ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ KIM THƯ

PHÁT TRIỂN MÔ HÌNH THAY THẾ AXIT AMIN CHO DỮ LIỆU HỆ GEN

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

HÀ NỘI - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ KIM THƯ

PHÁT TRIỂN MÔ HÌNH THAY THẾ AXIT AMIN CHO DỮ LIỆU HỆ GEN

Ngành đào tạo: Khoa học Máy tính
Mã số: 9480101

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. Lê Sỹ Vinh

HÀ NỘI – 2024

MỤC LỤC

Danh mục hình vẽ 5

Danh mục bảng 7

Danh mục ký hiệu và từ viết tắt 9

Danh mục ký hiệu và từ viết tắt 10

Lời cam đoan 11

Lời cảm ơn 12

Tóm tắt 13

Mở đầu 14

1. Đặt vấn đề 14

2. Động lực nghiên cứu 15

2.1 Mục đích nghiên cứu 15

2.2 Đối tượng nghiên cứu 15

2.3 Phạm vi nghiên cứu 15

2.4 Phương pháp nghiên cứu 15

3. Đóng góp chính của luận án 16

4. Bố cục của luận án 16

Chương 1: Cơ sở lý thuyết 18

1.1 Các khái niệm cơ bản 18

1.1.1 Thông tin di truyền 18

1.1.2 Các biến đổi di truyền 22

1.1.3 Mô hình thay thế nucleotit/axit amin 23

1.1.4 Sắp hàng đa trình tự tương đồng 29

1.1.5 Cây phân loài 30
1.1.6 Xây dựng cây phân loài bằng phương pháp cực đại khả năng 32
1.2 Bài toán ước lượng mô hình thay thế axit amin.....35
1.2.1 Bài toán 36
1.2.2 Các phương pháp ước lượng mô hình thay thế.....37
1.3 Tính không đồng nhất của quá trình biến đổi tại các vị trí khác nhau.....39
1.3.1 Mô hình tốc độ biến đổi 40
1.3.2 Mô hình đa ma trận 42
1.3.3 Mô hình lược đồ phân vùng.....43
1.3.4 Một số thuật toán phân hoạch sắp hàng 45
1.4 Tốc độ tiến hóa tại mỗi vị trí trên sắp hàng 51
1.5 Các phương pháp đánh giá mô hình 54
1.5.1 So sánh giá trị khả năng của cây phân loài xây dựng bằng phương pháp cực đại khả năng 54
1.5.2 Các độ đo AIC và BIC 55
1.5.3 So sánh cấu trúc cây.....56
1.6 Kết luận 58
Chương 2: Mô hình thay thế axit amin FLAVI cho Flavivirus 60
2.1 Giới thiệu.....60
2.2 Phương pháp 61
2.3 Thực nghiệm 63
2.3.1 Dữ liệu.....63
2.3.2 Tham số cài đặt 65

2.3.3 Thực nghiệm 68
2.4 Kết quả 69
2.4.1 Tính bền vững của mô hình 69
2.4.2 Phân tích và đánh giá mô hình 69
2.4.3 So sánh hiệu quả của FLAVI 74
2.5 Kết luận 78
Chương 3: Phương pháp phân hoạch sắp hàng sử dụng mô hình tiến hóa 80
3.1 Giới thiệu 80
3.2 Phương pháp 81
3.3 Thực nghiệm 87
3.3.1 Dữ liệu 87
3.3.2 Thực nghiệm 91
3.4 Kết quả 92
3.4.1 Dữ liệu DNA mô phỏng 92
3.4.2 Dữ liệu DNA thực 95
3.4.3 Dữ liệu protein thực 101
3.5 Kết luận 104
Chương 4: Phương pháp phân hoạch sắp hàng cho dữ liệu hệ gen 106
4.1 Mở đầu 106
4.2 Phương pháp 108
4.2.1 Thuật toán ước lượng nhanh tốc độ tiến hóa 108
4.2.2 gPartition 111
4.3 Thực nghiệm 113

4.3.1 Dữ liệu.....113
4.3.2 Tham số cài đặt.....114
4.3.3 Thực nghiệm.....114
4.4 Kết quả.....115
4.5 Kết luận.....119
Kết luận.....121
Tài liệu tham khảo.....124

