đưa vào, thỏa mãn nhiều tùy chọn khác nhau, chủ yếu là các điều kiện về %GC, về nhiệt độ bắt cặp của primer, kích thước sản phẩm… Đây là chương trình lớn với gần 100 tùy chọn khác nhau tương ứng với các điều kiện mà primer được tạo ra phải thỏa mãn. Có một thuận lợi là hầu hết các tùy chọn này đều có giá trị mặc định của nó, và người dùng có thể không thay đổi các thông số này nếu họ không có nhu cầu đặc biệt.

Khi muốn thiết kế primer cho một gen hay một đoạn trình tự nào đó, trước tiên người sử dụng phải đưa đoạn trình tự DNA của mình vào chương trình và xác định các thông số về trình tự primer, nhiệt độ nóng chảy cua primer, kích thước sản phẩm cần… Các thông số chi tiết khác có thể giữ mặc định.



Hình 3.10. Chương trình Primer3

Dựa trên những yêu cầu cho việc thiết kế primer đã xác định như trên chúng tôi thiết lập các thông số cho chương trình Primer3 như sau

+ Targets(mục tiêu): m, n với m là vị trí bắt đầu có microsatllite và n là chiều dài microsatellite

+Primer size (kích thước primer)

Min: 18; Max: 24

+ Primer Tm (nhiệt độ nóng chảy của pimer)

Min: 55.0; Max: 72.0; Max Tm difference: 2.0

+ Primer %GC

Min: 50.0; Max: 60%

+ Các thông số khác vẫn giữ mặc định.

3.3.5.2. Chương trình Perl ssrfinder_1_0

Đây là một chương trình của tác giả Steven Schroeder thuộc trường Đại học Missouri – Michigan. Chương trình gồm 6 Perl scripts có chức năng xác định SSR và thiết kế primer thích hợp cho mỗi SSR tìm được

- 1_ssr_repeat_finder.pl: tìm SSR, lấy ra trình tự SSR và vùng flanking cho những phân tích sau.

- 2_ssr_primer_designer.pl: thiết kế primer mà mục tiêu là khuếch đại vùng trình tự chứa SSR.

- 3_ssr_primer_rep_check.pl: sàng lọc lại các primer đã thiết kế để loại bỏ những primer có chứa trình tự lặp lại

- 4_ssr_primer_blast.pl: so sánh các primer đã thiết kế với cơ sở dữ liệu primer

- 5_ssr_order_filter.pl: tạo 1 file chỉ chứa SSR mà có primer duy nhất

- 6_ssr_primer_formatter.pl: tạo 1 file chỉ chứa SSR có primer duy nhất – file này được tạo đơn giản chỉ chứa những thông tin cần thiết cho việc chọn lựa primer.

Yêu cầu: vì chương trình này được viết cho hệ điều hành Unix hay Linux nên cần phải thực hiện sửa đổi một số lệnh lập trình cơ bản để có thể chạy trên môi trường Window.

Chương trình cần sự kết hợp với 3 phần mềm khác là Primer3, blastall và formatdb để thực thi. Ba phần mềm này có thể tải hoàn toàn miễn phí (có phiên bản dành cho Window) từ trang Primer3 http://frodo.wi.mit.edu/primer3/code và trang Blast của NCBI http://www.ncbi.nlm.nih.gov/BLAST/download.shtml

Các thông số của chương trình thiết kế primer đều được mặc định như sau

TARGET= m, n với m là vị trí bắt đầu có microsatllite và n là chiều dài microsatellite (mục tiêu)

PRIMER_PRODUCT_SIZE_RANGE=80-160 80-240 80-300 (kích thước sản phẩm)

PRIMER_OPT_SIZE=24 (kích thước tối ưu của primer)

PRIMER_MIN_SIZE=20 (kích thước tối thiểu của primer)

PRIMER_MAX_SIZE=28 (kích thước tối đa của primer)

PRIMER_OPT_TM=63 (nhiệt độ nóng chảy tối ưu của primer)

PRIMER_MIN_TM=60 (nhiệt độ nóng chảy tối thiểu của primer)

PRIMER_MAX_TM=65 (nhiệt độ nóng chảy tối đa của primer)

PRIMER_MAX_DIFF_TM=1 (độ chênh lệch nhiệt độ nóng chảy tối đa)

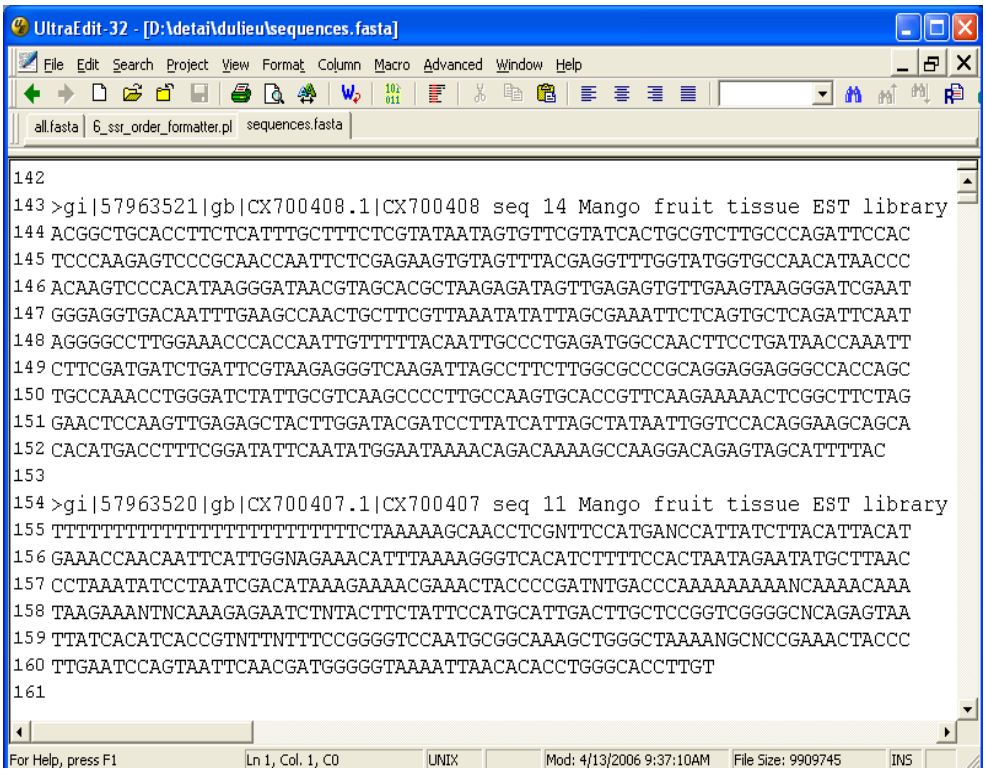
Với các thông số mặc định trên chương trình hoàn toàn có thể sử dụng cho mục tiêu của đề tài.

Phần 4

KẾT QUẢ VÀ THẢO LUẬN

4.1. Thu nhận trình tự EST của cây xoài

Khi sử dụng từ khóa và phương pháp như mục 3.3.1.2 chúng tôi đã tải được toàn bộ 15966 trình tự EST của cây xoài. Các trình tự này ở định dạng FASTA, mỗi trình tự có cấu trúc như hình 4.1



Hình 4.1 Trình tự EST ở định dạng FASTA

Đây là kiểu định dạng phổ biến trong cơ sở dữ liệu của GenBank gồm 2 phần chủ yếu. Phần 1 bắt đầu bằng một dấu “>”, theo sau là các thông tin về trình tự đó như tên trình tự, số gi, accession number…Phần 2 là các nucleotide của trình tự. Định dạng FASTA có ưu điểm là chỉ chứa trình tự và những thông tin thiết yếu về trình tự đó. Hơn nữa trong một file trình tự ở định dạng FASTA có thể có nhiều trình tự so với dạng Plain format chỉ chứa một trình tự.

Mặc khác việc lựa chọn kiểu định dạng trình tự để tải về ở dạng FASTA là sự thuận lợi cho các nghiên cứu sau vì phần lớn các chương trình, phần mềm của tin sinh học đều sử dụng định dạng FASTA.

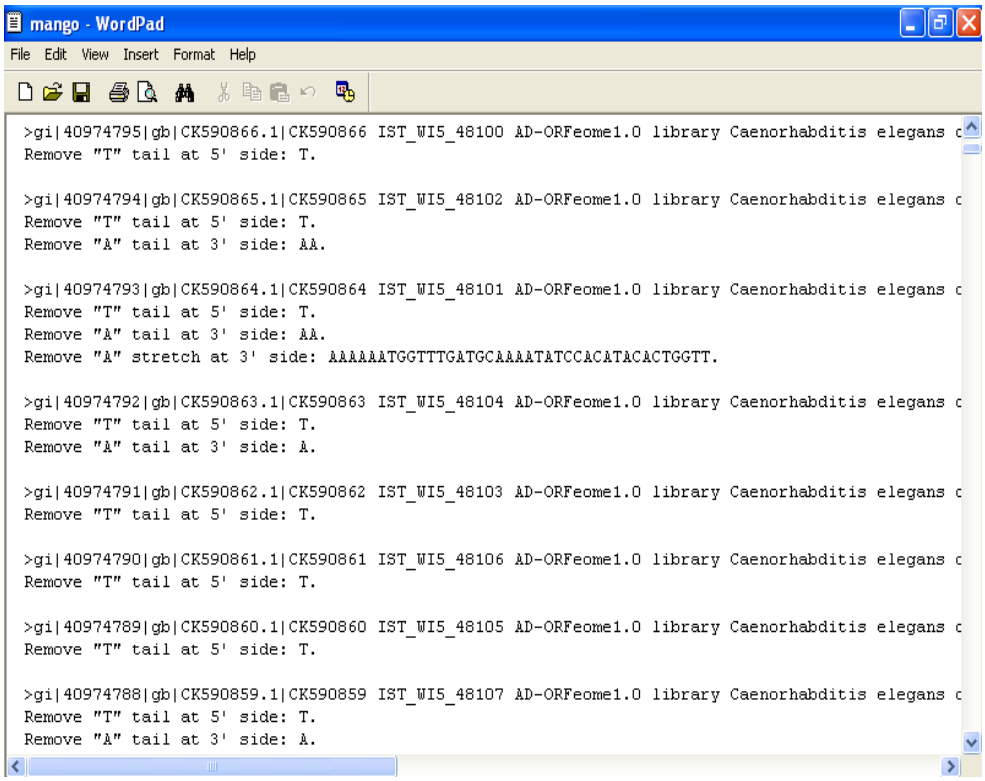
Do mục tiêu của đề tài là tìm kiếm tất cả các microsatellite có trong toàn bộ nguồn dữ liệu EST của cây xoài nên việc tải tất cả 15966 trình tự thuộc cùng một file là điều hợp lý, dễ dàng cho các công việc phân tích sau này.

4.2. Sắp xếp các trình tự

Sau khi chạy chương trình est_trimmer.pl với các thông số như mục 3.3.2 đã trình bày, kết quả đã được lưu trong 2 file

mango.results: chứa tất cả các trình tự thỏa mãn các thông số của bước kiểm tra với est_trimmer.pl

mango.txt: file này ghi nhận tất cả các tiến trình thực thi trên mỗi trình tự không đạt yêu cầu đề ra như loại bỏ trình tự có kích thước bé hơn 100 bp, loại bỏ đuôi poly A, poly T…



Hình 4.2. Tiến trình thực thi của est_trimmer.pl

4.3. Kết quả tìm kiếm microsatellite

4.3.1. Công cụ SSRIT

SSRIT cho phép tìm kiếm một lần là 500 trình tự và tốn khoảng 5 phút. Chính vì công cụ SSRIT không có khả năng tìm kiếm SSR với một số lượng quá lớn trình tự

(15966) nên phải chạy chương trình nhiều lần, tốn nhiều thời gian và phải lặp lại tiến trình. Hơn nữa bảng kết quả của SSRIT không lưu thành file để truy cập, phải xem kết quả trực tuyến, không có sự thống kê về các dạng microsatellite. Do đó chúng tôi xác định rằng công cụ này không phù hợp với mục đích nghiên cứu của đề tài.

4.3.2. Công cụ MISA

Thực thi chương trình MISA cho kết quả rất nhanh và có 2 file được tạo thành

mango.fasta.misa: chứa các thông tin về tên trình tự, dạng SSR, kích thước, vị trí bắt đầu và kết thúc của SSR…

ID	SSR nr.	SSR type	SSR	size	start	end
gi|89481901|gb|DY637695.	1	p2	(GA)6	12	37	48
gi|89478122|gb|DY633916.	1	p2	(GA)6	12	20	31
gi|40974333|gb|CK590404.	1	p3	(TGA)6	18	361	378
gi|40974332|gb|CK590403.	1	p3	(TGA)6	18	361	378
gi|40974329|gb|CK590400.	1	p3	(TGA)6	18	361	378
gi|40974328|gb|CK590399.	1	p3	(TGA)6	18	361	378
gi|40974327|gb|CK590398.	1	p3	(TGA)6	18	361	378
gi|40974326|gb|CK590397.	1	p3	(TGA)6	18	361	378
gi|40974325|gb|CK590396.	1	p3	(TGA)6	18	361	378
gi|40974324|gb|CK590395.	1	p3	(TGA)6	18	361	378
gi|40974323|gb|CK590394.	1	p3	(TGA)6	18	361	378
gi|40974322|gb|CK590393.	1	p3	(TGA)6	18	361	378
gi|40974321|gb|CK590392.	1	p3	(TGA)6	18	361	378
gi|40974320|gb|CK590391.	1	p3	(TGA)6	18	361	378
gi|40974319|gb|CK590390.	1	p3	(TGA)6	18	361	378
gi|40974318|gb|CK590389.	1	p3	(TGA)6	18	361	378
gi|40974299|gb|CK590370.	1	p3	(CCA)5	15	61	75
gi|40974277|gb|CK590348.	1	p3	(CCA)5	15	61	75
gi|40974259|gb|CK590330.	1	p3	(CCA)5	15	266	280
gi|40974258|gb|CK590329.	1	p3	(CCA)5	15	266	280
gi|40974247|gb|CK590318.	1	p3	(CCA)5	15	266	280
gi|40974246|gb|CK590317.	1	p3	(CCA)5	15	266	280
gi|40974206|gb|CK590277.	1	p3	(CCA)5	15	266	280

Hình 4.3 Nội dung file mango.fasta.misa

mango.fasta.stastistics: là file thống kê kết quả tìm kiếm microsatellite

Dựa vào file mango.fasta.statistics này chúng tôi xác định được số lượng trình tự không đáp ứng yêu cầu mà khi chạy với est_trimmer.pl đã chưa thống kê được: ban đầu chúng tôi có 15966 trình tự, sau khi chạy est_trimmer thì có 231 trình tự không thỏa mãn yêu cầu do đó tổng số trình tự được kiểm tra là 15735.

Kết quả của việc tìm kiếm microsatellite được liệt kê ở bảng 4.1, số lượng SSR được xác định chiếm tỷ lệ 1.7% tổng số trình tự EST được kiểm tra, phân bố tương đối nhiều trong bộ gen. Với nghiên cứu trên đối tượng là cây nho (Scott và ctv, 2000),

SSR chiếm 2.5% và trên cây bông vải (Qureshi, 2004) SSR chiếm 1.34% tổng số trình tự EST.

Bảng 4.1. Kết quả tìm kiếm microsatellite

Kết quả	**Số lượng**
Tổng số trình tự kiểm tra	15735
Tổng kích thước của trình tự đã kiểm tra (bp)	7400551
Tổng số SSR được xác định	267
Tổng số trình tự có SSR	265
Số trình tự có nhiều hơn 1 SSR	2
Số SSR hiện diện ở dạng SSR ghép	1

Sự phân bố các dạng lặp lại có số lượng khác nhau theo bảng 4.2.

Bảng 4.2. Sự phân bố các dạng lặp lại của SSR

Đơn vị lặp lại	**Số SSR**	**Tỷ lệ %**
2 (dinucleotide)	11	4.12
3 (trinucleotide)	255	95.51
4 (tetranucleotide)	1	0.37

Bảng trên cho thấy phần lớn các microsatellite của cây xoài là dạng trinucleotide (95.51%). Các dạng khác xuất hiện rất ít, điều này tương tự đối với microsatllite ghép (compound microsatellite). Trong các nghiên cứu khác trên đối tượng cây nho dạng trinucleotide là chỷ yếu chiếm tỷ lệ là 62.90%. Trên cây mía dạng dinucleotide có tỷ lệ 38.19%, trong khi dạng trinucleotide có tỷ lệ 31.49%. Sự phân bố các dạng lặp lại của microsatellite có sự khác biệt giữa các đối tượng nghiên cứu khác nhau.

Ưu điểm của MISA so với các phương pháp khác là kết quả tìm kiếm của MISA được lưu thành file, có sự thống kê các dạng microsatellite, chương trình thực thi nhanh, máy tính không cần nối mạng internet chỉ cần tải mã (code) của chương trình về máy tính sau đó có thể áp dụng đơn giản và hiệu quả.

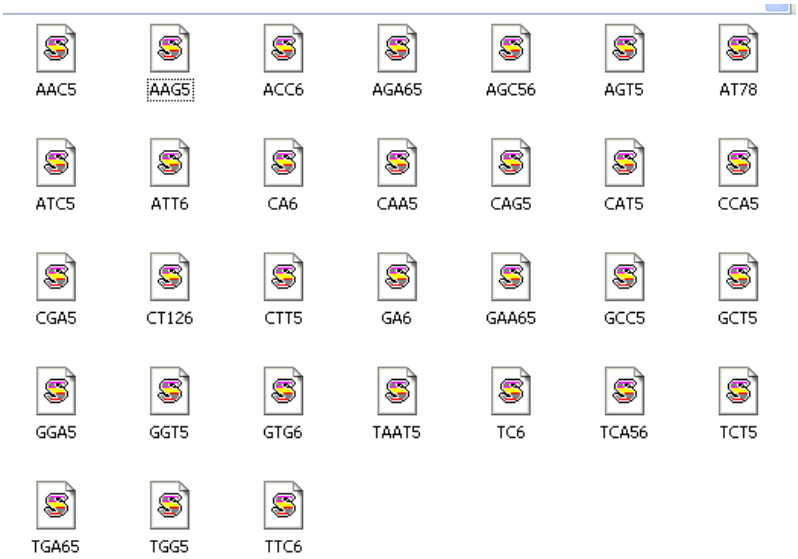
Nhược điểm của chương trình này là không có chức năng phân loại các trình tự theo dạng microsatellite đã tìm kiếm. Do đó chúng tôi phải tiến hành phân nhóm trình tự bằng cách thủ công dựa theo các dạng microsatellite đã xác định được

Kết quả đạt được cho thấy

Có tất cả 31 loại SSR trong đó dạng dinucleotide có 5 loại, dạng trinucleotide có 25 loại và tetranucleotide có 1 loại.

Bảng 4.3. Các loại SSR

Dạng dinucleotide	**Dạng trinucleotide**	**Dạng tetranucleotide**
AT; CA; CT; TC; GA	AAC; AAG; ACC; AGA; AGC; AGT; ATC; ATT; CAA; CAG; CAT; CCA; CGA; CTT; GAA; GCC; GCT; GGA; GGT; GTG; TCA; TCT; TGA; TGG; TTC	TAAT



Hình 4.4 Các file trình tự sau khi phân nhóm

Vì các dạng dinucleotide và tetranucleotide chiếm tỷ lệ và tần suất xuất hiện thấp nên chúng tôi chỉ tiếp tục nghiên cứu đối với dạng trinucleotide.