Danh mục hình vẽ

Hình 1. 1 Ví dụ minh họa cây phát sinh loài bò sát. 31
Hình 1. 2 Cây có một đỉnh gốc và hai lá.....34
Hình 1. 3 Sơ đồ các bước ước lượng mô hình thay thế axit amin bằng phương pháp cực đại khả năng.....39
Hình 1. 4 Hàm mật độ xác suất (pdf) của phân phối gamma với tham số α khác nhau 41
Hình 1. 5 Ví dụ về thuật toán k-means lặp tìm lược đồ phân vùng tự động [30] 49
Hình 1. 6 Hai cây phân loài không gốc có cấu trúc khác nhau trên cùng tập taxa ...58
Hình 2. 1: Cấu trúc virion và bộ gen của vi rút Zika [98].....67
Hình 2. 2 Tần số của 20 axit amin trong ba mô hình FLAVI, LG và HIVb.....71
Hình 2. 3 Ma trận hệ số hoán đổi của ba mô hình FLAVI, HIVb và LG. Vòng tròn màu đen, xám và trắng tại hàng X, cột Y thể hiện hệ số hoán đổi giữa axit amin X và Y của mô hình FLAVI, LG và HIVb 72
Hình 2. 4 So sánh tương quan các hệ số hoán đổi giữa FLAVI với HIVb (Hình a) và LG (Hình b). Các hình tròn hiển thị sự khác biệt tương đối giữa hệ số hoán đổi trong FLAVI với X (HIVb hoặc LG) giá trị được tính bằng (FLAVI_XY – HIVb_XY)/(FLAVI_XY + HIVb_XY) . Các hình tròn màu đen thể hiện hệ số của FLAVI lớn hơn X, màu trắng thể hiện hệ số của X lớn hơn FLAVI. Giá trị 1/3 hoặc 2/3 có nghĩa hệ số của FLAVI lớn hơn X 2 hoặc 5 lần. Giá trị -1/3 hoặc -2/3 có nghĩa hệ số của X lớn hơn FLAVI 2 hoặc 5 lần. 73
Hình 2. 5 Kết quả so sánh cây ML của 30 sắp hàng Dengue, West Nile và Zika trong tập dữ liệu kiểm tra bằng thuật toán AU. Cây Tốt hơn là cây có giá trị khả năng cao hơn, nhưng không thực sự đáng kể theo thuật toán kiểm tra AU; cây Tốt hơn đáng kể là cây có giá trị khả năng cao hơn thực sự đáng kể theo thuật toán.....77
Hình 3. 1 Khoảng cách nRF trung bình giữa cây gốc và cây ML được xây dựng trên các bộ sắp hàng cùng tham số mô phỏng. AS là các bộ dữ liệu mô phỏng sử dụng cấu trúc cây bất đối xứng; SS là có bộ dữ liệu mô phỏng sử dụng cấu trúc cây đối xứng;

miss: các bộ có dữ liệu mất mát. RP2: kết quả khi sử dụng lược đồ tạo bởi RatePartition với $d = 2$.....93
Hình 3. 2 Số lượng phân vùng trong mỗi lược đồ phân vùng tạo bởi thuật toán mPartition, RP4 và RP5 96
Hình 3. 3 Sự phân bố các vị trí bất biến và vị trí có biến đổi trong các lược đồ tạo bởi phương pháp mPartition.....98
Hình 3. 4 Sự phân phối của các vị trí bất biến và vị trí có biến đổi trong các phân vùng của lược đồ phân vùng cho sắp hàng bọ cánh cứng thủy sinh Noteridae tạo bởi mPartition.....99
Hình 3. 5 Cây bootstrap của sắp hàng bọ cánh cứng thủy sinh Noteridae. Cây sử dụng lược đồ phân vùng thu được bằng phương pháp mPartition. Giá trị X và Y ở mỗi nhánh là giá trị độ tin cậy của nhánh sử dụng lược đồ phân vùng bằng phương pháp tham lam [31] (trong đó đó khoảng gen và vị trí nucleotit trong codon được định nghĩa sẵn) và mPartition..... 100
Hình 4. 1 Thời gian (giây) tính tốc độ tiến hóa cho mỗi vị trí bằng thuật toán TIGER và fastTIGER. 116
Hình 4. 2 Thời gian chạy (giây) của các thuật toán RatePartition, mPartition (dùng tốc độ TIGER) và gPartition (dùng fastTIGER).Thời gian chạy, đơn vị: giây 117

Danh mục bảng

Bảng 1. 1 Tên, viết tắt 3 ký tự và viết tắt 1 ký tự của 20 axit amin 20
Bảng 1. 2 Danh sách 64 codon và axit amin tương ứng 21
Bảng 1. 3 Ví dụ về các phép biến đổi trên hai trình tự tương đồng 22
Bảng 1. 4 Ví dụ về mô hình thay thế axit amin (mô hình LG) 28
Bảng 1. 5 Ví dụ về sắp hàng đa trình tự tương đồng 29
Bảng 1. 6 Số lượng cây không có gốc tương ứng với số trình tự trong sắp hàng.....32
Bảng 1. 7 Danh sách phân nhánh trên hai cây trong Hình 1. 6.....58
Bảng 2. 1 Tổng quan về dữ liệu được sử dụng 67
Bảng 2. 2 Hệ số tương quan Pearson giữa FLAVI và các mô hình khác. Giá trị ở nửa trên là hệ số tương quan của ma trận hệ số hóa đổi, nửa dưới là tương quang của các vectơ tần số.....70
Bảng 2. 3 Sự khác nhau trên từng phần tử trong ma trận hoán đổi. Giá trị ở hàng "gấp đôi", cột "FLAVI>HIVb" là số hệ số trang ma trận FLAVI lớn hơn hoặc bằng hai lần hệ số ở vị trí tương ứng trong ma trận HIVb. Các giá trị khác trong bảng có ý nghĩa tương tự. 74
Bảng 2. 4 Kết quả so sánh giá trị khả năng của các cây phân loài được xây dựng bằng tám mô hình.....74
Bảng 2. 5 Bảng so sánh giá trị log-likelihood và AIC trung bình tưng ứng với các mô hình trên 30 cây cực đại khả năng trong dữ liệu kiểm tra. Lưu ý: giá trị log-likelihood cao hơn hoặc AIC nhỏ hơn tương ứng với mô hình tốt hơn. 75
Bảng 2. 6 Mô hình thay thế axit amin cho Flavivirus.....79
Bảng 3. 1 Tám bộ dữ liệu DNA thực dùng để so sánh các phương pháp phân hoạch sắp hàng.....88
Bảng 3. 2 Sắp hàng protein thực sử dụng để so sánh các lược đồ phân vùng 90
Bảng 3. 3 Trung bình khoảng cách nRF giữa cây gốc với cây ML thu được khi sử dụng các lược đồ phân vùng khác nhau trên dữ liệu mô phỏng 92