4.4. Xác định vùng bảo tồn

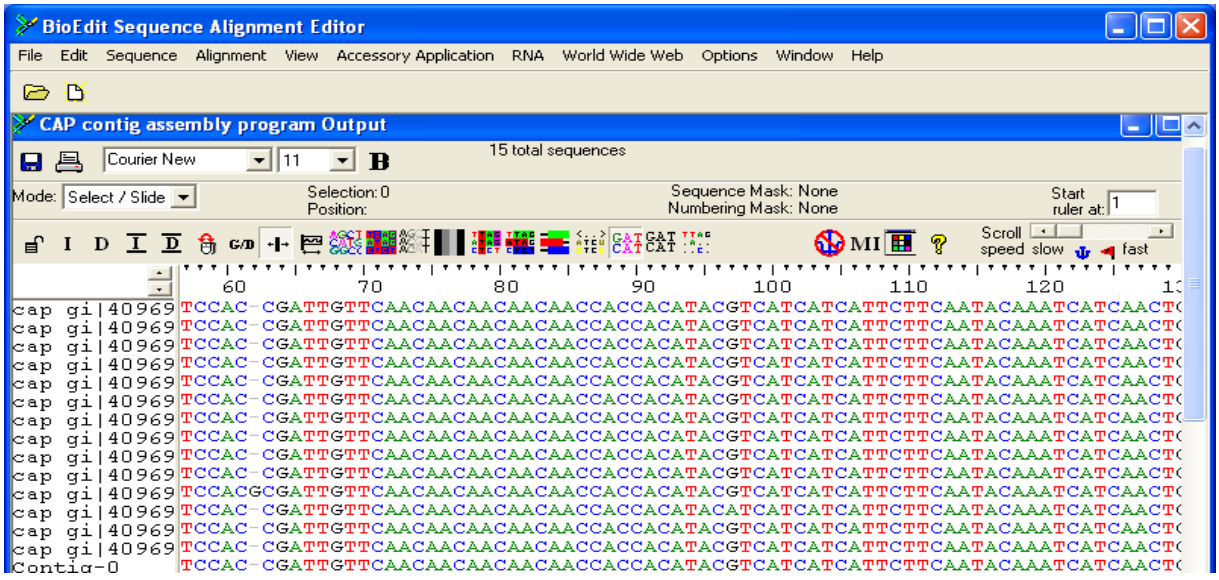
Để kết quả align (sắp gióng cột) và xác định vùng bảo tồn được chính xác, chúng tôi loại bỏ các loại microsatellite có tần suất xuất hiện thấp (<2.5%)

Sau quá trình sàng lọc chúng tôi còn lại 7 loại microsatellite

Bảng 4.4. Các loại microsatellite nghiên cứu

Số thự tự	Loại microsatellite	Số trình tự	Tần suất xuất hiện
1	AGA	31	11.61%
2	CAA	14	5.24%
3	CAT	24	8.99%
4	CCA	73	27.34%
5	TCA	7	2.62%
6	TCT	32	11.99%
7	TGA	16	5.99%

Thực hiện việc sắp gióng cột và tạo trình tự bảo tồn dựa trên công cụ CAP contig assembly program chúng tôi có được 6 loại microsatellite, microsatellite AGA sau khi thực hiện 2 tiến trình nói trên cho thấy sự khác biệt về trình tự ở vùng flanking nên không thể thiết kế được primer chung cho tất cả 31 trình tự.



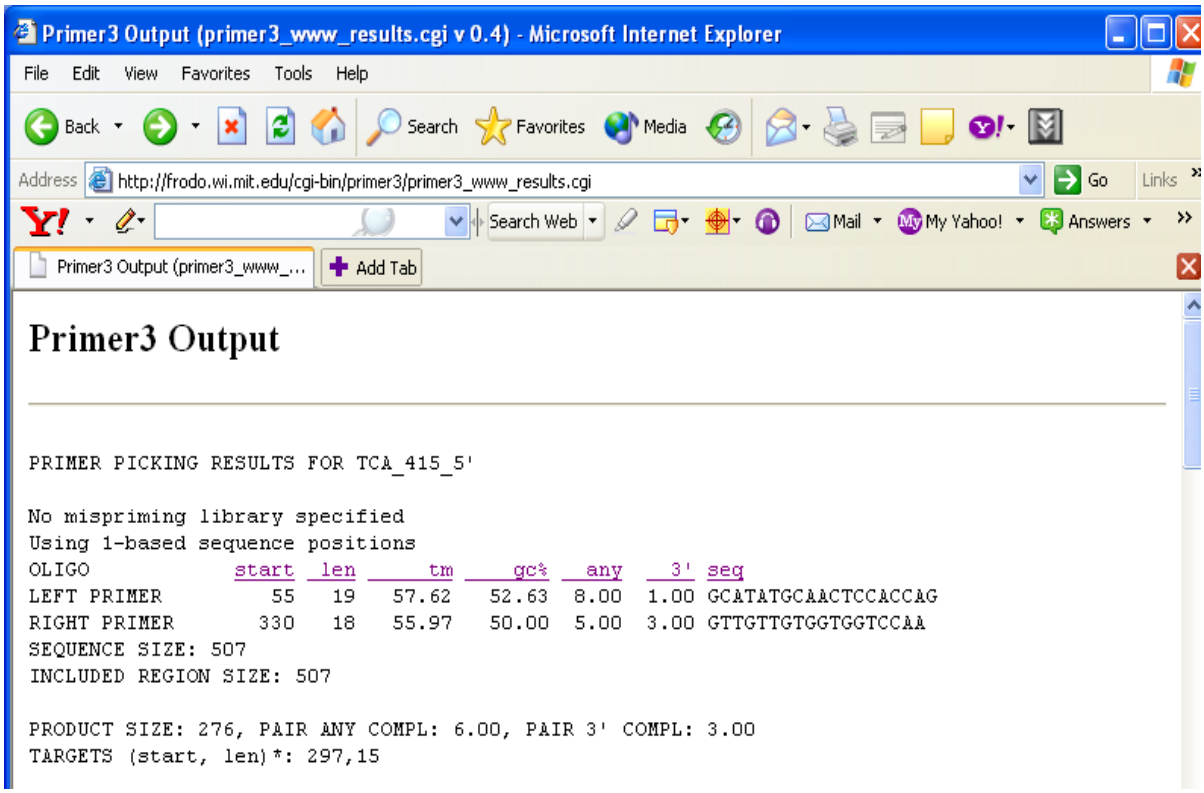
Hình 4.5. Xác định vùng bảo tồn của microsatellite CAA

4.5. Thiết kế primer đối với 6 microsatellite

4.5.1. Primer3

Việc thiết kế primer và chọn lựa các thông số phải thỏa các yêu cầu đã nêu trong mục 3.3.5.1.

Kết quả của chương trình Primer3 của 6 microsatellite



Hình 4.6. Kết quả thiết kế primer của microsatellite TCA

Chúng tôi đã thiết kế được 6 cặp primer cho 6 loại microsatellite. Theo bảng 4.5, chúng tôi liệt kê loại SSR, forward primer, reverve primer và nhiệt độ nóng chảy (Tm)

Bảng 4.5 Kết quả thiết kế primer từ chương trình Primer3

Loại SSR	Forward primer (5’ - 3’)	Reverse primer (5’ - 3’)
CAA	GTCTCAAACCCCAAATCC	TGTCCACCAGAACCAGAG
CAT	ACCGCCAGTCTATGGAACAC	TACTTAGCGGCCATCAAACC
CCA	TTCCAAGAAGCACTGTGTGG	TTGTGGTTGAGGTTGCTCAG
TCA	GCATATGCAACTCCACCAG	GTTGTTGTGGTGGTCCAA
TCT	TGCTGCAATCGAACCAAG	GCAGGAGGCACAAATAGTAGTC

TGA	AGCAGTGAATCCGGTGATCT	GGTTTCAGCGCCTTATCAAC

4.5.2. Chương trình Perl script ssrfinder_1_0

Lần lượt thực thi từng file từ 1 đến 6 của gói công cụ này, chúng tôi có được kết quả sau

- 1_ssr_repeat_finder.pl có 3 file kết quả

  new_id20060715.txt: liệt kê các số truy cập, tên trình tự có SSR

  labdout20060715.txt: liệt kê dạng SSR, vị trí và chiều dài SSR, trình tự

  ssrout20060715.txt: liệt kê SSR, trình tự có chứa SSR và vùng flanking

- 2_ssr_primer_designer.pl có 3 file kết quả

  primerin.txt: liệt kê các thông số mặc định cho quy trình thiết kế primer (xem mục 3.3.5.3)

  raw_primer20060715.out: liệt kê các kết quả tính toán cho mỗi primer được thiết kế

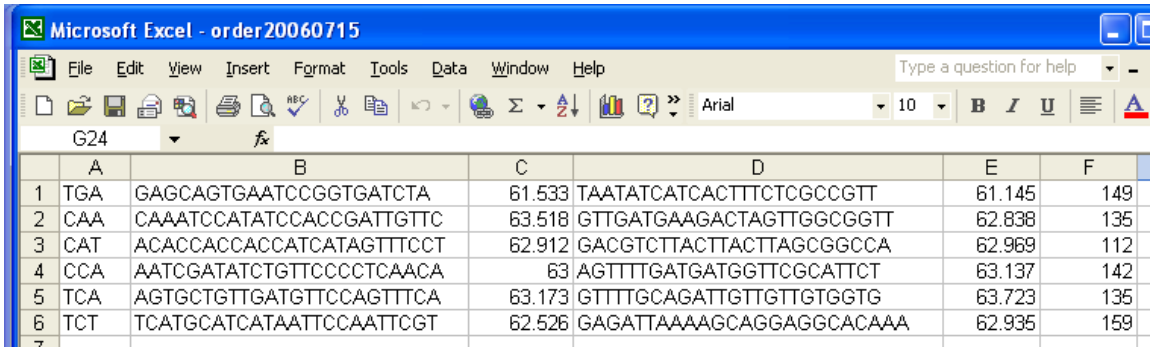
  primer_result20060715.txt: liệt kê SSR, trình tự, cặp primer đã thiết kế, nhiệt độ nóng chảy và chiều dài sản phẩm của primer.

TGATGATGATGATGATGA	TGA	ATATGGCGTCGAGTACAATGGCG	GAGCAGTGAATCCGGTG	61.533	TA
CAACAACAACAACAA	CAA	GGAAAAGTGATTGGTCCACTGAT	CAAATCCATATCCACCG	63.518	GT
CATCATCATCATCAT	CAT	AGCCTACGGGAATGTTCCACCGC	ACACCACCACCATCATA	62.912	GA
CCACCACCACCACCA	CCA	ACAAAGACTACAAACTCGAACCA	AATCGATATCTGTTCCCC	63	AG
TCATCATCATCATCA	TCA	TGAAGTATTGCCACCAACAGATG	AGTGCTGTTGATGTTCCA	63.173	GT
TCTTCTTCTTCTTCT	TCT	ACATCCGCCAAGCGCTCAATAAG	TCATGCATCATAATTCCA	62.526	GA

Hình 4.7 Nội dung file primer_result20060715.txt

- 3_ssr_primer_check.pl tạo file rescreened20060715.txt có nội dung giống file primer_result20060715.txt vì không có primer được thiết kế có nhiều trình tự lặp lại nên bước kiểm tra này vẫn giữ nguyên giá trị cũ.
- 4_ssr_primer_blast20060715.pl cũng cho kết quả giống primer_result20060715.txt vì cơ sở dữ liệu primer hiện tại không có.
- 5_ssr_order_filter.pl: chương trình này tạo file filter20060715.txt có chứa dạng SSR, trình tự primer, và các thông số của primer khác
- 6_ssr_order_formatter tạo file cuối cùng là order20060715.txt, file này chỉ chứa các thông tin cần thiết cho việc thiết kế mồi như dạng microsatellite, forward primer và reverse primer.

* Kết quả của chương trình thiết kế primer



	A	B	C	D	E	F
1	TGA	GAGCAGTGAATCCGGTGATCTA	61.533	TAATATCATCACTTTCTCGCCGTT	61.145	149
2	CAA	CAAATCCATATCCACCGATTGTTC	63.518	GTTGATGAAGACTAGTTGGCGGTT	62.838	135
3	CAT	ACACCACCACCATCATAGTTTCCT	62.912	GACGTCTTACTTACTTAGCGGCCA	62.969	112
4	CCA	AATCGATATCTGTTCCCCTCAACA	63	AGTTTTGATGATGGTTCGCATTCT	63.137	142
5	TCA	AGTGCTGTTGATGTTCCAGTTTCA	63.173	GTTTTGCAGATTGTTGTTGTGGTG	63.723	135
6	TCT	TCATGCATCATAATTCCAATTCGT	62.526	GAGATTAAAAGCAGGAGGCACAAA	62.935	159

Hình 4.8. Kết quả thiết kế primer

Trong hình 4.8 cột A là loại SSR, cột B là forward primer, cột C và E là nhiệt độ nóng chảy, cột D là reverse primer, cột F là chiều dài sản phẩm tạo thành.

Từ kết quả thiết kế primer thu được từ 2 chương trình khác nhau, chúng tôi nhận thấy các primer trên đều thích hợp cho phản ứng PCR tùy vào yêu cầu, mục đích thực nghiệm. Primer ở 2 chương trình có kết quả khác nhau là do việc lựa chọn các thông số cho tiến trình thiết kế mồi khác nhau.

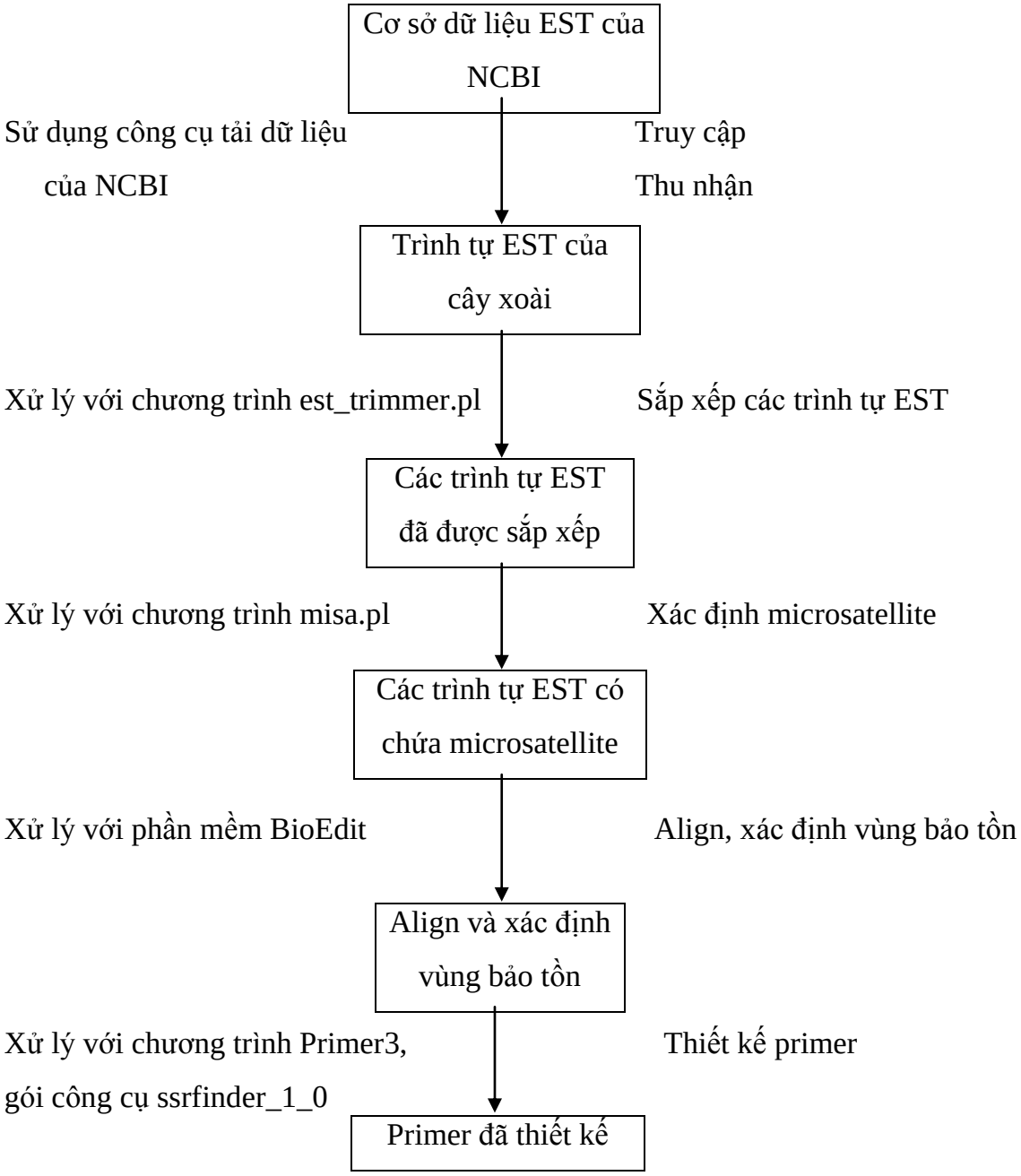
Phần 5

KẾT LUẬN VÀ ĐỀ NGHỊ

5.1. Kết luận

Từ quá trình thực hiện đề tài chúng tôi rút ra kết luận như sau

Xây dựng phương pháp phát hiện microsatellite đối với cây xoài từ nguồn cơ sở dữ liệu EST hiện có, đó là sử dụng các chương trình và công cụ est_trimmer.pl, misa.pl, CAP contig assembly program, Primer3 hay gói công cụ ssrfinder_1_0.



Hình 5.1. Sơ đồ phương pháp thực hiện

Kết quả đạt được

Đã tải được 15966 trình tự EST của cây xoài

Xác định và định vị được 267 microsatellite trong đó dạng trinucleotide chiếm tỉ lệ cao nhất (95.51%)

Xác định được vùng bảo tồn của 6 loại microsatellite và thiết kế được primer cho 6 loại microsatellite bao gồm các loại sau: CAA, CAT, CCA, TCA, TCT và TGA.

5.2 Đề nghị

Đề tài nên được tiếp tục phân tích bằng thực nghiệm để kiểm tra hiệu quả của việc thiết kế primer.

Viết và sử dụng chương trình hay phần mềm có khả năng phân loại các trình tự đã xác định SSR.

Áp dụng quy trình xác định microsatellite cho các đối tượng thí nghiệm khác nếu đối tượng thí nghiệm đó có sẵn nguồn dữ liệu từ các cơ sở dữ liệu.

Xây dựng cơ sở dữ liệu trình tự, microsatellite và các primer đã thiết kế.

Tăng cường việc đào tạo, phát triển ngành tin sinh học vì đó là một ngành hỗ trợ đắc lực cho công tác nghiên cứu.