Bảng 3. 4 Giá trị AICc và BIC của tám sắp hàng DNA thực khi sử dụng các lược đồ phân vùng khác nhau. Giá trị AICc (BIC) nhỏ hơn tương ứng với lược đồ phân vùng tốt hơn; giá trị tốt nhất được in đậm. Ký hiệu: NP: Không phân vùng; RP4 (RP5): RatePartition với hệ số phân chia $d = 4$ ($d = 5$); mPar: mPartition 94
Bảng 3. 5 Trung bình khoảng cách nRF trên các cặp cây xây dựng bằng các lược đồ phân vùng khác nhau.....96
Bảng 3. 6 Giá trị AICc của các cây được xây dựng sử dụng lược đồ tạo bởi mPartition và RatePartition trên các sắp hàng protein. RP2, RP3, RP4, và RP5 là RatePartition với $d = 2, 3, 4$ và 5. 102
Bảng 3. 7 Giá trị BIC của các cây được xây dựng sử dụng lược đồ tạo bởi mPartition và RatePartition trên các sắp hàng protein. RP2, RP3, RP4, và RP5 là RatePartition với $d = 2, 3, 4$ và 5. 103
Bảng 3. 8 Số lượng vị trí bất biến (Inv) và vị trí biến đổi (Var) trong các lược đồ của bốn sắp hàng Irissari..... 104
Bảng 4. 1 Tám bộ dữ liệu dùng trong thực nghiệm 113
Bảng 4. 2 Hệ số tương quan giữa tốc độ tiến hóa ước lượng được bằng fastTIGER và TIGER trong năm bộ dữ liệu thực nghiệm. 116
Bảng 4. 3 Giá trị AICc trung bình trên mỗi vị trí của cây ML sử dụng ba cách phân hoạch vị trí. Giá trị tốt nhất được tô đậm để đánh dấu..... 119

Danh mục ký hiệu và từ viết tắt

q_{ij}	*Hệ số thay thế tức thì giữa axit amin (nucleotit) i và j*
π_i	*Tần số của axit amin (nucleotit) i*
r_{ij}	*Hệ số hoán đổi giữa hai axit amin (nucleotit) i và j*
α	*Tham số hình dạng phân phối gamma*
D	*Một sắp hàng đa trình tự tương đồng*
n	*Số trình tự trong một sắp hàng*
l	*Độ dài của sắp hàng*
Q	*Ma trận tốc độ thay thế thức thì*
Π	*Vectơ tần số của các trạng thái (nucleotit /axit amin)*
R	*Ma trận hệ số hoán đổi*
T	*Cây phát sinh hoài*
V	*Mô hình tốc độ biến đổi*
D	*Một tập sắp hàng đa trình tự tương đồng*
N	*Số sắp hàng trong một tập sắp hàng*
T	*Tập cây tương ứng với các sắp hàng trong một tập sắp hàng*
M	*Tập mô hình thay thế axit amin*
PS	*Lược đồ phân vùng cho một sắp hàng*
S_i	*Tập con thứ i trong lược đồ phân vùng*
S_{ij}	*Vị trí thứ j trong tập con thứ I của một lược đồ phân vùng*
$r(d_j)$	*Tốc độ biến đổi tại vị trí thứ j*
d	*Hệ số phân chia trong thuật toán RatePartition*

Danh mục ký hiệu và từ viết tắt

DNA	*Deoxyribonucleic acid*	*Axit deoxyribonucleic*
RNA	*Ribonucleic acid*	*Axit ribonucleic*
RF	*Robinson-Fould distance*	*Khoảng cách Robinson-Fould*
nRF	*Normalised Robinson-Fould distance*	*Khoảng cách Robinson-Fould được chuẩn hóa*
ML	*Maximum likelihood*	*Cực đại khả năng*
AIC	*Akaike information criterion*	*Độ đo lý thuyết thông tin Akaike*
AICc	*Corrected akaike information criterion*	*Độ đo lý thuyết thông tin Akaike đã điều chỉnh*
BIC	*Bayesian information criterion*	*Độ đo lý thuyết thông tin Bayesian*
RP	*RatePartition*	*Thuật toán RatePartition*

Lời cam đoan

Tôi cam đoan đây là công trình nghiên cứu do tôi thực hiện dưới sự hướng dẫn của PGS.TS. Lê Sỹ Vinh tại bộ môn Khoa học Máy tính, Khoa Công nghệ Thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội. Trừ những phần nội dung đã được chỉ rõ, các số liệu kết quả trình bày trong luận án là trung thực và chưa được công bố bởi bất kỳ tác giả nào hay ở bất kỳ công trình nào khác.