Phần 6

TÀI LIỆU THAM KHẢO

TIẾNG VIỆT

1. Phạm Văn Duệ, 2005. *Giáo trình kỹ thuật trồng cây ăn quả*. Nhà xuất bản Hà Nội. Trang 105.
2. Phạm Thị Hương, Trần Thế Tục, Nguyễn Quang Thạch, 2003. *Cây xoài và những điều cần biết*. Nhà xuất bản Nông Nghiệp Hà Nội. 95 trang.
3. Trần Thế Tục, 1994. *Kỹ thuật trồng và chăm sóc Xoài Na Hồng Xiêm*. Nhà xuất bản Nông Nghiệp Hà Nội. Trang 25.
4. Lê Minh Trung, Quốc Bình, 2002. *Ngôn ngữ lập trình Perl cho người mới học*. Nhà xuất bản Thống kê. 426 trang.
5. Trần Văn Lăng, 2003. *Một số tổng quan về sinh tin học*. Phân viện Công nghệ Thông tin tại TP Hồ Chí Minh.
6. Trần Linh Thước, 2004. *Thực tập bioinformatics*. Đại học Khoa học Tự nhiên TP Hồ Chí Minh.
7. Lưu Phúc Lợi, 2005. *Giáo trình sinh tin học ứng dụng*. Bộ môn Công nghệ Sinh học – Đại học Nông Lâm TP Hồ Chí Minh.

TIẾNG NƯỚC NGOÀI

8. Andreas D. Baxevanis, B. F. Francis Ouellette, 2005. *Bioinformatics a practical guide to the analysis of genes and proteins*. 3rd edition, A John Wiley & Sons, Inc., Publication.
9. James Tisdall, 2001. *Beginning Perl for Bioinformatics*. 1st edition, O'Reilly, Publication.
10. Lorenzo Cerutti, 2002. *EST clustering*. Swiss Institute of Bioinformatics
11. Scott K. D., 2001. *Microsatellites derived from ESTs, and their comparison with those derived by other methods*. Centre for plant conservation genetics, Southern Cross University, Lismore, Australia.
12. Scott K. D., P. Eggler, G. Seaton, M. Rossetto, E. M. Ablett, L. S. Lee, R. J. Henry, 2000. *Analysis of SSRs derived from grape ESTs*.

13. Samina N. Qureshi, Sukumar Saha, Ramesh V. Kantety, and J. N. Jenkins, 2004. *EST-SSR: A New Class of Genetic Markers in Cotton*. The journal of cotton science 8:112 – 123. The cotton foundation.

TÀI LIỆU TỪ CÁC TRANG WEB

14. Julian P. Robinson, Stephen A. Harris, 1999. *Amplified Fragment Length Polymorphisms and Microsatellites: A phylogenetic perspective*. Department of Plant Sciences, University of Oxford, South Parks Road, Oxford OX1 3RB, Great Britain
http://webdoc.gwdg.de/ebook/y/1999/whichmarker/m12/Chap12.htm
15. Vincent R. Prezioso. *General notes on Primer Design in PCR*. Biosystems Laboratory, Brinkmann Instruments Inc., Westbury, New York.
16. http://www.maizemap.org/bioinformatics/SSRFINDER/README.ssrfinder
17. http://www.eppendorfna.com/applications/PCR_appl_primer.asp
18. http://www.woodrow.org/teachers/esi/2002/Biology/Projects/p3/definition.htm
19. http://www.bioteach.ubc.ca/Bioinformatics/whatisbioinform/index.htm
20. http://www.ncbi.nlm.nih.gov/About/primer/est.html
21. http://www.answers.com/microsatellite
22. http://bmj.bmjjournals.com/cgi/content/full/324/7344/1018
23. http://tinsinhhoc.org

PHỤ LỤC

1. Mã chương trình est_trimmer.pl

```
#!/usr/bin/perl -w
# Check for arguments. If none display syntax #
if (@ARGV == 0)
  {open (IN,"<$0");
  while (<IN>) {if (/^\#\# (.*)/) {$message .= "$1\n"}};
  close (IN);
  die $message;}
# Check if help is required #
if ($ARGV[0] =~ /-help/i)
  {open (IN,"<$0");
  while (<IN>) {if (/^\#\#\#(.*)/) {$message .= "$1\n"}};
  close (IN);
  die $message;}
# Open FASTA file #
open (IN,"<$ARGV[0]") || die ("\nError: File doesn't exist !\n\n");
# Checking arguments #
$arg = @ARGV;
$arg > 1 || die ("\nError: No arguments determined !\n\n");
for ($i = 1; $i < $arg; $i++)
    {if (($amb_n,$amb_win) = ($ARGV[$i] =~ /-amb=(\d+),(\d+)/i))
      {$message .= "$i. Check for ambiguous bases (search for $amb_n
ambiguous bases in a $amb_win bp window).\n";}
    elsif (($tr3_b,$tr3_n,$tr3_win) = ($ARGV[$i] =~ /-
tr3=([ACGT]),(\d+),(\d+)/i))
      {$message .= "$i. Trim 3' end: Remove \"$tr3_b\" tails.";
      if ($tr3_win > 0) {$message .= " Check for $tr3_n x $tr3_b in a
$tr3_win bp window.\n"} else {$message .= "\n"};}
    elsif (($tr5_b,$tr5_n,$tr5_win) = ($ARGV[$i] =~ /-
tr5=([ACGT]),(\d+),(\d+)/i))
      {$message .= "$i. Trim 5' end: Remove \"$tr5_b\" tails.";
      if ($tr5_win > 0) {$message .= " Check for $tr5_n x $tr5_b in a
$tr5_win bp window.\n"} else {$message .= "\n"};}
    elsif (($cut_min,$cut_max) = ($ARGV[$i] =~ /-cut=(\d+),(\d+)/i))
      {$message .= "$i. Size restrictions: Size cutoff is $cut_min bp.
Restrict sequence size to $cut_max bp.\n";}
    elsif (($file) = ($ARGV[$i] =~ /-id=(.*)/i))
      {}
```

```
    else {die ("\nError: Argument nr. ",++$i," is invalid !\n\n")}};
# Open results & log file#
if ($file) {open (OUT,">$file.results") || die ("\nError: Output file
name not valid !\n\n");open (LOG,">$file.log")}
else {open (OUT,">$ARGV[0].results");open (LOG,">$ARGV[0].log")};
print "\nEST TRIMMER\n==========\n\nPerforming following steps:\n\n";
print LOG "\nLOG FILE OF ALL PERFORMED MODIFICATIONS\n";
print LOG "=====================================\n\n";
print LOG "\nPerforming following steps:\n-------------------------
\n\n";
print $message;
print LOG $message;
print LOG "\n\nProcessing steps for each sequence:\n--------------------
---------------\n";
# core #
$/ = ">";
while (<IN>)
  {next unless (($seqname,$seq) = /(.*?)\n(.*)/s);
  $seq =~ s/[\d\s>]//g;
  $seq =~ s/[^ACGTN]/N/gi; # only "acgtn" characters allowed
  $message = '';
  $discard = 'nö';
  for ($i = 1; $i < $arg; $i++)
    {if (($amb_n,$amb_win) = ($ARGV[$i] =~ /-amb=(\d+),(\d+)/i))
      {&Trim_5_n;
      &Trim_3_n;}
    elsif (($tr3_b,$tr3_n,$tr3_win) = ($ARGV[$i] =~ /-
tr3=([ACGT]),(\d+),(\d+)/i))
      { &Trim_3_oligoN }
    elsif (($tr5_b,$tr5_n,$tr5_win) = ($ARGV[$i] =~ /-
tr5=([ACGT]),(\d+),(\d+)/i))
      {&Trim_5_oligoN }
    elsif (($cut_min,$cut_max) = ($ARGV[$i] =~ /-cut=(\d+),(\d+)/i))
     {if ($cut_max < length $seq)
     {$seq = substr $seq,0,$cut_max;
        $message .= "Restrict sequence size to $cut_max bp.\n"};
      if ($cut_min and (length $seq < $cut_min))
        {$message .= "Discard sequence (below $cut_min bp cutoff).\n";
        $discard = "yo;}
  if ($discard eq 'nö')
    {$seq = join ("\n",grep($_,split(/(.{70})/,$seq)));
    print OUT ">$seqname\n$seq\n"};
```

```
  if ($message ne '') {print LOG "\n>$seqname\n$message"} };
print "\nDONE\n";
close (IN);
close (OUT);
close (LOG);
# subroutines #
sub Trim_5_n
  # trim 5' end until specified window contains less than n ambiguous
bases
  {$check = '0';
  while ((length $seq > $amb_win) and !($check eq '1'))
    {$window = substr $seq,0,$amb_win;
    if ($window =~ /^(([^N]*N){$amb_n})/i)
      {$seq =~ s/^($1N*)//i;
      $message .= "Ambiguous sequence at 5' side: $1.\n" }
   else {$check = '1'};};
  sub Trim_3_n
  # trim 3' end until specified window contains less than n ambiguous
bases
  {$check = '0';
  while ((length $seq > $amb_win) and !($check eq '1'))
    {$window = substr $seq,-$amb_win;
    if ($window =~ /.*((N[^N]*){$amb_n})/i)
      { $seq =~ s/(N*$1)$//i;
      $message .= "Ambiguous sequence at 3' side: $1.\n" }
    else {$check = '1'}; };
sub Trim_5_oligoN
  # remove oligoN stretches from 5' side
  {$check = '0';
  if ($seq =~ s/^($tr5_b+)//i) {$message .= "Remove \"$tr5_b\" tail at
5' side: $1.\n"};
  while (!($check eq '1'))
    { if (length $seq > $tr5_win) {$window = substr $seq,0,$tr5_win}
else {$window = $seq};
    if ($window =~ /^(.*?($tr5_b){$tr5_n})/i)
      { $seq =~ s/^($1$tr5_b*)//i;
      $message .= "Remove \"$tr5_b\" stretch at 5' side: $1.\n" }
    else {$check = '1'};  };
sub Trim_3_oligoN
  # remove oligoN stretches from 3'side
  {$check = '0';
```

```
    if ($seq =~ s/($tr3_b+)$//i) {$message .= "Remove \"$tr3_b\" tail at
  3' side: $1.\n"};
    while (!($check eq '1'))
      {if (length $seq > $tr3_win) {$window = substr $seq,-$tr3_win} else
  {$window = $seq};
      if ($window =~ /.*(($tr3_b){$tr3_n}.*)/i)
        { $seq =~ s/($tr3_b*$1)$//i;
        $message .= "Remove \"$tr3_b\" stretch at 3' side: $1.\n"}
      else {$check = '1'};}
    };
```