Tác giả.

Lời cảm ơn

Trước tiên, tôi xin gửi lời cảm ơn sâu sắc tới PGS. TS. Lê Sỹ Vinh, người đã tận tình hướng dẫn, chỉ bảo, động viên và cho tôi cơ hội được đối mặt với rất nhiều khó khăn trong quá trình nghiên cứu. Cảm ơn Thầy đã nghiêm khắc nhắc nhở, phê bình để em làm việc nghiêm túc và tiến lên phía trước.

Tôi xin chân thành cảm ơn thầy Đặng Cao Cường và cô Hoàng Thị Điệp đã giúp đỡ tôi làm quen với lĩnh vực nghiên cứu và đưa ra những nhận xét đáng giá, giúp ích cho tôi rất nhiều trong quá trình nghiên cứu.

Tôi xin chân thành cảm ơn các thầy cô giáo khoa Công nghệ Thông tin, trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, đặc biệt là các thầy cô trong bộ môn Khoa học Máy tính và bộ môn Khoa học và Kỹ thuật tính toán đã tận tình đào tạo, cung cấp cho tôi những kiến thức vô cùng quý giá; đã tạo điều kiện cho tôi về môi trường làm việc trong quá trình học tập, nghiên cứu tại Trường.

Bên cạnh đó, tôi xin chân thành cảm ơn các đồng nghiệp trong Viện Toán ứng dụng và Tin học, Đại học Bách Khoa Hà Nội đã tạo điều kiện về thời gian và nhiệm vụ cho tôi trong suốt quá trình làm nghiên cứu sinh.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc tới gia đình và bạn bè, đặc biệt là chồng tôi Vũ Ngọc Hiếu, những người đã cho tôi điểm tựa vững chắc để tôi có thể đạt được kết quả như ngày hôm nay.

Tóm tắt

Sự xuất hiện của dữ liệu di truyền và các loại dữ liệu sinh học khác đã giúp xây dựng cây phân loài chính xác hơn so việc chỉ sử dụng thông tin hình thái, tuy nhiên việc phân tích các tập dữ liệu lớn và phức tạp đòi hỏi các công cụ, thuật toán tính toán hiệu quả và đặc biệt là cần mô hình biểu diễn dữ liệu phù hợp.

Luận án tập trung nghiên cứu tối ưu mô hình tiến hóa theo hai hướng: xây dựng mô hình thay thế axit amin cho vi rút Flavivirus từ dữ liệu hệ gen và xây dựng thuật toán phân hoạch tập vị trí để biểu diễn tính không đồng nhất về tốc độ tiến hóa giữa các vị trí trên hệ gen.

Mô hình thay thế axit amin FLAVI được ước lượng trên bộ dữ liệu gồm ba vi rút Dengue, West Nile và Zika, là những vi rút gây ra dịch bệnh nghiêm trọng thời gian gần đây. Mô hình cho kết quả tốt hơn các mô hình hiện có khi sử dụng cho dữ liệu vi rút thuộc chi Flavivirus.

Để mô hình hóa tính không đồng nhất về tốc độ tiến hóa giữa các vị trí trên hệ gen, luận án đề xuất hai thuật toán phân hoạch sắp hàng là mPartition và gPartition. Với một sắp hàng đầu vào, các thuật toán tính toán đặc trưng của từng vị trí (tốc độ tiến hóa và mô hình thay thế phù hợp nhất) để gom những vị trí có quá trình tiến hóa tương đồng vào cùng một nhóm.

Kết quả thực nghiệm cho thấy cả hai thuận toán đều tốt hơn thuật toán phân hoạch hiện có chỉ sử dụng tốc độ tiến hóa. mParition cho kết quả tốt và có thể sử dụng cho dữ liệu trình tự bất kỳ, bao gồm cả DNA hoặc protein; tuy nhiên thuật toán chưa áp dụng được cho dữ liệu lớn cỡ hệ gen. Thuật toán gPartition có thể phân hoạch sắp hàng cỡ hệ gen trong thời gian ngắn. Sử dụng lược đồ phân vùng tạo bởi các thuật toán giúp biểu diễn chính xác hơn quá trình tiến hóa trong dữ liệu, qua đó nâng cao tính đúng đắn của cây phân loài cực đại khả năng xây dựng được.

Mở đầu

1. Đặt vấn đề

Học thuyết Tiến hóa của Darwin ra đời nghiên cứu về mối quan hệ tiến hóa (phát sinh loài) giữa các loài sinh vật. Câu hỏi đặt ra là làm sao xác định mối quan hệ này hay nói cách khác là xác định cây phân loài (hay còn gọi là cây phân loài) trên một nhóm loài cho trước. Để xây dựng cây phân loài, hầu hết các phương pháp tin sinh học đều sử dụng một hoặc một vài mô hình tiến hóa để tính khoảng cách tiến hóa di truyền giữa các trình tự, hoặc tính giá trị hợp lý của cây thông qua các mô hình xác suất. Do quá trình tiến hóa của mỗi loài sinh vật có thể có nhiều đặc điểm riêng, việc lựa chọn các mô hình tiến hóa phù hợp với dữ liệu là một trong những yếu tố có tác động lớn tới độ chính xác của kết quả. Nhiều nghiên cứu đã chỉ ra rằng việc lựa chọn mô hình tiến hóa không phù hợp có thể gây sai lệch lớn tới cây phân loài mà chúng ta xây dựng.