2. Mã chương trình misa.pl

```
#!/usr/bin/perl -w
#$$$$$ DECLARATION $$$$$#
# Check for arguments. If none display syntax #
if (@ARGV == 0)
  {open (IN,"<$0");
  while (<IN>) {if (/^\#\# (.*)/) {$message .= "$1\n"}};
  close (IN);
  die $message;};
# Check if help is required #
if ($ARGV[0] =~ /-help/i)
  {open (IN,"<$0");
  while (<IN>) {if (/^\#\#\#(.*)/) {$message .= "$1\n"}};
  close (IN);
  die $message;};
# Open FASTA file #
open (IN,"<$ARGV[0]") || die ("\nError: FASTA file doesn't exist !\n\n");
open (OUT,">$ARGV[0].misa");
print OUT "ID\tSSR nr.\tSSR type\tSSR\tsize\tstart\tend\n";
# Reading arguments #
open (SPECS,"misa.ini") || die ("\nError: Specifications file doesn't
exist !\n\n");
my %typrep;
my $amb = 0;
while (<SPECS>)
   { %typrep = $1 =~ /(\d+)/gi if (/^def\S*\s+(.*)/i);
   if (/^int\S*\s+(\d+)/i) {$amb = $1}};
my @typ = sort { $a <=> $b } keys %typrep;
#$$$$$ CORE $$$$$#
$/ = ">";
my $max_repeats = 1; #count repeats
```

```
my $min_repeats = 1000; #count repeats
my (%count_motif,%count_class); #count
my ($number_sequences,$size_sequences,%ssr_containing_seqs); #stores number
and size of all sequences examined
my $ssr_in_compound = 0;
my ($id,$seq);
while (<IN>)
  {next unless (($id,$seq) = /(.*?)\n(.*)/s);
  my ($nr,%start,@order,%end,%motif,%repeats); # store info of all SSRs
from each sequence
  $seq =~ s/[\d\s>]//g; #remove digits, spaces, line breaks,...
  $id =~ s/^\s*//g; $id =~ s/\s*$//g;$id =~ s/\s/_/g; #replace whitespace
with "_"
  $number_sequences++;
  $size_sequences += length $seq;
  for ($i=0; $i < scalar(@typ); $i++) #check each motif class
    { my $motiflen = $typ[$i];
    my $minreps = $typrep{$typ[$i]} - 1;
    if ($min_repeats > $typrep{$typ[$i]}) {$min_repeats =
$typrep{$typ[$i]}}; #count repeats
    my $search = "(([acgt]{$motiflen})\\2{$minreps,})";
    while ( $seq =~ /$search/ig ) #scan whole sequence for that class
      { my $motif = uc $2;
      my $redundant; #reject false type motifs [e.g. (TT)6 or (ACAC)5]
      for ($j = $motiflen - 1; $j > 0; $j--)
        { my $redmotif = "([ACGT]{$j})\\1{".($motiflen/$j-1)."}";
        $redundant = 1 if ( $motif =~ /$redmotif/ )};
      next if $redundant;
      $motif{++$nr} = $motif;
      my $ssr = uc $1;
      $repeats{$nr} = length($ssr) / $motiflen;
      $end{$nr} = pos($seq);
      $start{$nr} = $end{$nr} - length($ssr) + 1;
      # count repeats
      $count_motifs{$motif{$nr}}++; #counts occurrence of individual motifs
      $motif{$nr}->{$repeats{$nr}}++; #counts occurrence of specific SSR in
its appearing repeat
      $count_class{$typ[$i]}++; #counts occurrence in each motif class
      if ($max_repeats < $repeats{$nr}) {$max_repeats = $repeats{$nr}};};
    };
  next if (!$nr); #no SSRs
  $ssr_containing_seqs{$nr}++;
```

```
  @order = sort { $start{$a} <=> $start{$b} } keys %start; #put SSRs in
right order
  $i = 0;
  my $count_seq; #counts
  my ($start,$end,$ssrseq,$ssrtype,$size);
  while ($i < $nr)
    { my $space = $amb + 1;
    if (!$order[$i+1]) #last or only SSR
      {$count_seq++;
      my $motiflen = length ($motif{$order[$i]});
      $ssrtype = "p".$motiflen;
      $ssrseq = "($motif{$order[$i]})$repeats{$order[$i]}";
      $start = $start{$order[$i]}; $end = $end{$order[$i++]};
      next};
    if (($start{$order[$i+1]} - $end{$order[$i]}) > $space)
      { $count_seq++;
      my $motiflen = length ($motif{$order[$i]});
      $ssrtype = "p".$motiflen;
      $ssrseq = "($motif{$order[$i]})$repeats{$order[$i]}";
      $start = $start{$order[$i]}; $end = $end{$order[$i++]};
      next };
    my ($interssr);
    if (($start{$order[$i+1]} - $end{$order[$i]}) < 1)
      { $count_seq++; $ssr_in_compound++;
      $ssrtype = 'c*';
      $ssrseq =
"($motif{$order[$i]})$repeats{$order[$i]}($motif{$order[$i+1]})$repeats{$or
der[$i+1]}*";
      $start = $start{$order[$i]}; $end = $end{$order[$i+1]} }
    else
      {$count_seq++; $ssr_in_compound++;
      $interssr = lc substr($seq,$end{$order[$i]},($start{$order[$i+1]} -
$end{$order[$i]}) - 1);
      $ssrtype = 'c';
      $ssrseq =
"($motif{$order[$i]})$repeats{$order[$i]}$interssr($motif{$order[$i+1]})$re
peats{$order[$i+1]}";
      $start = $start{$order[$i]};  $end = $end{$order[$i+1]};
      #$space -= length $interssr };
    while ($order[++$i + 1] and (($start{$order[$i+1]} - $end{$order[$i]})
<= $space))
      { if (($start{$order[$i+1]} - $end{$order[$i]}) < 1)
```

```
        { $ssr_in_compound++;
        $ssrseq .= "($motif{$order[$i+1]})$repeats{$order[$i+1]}*";
        $ssrtype = 'c*';
        $end = $end{$order[$i+1]}}
      else
        { $ssr_in_compound++;
        $interssr = lc substr($seq,$end{$order[$i]},($start{$order[$i+1]} -
$end{$order[$i]}) - 1);
        $ssrseq .= "$interssr($motif{$order[$i+1]})$repeats{$order[$i+1]}";
        $end = $end{$order[$i+1]};
        #$space -= length $interssr}};
    $i++;}
  continue
    {print OUT "$id\t$count_seq\t$ssrtype\t$ssrseq\t",($end - $start +
1),"\t$start\t$end\n"};};
close (OUT);
open (OUT,">$ARGV[0].statistics");
#§§§§§ INFO §§§§§#
#§§§ Specifications §§§#
print OUT "Specifications\n==============\n\nSequence source file:
\"$ARGV[0]\"\n\nDefinement of microsatellites (unit size / minimum number
of repeats):\n";
for ($i = 0; $i < scalar (@typ); $i++) {print OUT
"($typ[$i]/$typrep{$typ[$i]}) "};print OUT "\n";
if ($amb > 0) {print OUT "\nMaximal number of bases interrupting 2 SSRs in
a compound microsatellite:  $amb\n"};
print OUT "\n\n\n";
#§§§ OCCURRENCE OF SSRs §§§#
#small calculations
my @ssr_containing_seqs = values %ssr_containing_seqs;
my $ssr_containing_seqs = 0;
for ($i = 0; $i < scalar (@ssr_containing_seqs); $i++)
{$ssr_containing_seqs += $ssr_containing_seqs[$i]};
my @count_motifs = sort {length ($a) <=> length ($b) || $a cmp $b }
keys %count_motifs;
my @count_class = sort { $a <=> $b } keys %count_class;
for ($i = 0; $i < scalar (@count_class); $i++) {$total +=
$count_class{$count_class[$i]}};
#§§§ Overview §§§#
print OUT "RESULTS OF MICROSATELLITE
SEARCH\n============================\n\n";
```

```
print OUT "Total number of sequences examined:
$number_sequences\n";
print OUT "Total size of examined sequences (bp):
$size_sequences\n";
print OUT "Total number of identified SSRs:                 $total\n";
print OUT "Number of SSR containing sequences:
$ssr_containing_seqs\n";
print OUT "Number of sequences containing more than 1 SSR:
",$ssr_containing_seqs - ($ssr_containing_seqs{1} || 0),"\n";
print OUT "Number of SSRs present in compound formation:
$ssr_in_compound\n\n\n";
#§§§ Frequency of SSR classes §§§#
print OUT "Distribution to different repeat type classes\n-----------------
----------------------------\n\n";
print OUT "Unit size\tNumber of SSRs\n";
my $total = undef;
for ($i = 0; $i < scalar (@count_class); $i++) {print OUT
"$count_class[$i]\t$count_class{$count_class[$i]}\n"};
print OUT "\n";
#§§§ Frequency of SSRs: per motif and number of repeats §§§#
print OUT "Frequency of identified SSR motifs\n----------------------------
------\n\nRepeats";
for ($i = $min_repeats;$i <= $max_repeats; $i++) {print OUT "\t$i"};
print OUT "\ttotal\n";
for ($i = 0; $i < scalar (@count_motifs); $i++)
  {my $typ = length ($count_motifs[$i]);
  print OUT $count_motifs[$i];
  for ($j = $min_repeats; $j <= $max_repeats; $j++)
    {if ($j < $typrep{$typ}) {print OUT "\t-";next};
    if ($count_motifs[$i]->{$j}) {print OUT "\t$count_motifs[$i]->{$j}"}
else {print OUT "\t"}; };
  print OUT "\t$count_motifs{$count_motifs[$i]}\n";};
print OUT "\n";
#§§§ Frequency of SSRs: summarizing redundant and reverse motifs §§§#
# Eliminates %count_motifs !
print OUT "Frequency of classified repeat types (considering sequence
complementary)\n-----------------------------------------------------------
--------------\n\nRepeats";
my (%red_rev,@red_rev); # groups
for ($i = 0; $i < scalar (@count_motifs); $i++)
  { next if ($count_motifs{$count_motifs[$i]} eq 'X');
  my (%group,@group,$red_rev); # store redundant/reverse motifs
```

```
  my $reverse_motif = $actual_motif = $actual_motif_a = $count_motifs[$i];
  $reverse_motif =~ tr/ACGT/TGCA/;
  my $reverse_motif_a = $reverse_motif;
  for ($j = 0; $j < length ($count_motifs[$i]); $j++)
  {if ($count_motifs{$actual_motif}) {$group{$actual_motif} = "1";
$count_motifs{$actual_motif}='X'};
    if ($count_motifs{$reverse_motif}) {$group{$reverse_motif} = "1";
$count_motifs{$reverse_motif}='X'};
    $actual_motif =~ s/(.)(.*)/$2$1/;
    $reverse_motif =~ s/(.)(.*)/$2$1/;
    $actual_motif_a = $actual_motif if ($actual_motif lt $actual_motif_a);
    $reverse_motif_a = $reverse_motif if ($reverse_motif lt
$reverse_motif_a) };
  if ($actual_motif_a lt $reverse_motif_a) {$red_rev =
"$actual_motif_a/$reverse_motif_a"}
  else {$red_rev = "$reverse_motif_a/$actual_motif_a"}; # group name
  $red_rev{$red_rev}++;
  @group = keys %group;
  for ($j = 0; $j < scalar (@group); $j++)
    {for ($k = $min_repeats; $k <= $max_repeats; $k++)
      { if ($group[$j]->{$k}) {$red_rev->{"total"} += $group[$j]-
>{$k};$red_rev->{$k} += $group[$j]->{$k}} } };
for ($i = $min_repeats; $i <= $max_repeats; $i++) {print OUT "\t$i"};
print OUT "\ttotal\n";
@red_rev = sort {length ($a) <=> length ($b) || $a cmp $b } keys %red_rev;
for ($i = 0; $i < scalar (@red_rev); $i++)
  {my $typ = (length ($red_rev[$i])-1)/2;
  print OUT $red_rev[$i];
  for ($j = $min_repeats; $j <= $max_repeats; $j++)
    {if ($j < $typrep{$typ}) {print OUT "\t-";next};
    if ($red_rev[$i]->{$j}) {print OUT "\t",$red_rev[$i]->{$j}}
    else {print OUT "\t"}};
  print OUT "\t",$red_rev[$i]->{"total"},"\n"; };
```