Ngày nay, nhờ có công nghệ giải trình tự thế hệ mới, hàng triệu đoạn ADN có thể được giải mã đồng thời khiến cho số lượng trình tự được giải mã tăng lên không ngừng, các bộ dữ liệu kích thước lớn đến rất lớn trở nên phổ biến. Kích thước dữ liệu tăng lên kéo theo một số thay đổi trong cách chọn lựa mô hình tiến hóa sử dụng để xây dựng cây phân loài. Trước đây các nhà khoa học thường sử dụng một trong số ít các mô hình tiến hóa sẵn có; các mô hình này có nhiều ràng buộc trên các tham số hơn và không hoàn toàn phù hợp để biểu diễn quá trình tiến hóa của các bộ dữ liệu lớn. Hiện tại, có nhiều mô hình tiến hóa mới được đề xuất, các mô hình này có khả năng biểu diễn tốt hơn quá trình tiến hóa của các nhóm dữ liệu có tính chuyên biệt. Bên cạnh đó, do khả năng tính toán của các loại máy tính tăng lên, các cách tiếp cận như sử dụng mô hình đa ma trận hoặc lược đồ phân vùng cũng giúp việc tính toán giá trị khả năng được chính xác hơn. Tối ưu các mô hình này góp phần tăng độ chính xác cho kết quả của bài toán xây dựng cây phân loài.

2. Động lực nghiên cứu

Mục đích, phạm vi, phương pháp nghiên cứu

2.1 Mục đích nghiên cứu

Mục đích nghiên cứu của luận án là trả lời câu hỏi chung:

1. *Biểu diễn quá trình tiến hóa trên dữ liệu lớn như thế nào để nâng cao tính đúng đắn của cây phân loài xây dựng được?*

Cụ thể, các nghiên cứu được tiến hành theo hai hướng tương ứng với các câu hỏi:

2. *Các mô hình tiến hóa chung có phù hợp để sử dụng cho bộ dữ liệu của một loài cụ thể không?*
3. *Chỉ sử dụng một mô hình tiến hóa cho toàn bộ dữ liệu có cho kết quả tốt không? Đặc biệt là với những bộ dữ liệu có kích thước lớn ?*

2.2 Đối tượng nghiên cứu

Dữ liệu sinh học phân tử kích thước lớn, gồm dữ liệu trình tự DNA và trình tự protein.

2.3 Phạm vi nghiên cứu

Việc lựa chọn mô hình tiến hóa để tính giá trị khả năng cho các cây ứng viên đóng vai trò quan trọng trong quá trình xây dựng cây phân loài bằng phương pháp cực đại khả năng - một trong những phương pháp xây dựng cây phân loài được sử dụng rộng rãi nhất. Luận án tập trung vào tối ưu cách biểu diễn tri thức về quá trình tiến hóa và xử lý tính không đồng nhất trong quá trình tiến hóa của dữ liệu; các đề xuất được đưa ra dưới giả thiết là mô hình có tính thuận nghịch về thời gian.

2.4 Phương pháp nghiên cứu

Luận án sử dụng phương pháp nghiên cứu tài liệu và làm thực nghiệm:

- NCS thực hiện nghiên cứu tổng quan lý thuyết và thu thập dữ liệu có liên quan tới nội dung nghiên cứu đã được các nhóm nghiên cứu khác công bố

- Đề xuất các phương pháp mới để trả lời cho câu hỏi nghiên cứu và làm thực nghiệm trên dữ liệu đã thu thập.
- Phân tích kết quả thực nghiệm và thay đổi mô hình theo hướng tối ưu nhất.

3. Đóng góp chính của luận án

Luận án đã đạt được một số kết quả chính và đóng góp như sau :

1. Luận án đề xuất một số mô hình thay thế axit amin cho virút trong chi Flavivirus và thực hiện phân tích, đánh giá tính hiệu quả của mô hình khi xây dựng cây phân loài so với các mô hình sẵn có.
   Các kết quả này được công bố trong công trình [CT1] và [CT4].
2. Đề xuất một số thuật toán để phân hoạch tập vị trí cho một bộ dữ liệu cho trước. Mỗi tập con của phân hoạch thể hiện một nhóm vị trí có quá trình tiến hóa tương đồng trên hệ gene và có thể được gán một mô hình tiến hóa khác nhau do vậy quá trình tiến hóa được biểu dẫn gần với thực tế hơn.
   Các kết quả này được công bố trong công trình [CT2], [CT3], [CT5] và [CT7]
3. Đề xuất một cách tính nhanh tốc độ biến đổi tương đối giữa các vị trí trong sắp hàng.
   Kết quả này được công bố trong công trình [CT6]

4. Bố cục của luận án

Ngoài phần Mở đầu và Kết luận, luận án gồm 4 chương, được tổ chức như sau:

Chương 1 giới thiệu các khái niệm cơ bản (bao gồm các loại trình tự; mô hình thay thế, sắp hàng đa trình tự tương đồng và cây phân loài) và một số kiến thức nền tảng quan trọng như quá trình xây dựng cây phân loài bằng phương pháp cực đại khả năng và hai bài toán liên quan tới mô hình hóa quá trình tiến hóa trên dữ liệu trình tự – cụ thể là bài toán xác định mô hình thay thế và bài toán biểu diễn tính không đồng nhất về biến đổi tại các vị trí khác nhau trên hệ gen. Chương 1 cũng giới thiệu các phương pháp đánh giá và so sánh các mô hình thường dùng.

Chương 2 trình bày quá trình ước lượng mô hình thay thế axit amin FLAVI cho chi Flavivirus và các thực nghiệm đã thực hiện để so sánh mô hình FLAVI với các mô hình hiện có, theo kết quả thực nghiệm, FLAVI giúp xây dựng cây tốt hơn, phù hợp để phân tích dữ liệu trình tự các virút thuộc chi Flavivirus.

Chương 3 trình bày thuật toán phân hoạch sắp hàng mPartition. Thuật toán tự động phân loại tập các vị trí trong một sắp hàng thành lược đồ phân vùng; mỗi phân vùng trong lược đồ chứa các vị trí có mô hình tiến hóa tương đồng trong khi các phân vùng khác nhau có thể tuân theo các mô hình tiến hóa khác nhau. mPartition thực hiện lặp đi lặp lại việc chia nhỏ tập vị trí ban đầu thành ba tập con cho đến khi việc phân chia không thu được lược đồ mới tốt hơn. Thực nghiệm cho thấy phương pháp này có thể áp dụng cho cả hai loại dữ liệu DNA, protein và cho kết quả tốt hơn các phương đã đề xuất.

Chương 5 trình bày thuật toán gPartition phân hoạch nhanh sắp hàng cỡ hệ gen. gPartition sử dụng thuật toán fastTIGER để ước lượng nhanh tốc độ biến đổi tại mỗi vị trí của sắp hàng. Thuật toán có thể chạy trên các sắp hàng có độ dài lên đến hàng triệu vị trí trong thời gian chấp nhận được (dưới 24 giờ đối với các bộ dữ liệu đã thử nghiệm). gPartition sử dụng kết hợp cả tốc độ biến đổi tại mỗi vị trí và mô hình thay thế giúp giải quyết được vấn đề còn tồn tại của các thuật toán chỉ sử dụng tốc độ biến đổi.

Chương 1: Cơ sở lý thuyết

Chương 1 trình bày các kiến thức nền tảng của luận án. Phần đầu chương giới thiệu các đối tượng nghiên cứu cơ bản trong lĩnh vực tin sinh học như trình tự tương đồng, sắp hàng đa trình tự tương đồng, mô hình tiến hóa và cây phân loài; phần tiếp theo giới thiệu sơ lược về các phương pháp xây dựng cây phân loài, và trình bày cụ thể hơn về phương pháp cực đại khả năng. Phần sau của chương tập trung vào các bài toán liên quan tới các mô hình biểu diễn quá trình tiến hóa trong sắp hàng. Cuối cùng là các phương pháp đánh giá và so sánh các mô hình khác nhau dựa trên đánh giá cây phân loài tương ứng.

1.1 Các khái niệm cơ bản

1.1.1 Thông tin di truyền

Ngày nay, cùng với sự phát triển của công nghệ giải trình tự; lượng dữ liệu sinh học phân tử khổng lồ thu thập được trong phòng thí nghiệm là điều kiện thuận lợi để các nhà khoa học nghiên cứu sự thay đổi của thông tin di truyền và những ảnh hưởng của nó tới quá trình tiến hóa.

Thông thường, thông tin di truyền được lưu trong vật liệu di truyền **DNA** (Axit Deoxyribo Nucleic); đối với một số vi rút, thông tin này được lưu trong **RNA** (Axit Ribo Nucleic) [1]. Các thông tin DNA này trải qua quá trình phiên mã và dịch mã để tổng hợp protein. Ngoài các đoạn chứa thông tin mã hóa protein, những đoạn còn lại gọi là DNA không mã hóa protein.

Phân tử DNA có cấu trúc dạng chuỗi xoắn kép, bao gồm hai chuỗi polynucleotit xoắn quanh một trục theo chiều từ trái sang phải. Mỗi chuỗi polynucleotit được tạo bởi bốn loại nucleotit. Mỗi nucleotit gồm ba thành phần là đường, nhóm phốt phát và bazơ

nitơ, trong đó bazơ nitơ của các nucleotit khác nhau và quyết định tên gọi của nucleotit tương ứng - A-ađênin, T-timin, C-xitôzin và G-guanin).