3. Mã chương trình ssrfinder_1_0

❖ 1_ssr_repeat_finder.pl

```
#!/usr/bin/perl -w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
$runtype = 1; # 1 = genbank fasta, 0 = local (fasta header differences)
```

```
# no need to change anything below this point
$seqcount = 0;
$ssr_count = 0;
@ssr_label = qw(a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab
ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az);
# set the flanking length here ( in bases )
$flank_length = 150;  # i.e. 150 bp
# set the minimum length of the repeat
$min_pattern_length = 12;  # i.e. 12 bp
# open the input sequence file - fasta format 3rd field of header is the
accession number
open (SEQFILE, "../$datename/sequence$datename.txt") || die "file not found:
$!";
# open the output file for the sequence ids
open (IDFILE, ">>../$datename/new_ids$datename.txt") || die "couldn't
create file";
# open the ouput file for the ssr results
open (SSROUTFILE, ">>../$datename/ssrout$datename.txt") || die "couldn't
create file";
#open the output file for the overall output for loading into labdb
open (LABDBTXT, ">>../$datename/labdbout$datename.txt") || die "couldn't
create file";
# read in file of previously checked IDs
$CheckedIDs = `cat ../$datename/CheckedIDs.txt`;
# parse the sequence file and process
if ($runtype == 1)  {
  while (defined ($line = <SEQFILE>))  {
    chomp $line;
    if ($line =~ /^>/)  {
      if (defined ($Seq)) {
      # check if genbank id has been done before
        if ($CheckedIDs !~ /$SeqHead[3]/)  {
          &SSRSearch;
          print "new seq\n";
        } else {
          print "old seq\n";} }
      $seqcount++;
      print $seqcount;
      $HeadLine = $line;
      undef $Seq;
      @SeqHead = split(/\|/,$HeadLine);
      print IDFILE "$SeqHead[3]\t$SeqHead[4]\n";
```

```
    } else  {
      $Seq = "$Seq" . "$line";  }
  if (defined ($Seq))  {
  # check if genbank id has been done before
    if ($CheckedIDs !~ /$SeqHead[3]/)  {
      &SSRSearch;
      print "new seq\n";
    } else {
      print "old seq\n"; } }
} elsif ($runtype == 2) {
  while (defined ($line = <SEQFILE>))  {
    chomp ($line);
    @LineIn = split(/\t/, $line);
    $SeqHead[3] = $LineIn[0];
    print $SeqHead[3];
    $Seq = $LineIn[1];
    if (defined ($Seq)) {
      &SSRSearch;}
    $seqcount++;
    print $seqcount;
    if ( ($seqcount%5) == 0 ) {
      print "\n";
    } else  {
      print "\t"; }
    undef $Seq;
    @SeqHead = split(/\|/,$HeadLine) }
} elsif ($runtype == 3) {
  while (defined ($line = <SEQFILE>))  {
    chomp ($line);
    if ($line =~ /^>/)  {
      if (defined ($Seq)) {
        &SSRSearch;
      } $seqcount++;
      print $seqcount;
      if ( ($seqcount%5) == 0 ) {
        print "\n";
      } else  {
        print "\t";}
      undef $Seq;
      $HeadLine = $line;
      @templine1 = split(/>/,$HeadLine);
      @templine2 = split(/\./, $templine1[1]);
```

```
      $SeqHead[3] = $templine2[0]."_".$templine2[3];
      print IDFILE "$SeqHead[3]\n";
    } else  {
      $Seq = "$Seq" . "$line"; }
  if (defined ($Seq))  {
    &SSRSearch; }
close (SEQFILE);
close (IDFILE);
close (SSROUTFILE);
close (LABDBTXT);
print "Number of sequences in input file = $seqcount \n";
print "Number of Repeats found = $ssr_count \n";
exit 0;
# subroutines
sub SSRSearch()  {
  print "*";
  $suffix = -1;
  while ( $Seq =~ /(([ATGC]{2,})\2{3,})/gi ) {
    print "+";
    $fullmatch = $1;
    $minmatch = $2;
# minimum matches: di-nt 6 repeats, tri-nt 4 repeats, tetra-nt 4 repeats
# check that repeat is greater than minimum total length of match
# and that it is not a single nt repeat - AAAAAAA, TTTTTTTT, etc.
    if ( (( $length_sub = length($fullmatch)) >= $min_pattern_length )
      && !( $fullmatch =~ /([ATGC])\1{8,}/ ) )  {
      print "-";
      $ssr_count++;
      $suffix++;
      $Accession = "$SeqHead[3]" . "$ssr_label[$suffix]";
      print SSROUTFILE "$Accession\t$fullmatch\t$minmatch\t";
      print LABDBTXT "$SeqHead[3]\t$Accession\t$fullmatch\t$minmatch\t";
      $pos2 = index($Seq,$fullmatch);
      $pos3 = $pos2 + $length_sub;
      if ($pos2 < $flank_length)  {
        $pos1 = 0;
      } else  {
        $pos1 = $pos2 - $flank_length; }
      if ( ( ( $max = length($Seq)) - $pos3 ) < $flank_length )  {
        $pos4 = $max;
      } else  {
        $pos4 = $pos3 + $flank_length; }
```

```
      $seqleft = substr($Seq,$pos1,$pos2-$pos1);
      $seqcenter = substr($Seq,$pos2,$pos3-$pos2);
      $seqright = substr($Seq,$pos3,$pos4-$pos3);
      print SSROUTFILE $seqleft, "[", $seqcenter,"]",$seqright,"\n";
      print LABDBTXT $pos2,",",$length_sub,"\t",$seqleft, "[",
$seqcenter,"]",$seqright,"\n";  }
  print "\n";}
```