RNA có dạng một chuỗi polynucleotit đơn tạo bởi bốn loại bazơ nitơ (A-adenine, T-thymine (U- uracil trong RNA), C-cytosine và G-guanine) trong đó A, G có kích thước lớn hơn T, C. Tương tự trong phân tử DNA, nucleotit trong RNA gồm ba thành phần: một bazơ nitơ, một nhóm đường (phân tử đường cấu tạo RNA ít hơn 1 ôxi so với DNA) và một nhóm phốtphát. Trong RNA các bazơ nối với nhau tạo thành xương sống của chuỗi bởi liên kết cộng hóa trị giữa axit photphoric của nucleotit này với nhóm đường của nucleotit tiếp theo. Hai chuỗi được nối với nhau bởi các liên kết hyđro giữa cặp bazơ nitríc đứng đối diện nhau theo nguyên tắc bổ sung, cụ thể, cặp A, T liên kết bằng hai liên kết hyđro và cặp G, X liên kết bằng ba liên kết hyđro.

Các đoạn trình tự protein là một chuỗi các phân tử đơn, mỗi phân tử là một **axit amin**. Một đoạn protein có thể có độ dài lên tới hàng nghìn axit amin [2]. Có 20 axit amin, các axit amin này được viết tắt bởi một trong hai cách – cách sử dụng một chữ cái và cách sử dụng ba chữ cái (Bảng 1. 1). Ký hiệu một chữ cái luôn được dùng khi phân tích trình tự axit amin; khi đó mỗi trình tự là một chuỗi ký tự, mỗi ký tự là một trong 20 ký tự tương ứng với các axit amin đã định nghĩa.

Quá trình tổng hợp protein bắt đầu bằng quá trình phiên mã, sao chép các thông tin được lưu trong DNA sang RNA thông tin (mRNA). Mã di truyền trong mRNA qua quá trình dịch mã sẽ được chuyển thành trình tự axit amin theo quy tắc mã hóa bộ ba. Nghĩa là, mỗi bộ ba nucleotit liền nhau trên RNA tạo thành một codon, tương ứng với một axit amin. Với bốn loại nucleotit, có 64 codon khác nhau. Trong số đó, codon bắt đầu đánh dấu vị trí bắt đầu dịch mã đầu tiên trên mRNA thường là AUG (starting codon, AUG là codon mã hóa cho axit amin Methionine). Các bộ ba nucleotit liên tiếp tiếp theo sẽ được đọc đến khi gặp codon kết thúc (UAA, UAG hoặc UGA) đánh dấu điểm kết thúc dịch mã. 61 codon còn lại xác định 20 axit amin, một axit amin có thể được mã hóa bởi nhiều hơn một codon khác nhau (Bảng 1. 2).

Bảng 1. 1 Tên, viết tắt 3 ký tự và viết tắt 1 ký tự của 20 axit amin

STT	Tên axit amin	Tên viết tắt (3 ký tự)	Tên viết tắt (1 ký tự)
1	Alanine	Ala	A
2	Arginine	Arg	R
3	Asparagine	Asn	N
4	Aspartic	Asp	D
5	Cysteine	Cys	C
6	Glutamine	Gln	Q
7	Glutamic	Glu	E
8	Glycine	Gly	G
9	Histidine	His	H
10	Isoleucine	Ile	I
11	Leucine	Leu	L
12	Lysine	Lys	K
13	Methionine	Met	M
14	Phenylalanine	Phe	F
15	Proline	Pro	P
16	Serine	Ser	S
17	Threonine	Thr	T
18	Tryptophan	Trp	W
19	Tyrosine	Tyr	Y
20	Valine	Val	V

Bảng 1. 2 Danh sách 64 codon và axit amin tương ứng

	U		**C**		**A**		**G**		
	Codon	Axit amin	Codon	Axit amin	Codon	Axit amin	Codon	Axit amin	
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	**U**
	UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys	**C**
	UUA	Leu	UCA	Ser	UAA	Dừng	UGA	Dừng	**A**
	UUG	Leu	UCG	Ser	UAG	Dừng	UGG	Trp	**G**
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	**U**
	CUC	Leu	CCC	Pro	CAC	His	CGC	Arg	**C**
	CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg	**A**
	CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg	**G**
A	AUT	Ile	ACU	Thr	AAU	Asn	AGU	Ser	**U**
	AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser	**C**
	AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg	**A**
	AUG	Met	ACG	Thr	AAG	Lys	AGG	Arg	**G**
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	**U**
	GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly	**C**
	GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly	**A**
	GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly	**G**

1.1.2 Các biến đổi di truyền

Trong quá trình tiến hóa, việc sao chép thông tin di truyền có thể xảy ra lỗi dẫn đến biến đổi trong thông tin di truyền. Ba phép biến đổi cơ bản là [2]:

- Biến đổi chèn (insertion): khi một nucleotit được chèn thêm vào trình tự nucleotit khiến thông tin bị sai lệch so với thông tin ban đầu. Độ dài của trình tự mới tăng một so với trình tự ban đầu.
- Biến đổi xóa (deletion): khi một nucleotit bị xóa khỏi trình tự nucleotit, khi đó độ dài của trình tự mới giảm một so với trình tự ban đầu.
- Biến đổi thay thế/ biến đổi điểm (substition): khi một nucleotit trên trình tự bị thay thế bằng một nucleotit khác. Độ dài của trình tự không thay đổi; việc thay thế một nucleotit có thể làm thay đổi axit amin hoặc không thay đổi axit amin do một axit amin có thể tương ứng với nhiều codon khác nhau. Đối với đột biến thay thế, người ta chia làm hai loại: đột biến đồng hoán nếu bazơ nitơ cùng nhóm (purin – A và G hoặc pirimiđin – T và C) thay thế cho nhau, và đột biến dị hoán nếu ngược lại.