❖ 2_ssr_primer_designer.pl

```
#!/usr/bin/perl -w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
$primer3app = 'd:\\detai\\ssrfinder_1_0\\primer3'; # full path to the
primer3 command
# no need to change anything below this point
# open the ouput file with the ssr results
open (SSRINFILE, "../$datename/ssrout$datename.txt") || die "couldn't open
file";
open (RAWPRIMER, ">../$datename/raw_primer3$datename.out") || die "couldn't
open file: $!";
open (PRIMEROUT, ">../$datename/primer_results$datename.txt") || die
"couldn't open file: $!";
$counter = 0;
while (defined($line = <SSRINFILE>))  {
  chomp ($line);
  @columns = split(/\t/,$line);
  @seq = split(/\[|\]/,$columns[3]);
  $length_left = length($seq[0]);
  $length_mask = length($seq[1]);
  $length_right = length($seq[2]);
  $t1 = $length_left+1;
  $t2 = $length_mask;
  open (PRIMERIN, ">../$datename/primerin.txt") || die "couldn't open file:
$!";
  print PRIMERIN "PRIMER_SEQUENCE_ID=",$columns[1],"\n";
  print PRIMERIN "SEQUENCE=",$seq[0],$seq[1],$seq[2],"\n";
  print PRIMERIN "TARGET=", $t1, ",", $t2, "\n";
  print PRIMERIN "PRIMER_PRODUCT_SIZE_RANGE=80-160 80-240 80-300\n";
  print PRIMERIN "PRIMER_OPT_SIZE=24\n";
  print PRIMERIN "PRIMER_MIN_SIZE=20\n";
  print PRIMERIN "PRIMER_MAX_SIZE=28\n";
  print PRIMERIN "PRIMER_OPT_TM=63\n";
```

```
  print PRIMERIN "PRIMER_MIN_TM=60\n";
  print PRIMERIN "PRIMER_MAX_TM=65\n";
  print PRIMERIN "PRIMER_MAX_DIFF_TM=1\n";
  print PRIMERIN "=\n";
  close (PRIMERIN);
#  $primer3 = `../devel/primer/primer3_0_9_test/src/primer3_core
< ../$datename/primerin.txt`;
  $primer3 = `$primer3app < ../$datename/primerin.txt`;
  print RAWPRIMER "######## $columns[1] #########\n";
  print RAWPRIMER $primer3, "\n";
  @prime_out = split(/\n/, $primer3);
  foreach $i (0..$#prime_out)  {
    ($varname,$varvalue) = split(/=/, $prime_out[$i]);
    $primehash{$varname} = $varvalue;}
  if ($primehash{'PRIMER_LEFT_SEQUENCE'})  {
    $counter++;
    print PRIMEROUT "$columns[0]\t$columns[1]\t$columns[2]";
    print PRIMEROUT "\t$seq[0]$seq[1]$seq[2]";
    print PRIMEROUT "\t", $primehash{'PRIMER_LEFT_SEQUENCE'};
    print PRIMEROUT "\t", $primehash{'PRIMER_LEFT_TM'};
    print PRIMEROUT "\t", $primehash{'PRIMER_RIGHT_SEQUENCE'};
    print PRIMEROUT "\t", $primehash{'PRIMER_RIGHT_TM'};
    print PRIMEROUT "\t", $primehash{'PRIMER_PRODUCT_SIZE'};
    print PRIMEROUT "\n";  }
  undef %primehash;}
close (PRIMEROUT);
close (SSRINFILE);
close (RAWPRIMER);
print "repeats primed: ", $counter, "\n";
exit 0;
```

❖ 3_ssr_primer_rep_check.pl

```
#!/usr/bin/perl -w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
# no need to change anything below this point
open (OUTFILE, ">rescreened$datename.txt");
$good = 0;
$bad = 0;
$temp = `type primer_results$datename.txt`;
@temp1 = split(/\n/, $temp);
foreach $i (0..$#temp1) {
```

```
  @temp2 = split(/\t/, $temp1[$i]);
  print $temp2[0],"\t",$temp2[1],"\t",$temp2[2],"\t",$temp2[3];
  print
"\t",$temp2[4],"\t",$temp2[5],"\t",$temp2[6],"\t",$temp2[7],"\t",$temp2[8],
"\t";
  if (( $temp2[4] =~ /(([ATGC]{2,3})\2{3,})/gi ) || ( $temp2[6] =~
/(([ATGC]{2,3})\2{3,})/gi )) {
    print "bad\n";
    $bad++;
  } else {
    print "good\n";
    $good++;
    print OUTFILE
$temp2[0],"\t",$temp2[1],"\t",$temp2[2],"\t",$temp2[3],"\t";
    print OUTFILE
$temp2[4],"\t",$temp2[5],"\t",$temp2[6],"\t",$temp2[7],"\t",$temp2[8],"\n"  }
print "good: $good\nbad:  $bad\n";
close (OUTFILE);
exit(0);
```

❖ 4_ssr_primer_blast.pl

```
#!/usr/bin/perl -w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
$blastapp = 'd:\\detai\\ssrfinder_1_0\\blastall'; # full path to the
blastall command
$blastdbdir = 'd:\\detai\\ssrfinder_1_0\\db'; #full path to the blast
database directory
$blastdbname = 'AllPrimers.nt'; # name of the blast database to use
$formatdbapp = 'd:\\detai\\ssrfinder_1_0\\formatdb'; # full path to the
formatdb command
# no need to change anything below this point
open (PRIMERSIN, "../$datename/rescreened$datename.txt") || die;
open (BLASTOUTFILE, ">../$datename/blastout$datename.txt") || die;
open (FULLBLASTOUT, ">../$datename/fullblastoutput$datename.txt") || die;
$counter = 0;
$b0hit = 0;
while (defined($line=<PRIMERSIN>))  {
  chomp ($line);
  @columns = split(/\t/,$line);
  $Accession = $columns[0];
  $sequence = $columns[3];
```

```
  $forward = $columns[4];
  $reverse = $columns[6];
  $counter++;
  print $counter, "\n";
  print BLASTOUTFILE $Accession, "\t", $columns[1], "\t", $columns[2];
  print BLASTOUTFILE "\t", $sequence, "\t", $forward,  "\t", $columns[5],
"\t";
  print BLASTOUTFILE $reverse, "\t", $columns[7], "\t", $columns[8], "\t";
  &BlastIt;
print "Blasted Sequences= ",$counter,"\n";
print "Blast NON-Hits= ",$b0hit,"\n";
close (PRIMERSIN);
close (BLASTOUTFILE);
close (FULLBLASTOUT);
exit 0;
sub BlastIt()  {
      open (TMPSEQFILE, ">../$datename/repeats/$Accession.fasta");
      print TMPSEQFILE "> ",$Accession,"\n",$sequence,"\n";
      close (TMPSEQFILE);
#     $blastout = `blastall -p blastn -d db/AllPrimers.nt -e 0.01 -
i ../$datename/repeats/$Accession.fasta`;
      $blastout = `$blastapp -p blastn -d $blastdbdir/$blastdbname -e 0.01
-i ../$datename/repeats/$Accession.fasta`;
      print FULLBLASTOUT
"XXXXX\t",$Accession,"\tXXXXXXXXXXXXXXXXXXXXXXXXX\n\n";
      print FULLBLASTOUT $blastout;
      &ParseBlast;
sub DBup()  {
#     open (BLASTDBFASTA, ">>../blast/db/AllPrimers.nt");
      open (BLASTDBFASTA, ">>$blastdbdir/$blastdbname");
      print BLASTDBFASTA "> ",$Accession,"_f\n",$forward,"\n";
      print BLASTDBFASTA "> ",$Accession,"_r\n",$reverse,"\n";
      close (BLASTDBFASTA);
#     $formatdbstatus = system("formatdb -i db/AllPrimers.nt -p F -o T");
      $formatdbstatus = system("$formatdbapp -i $blastdbdir/$blastdbname -p
F -o T");}
sub ParseBlast()  {
# split the blast output into records (splits at blank lines)
  @blastrecs = split(/\n\n/, $blastout);
# the record that contains the hit results SHOULD be in record 6 - WATCH
OUT
  if ($blastrecs[5] =~ /Sequence/)  {
```

```
    @blastmatch = split(/\n/, $blastrecs[6]);
    print BLASTOUTFILE $#blastmatch-1, "\t";
    for $entry (0 .. $#blastmatch)  {
      ($name,$score, $E) = split(/[ ]+ /, $blastmatch[$entry]);
      print $name,"\t",$score,"\t",$E,"\n";
      print BLASTOUTFILE "\t",$name,"\t",$score,"\t",$E;}
  } else  {
    print BLASTOUTFILE "0\t";
    $b0hit++;
    print "No Hits found\n";
    print BLASTOUTFILE "\tnone\tnull\tnull";
    &DBup; }
  print BLASTOUTFILE "\n";}
```

❖ 5_ssr_order_filter.pl

```
#!/usr/bin/perl –w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
# no need to change anything below this point
open (OUTFILE, ">filter$datename.txt");
$infile = `cat blastout$datename.txt`;
$count = 0;
@temp1 = split(/\n/, $infile);
foreach $f (0..$#temp1) {
  @temp2 = split(/\t/, $temp1[$f]);
  if ($temp2[11] eq 'none') {
    print OUTFILE $temp2[0],"\t",$temp2[1],"\t";
    print OUTFILE $temp2[2],"\t",$temp2[3],"\t";
    print OUTFILE $temp2[4],"\t",$temp2[5],"\t";
    print OUTFILE $temp2[6],"\t",$temp2[7],"\t";
    print OUTFILE $temp2[8],"\n";
    print $temp2[0],"\t",$temp2[10],"\t",$temp2[11],"\n";
    $count++;}
print "non-hits: ",$count,"\n";
close (OUTFILE);
exit (0);
```

❖ 6_ssr_order_formatter.pl

```
#!/usr/bin/perl –w
# change these parameters for each run!!!!!!!!
$datename = '20060715'; # date for directory name and datafile info
```

```
# no need to change anything below this point
open (OUTFILE, ">order$datename.txt");
$infile = `cat blastout$datename.txt`;
$count = 0;
@temp1 = split(/\n/, $infile);
foreach $f (0..$#temp1) {
  @temp2 = split(/\t/, $temp1[$f]);
  if ($temp2[11] eq 'none') {
    print OUTFILE $temp2[0],"\t";
    print OUTFILE $temp2[4],"\t",$temp2[5],"\t";
    print OUTFILE $temp2[6],"\t",$temp2[7],"\t";
    print OUTFILE $temp2[8],"\n";
    print $temp2[0],"\t",$temp2[10],"\t",$temp2[11],"\n";
    $count++; }}
print "non-hits: ",$count,"\n";
close (OUTFILE);
exit (0);
```