Các biến đổi chèn hoặc xóa nucleotit được gọi là biến đổi dịch khung do làm lệch khung đọc trình tự nucleotit so với ban đầu. Trong khi biến đổi thay thế không làm dịch khung mà chỉ thay đổi một bộ ba chứa nucleotit bị thay đổi, việc này có thể dẫn tới sự thay đổi axit amin (nếu bộ ba mới và bộ ba cũ mã hóa hai axit amin khác nhau) hoặc không thay đổi axit amin (nếu bộ ba mới và bộ ba cũ cùng mà hóa một axit amin).

Hai nucleotit/axit amin cùng tiến hóa từ một nucleotit/axit amin tổ tiên được gọi là hai nucleotit/axit amin tương đồng. Tương tự, hai trình tự DNA (và trình tự axit amin tương ứng) có chung tổ tiên được gọi là hai trình tự tương đồng. Ví dụ, trong Bảng 1. 3 Ví dụ về các phép biến đổi trên hai trình tự tương đồng, thể hiện các nucleotit trên hai trình tự tương đồng có tên ký hiệu là ADVST và BDSAT; phép biến đổi thay thế đã xảy ra tại cột 6, và 12. Trong đó thay thế tại cột 6 không làm thay đổi trình tự

Bảng 1. 3 Ví dụ về các phép biến đổi trên hai trình tự tương đồng

	1	2	3	4	5	6	7	8	9	10	11	12
ADVST	A	U	G	C	A	U	G	C	C	U	G	U
BDSAT	A	U	G	C	A	C	G	-	-	U	G	C

axit amin tương ứng còn 12 có làm thay đổi; tại cột 8 và 9 đã xảy ra phép xóa hoặc phép chèn.

Các trình tự ở các sinh vật có quan hệ gần nhau thường có cấu trúc và chức năng gần giống nhau [2]. Sự thay đổi thông tin di truyền trong quá trình tiến hóa thể hiện qua những điểm khác biệt khi so sánh trình tự tương đồng, tương ứng với nó là sự thay đổi về thứ tự, cấu trúc và chức năng của các protein tương ứng của các sinh vật đang tồn tại. Khi khác biệt giữa các trình tự càng lớn càng có khả năng cao rằng các sinh vật tương ứng có quan hệ họ hàng cách xa nhau. Tuy nhiên, nếu chỉ dựa trên dữ liệu trình tự, ta có thể quan sát được không quá một biến đổi tại mỗi vị trí tương đồng.

Trong thực tế, vị trí đó có thể đã trải qua nhiều biến đổi phức tạp như đa biến đổi (nhiều biến đổi đã xảy ra tại một vị trí), biến đổi song song (từ tổ tiên chung xảy ra hai biến đổi giống hệt nhau nên ta không thấy sự khác biệt trên hai trình tự quan sát được) hoặc biến đổi ngược (tại một vị trí, nucleotit/axitt amin quan sát được là giống nhau nhưng thực tế đã có nhiều phép biến đổi xảy ra với nucleotit/axitt amin cuối cùng giống nucleotit/axitt amin ban đầu. Do vậy, việc phân tích sự khác biệt giữa các trình tự chỉ bằng so sánh giá trị tại các vị trí tương đồng không thể đánh giá chính xác quá trình biến đổi giữa hai trình tự. Để biểu diễn quá trình này ta cần sử dụng mô hình xác suất ngẫu nhiên để mô phỏng quá trình biến đổi giữa các nucleotit/axit amin.

1.1.3 Mô hình thay thế nucleotit/axit amin

Mô hình thay thế nucleotit/axit amin mô tả quá trình tiến hóa, cụ thể là biểu diễn thông tin về quá trình biến đổi giữa các nucleotit/axit amin của những loài trong bộ dữ liệu được xét. Xét quá trình biến đổi giữa các nucleotit/axit amin tại một vị trí trên tập trình tự. Quá trình biến đổi này là một quá trình ngẫu nhiên và liên tục theo thời gian với tập trạng thái là bốn nucleotit hoặc 20 axit amin tương ứng. Quá trình biến

đổi có thể được mô hình hóa bởi một mô hình Markov [2] biểu diễn bởi một ma trận để xác định tốc độ biến đổi giữa các trạng thái của các vị trí trên tập trình tự. Mô hình Markov thỏa mãn các tính chất sau:

- *Độc lập với quá khứ (memoryless)*: tốc độ biến đổi từ trạng thái i sang trạng thái j là độc lập với các trạng thái trước đó của i.
- *Liên tục (continuous)*: Quá trình biến đổi giữa các trạng thái xảy ra liên tục tại bất cứ thời điểm nào trong quá trình tiến hóa.
- *Ổn định (stationary)*: Tần số của mỗi nucleotit/axit amin ở trạng thái cân bằng, không thay đổi trong suốt quá trình tiến hóa.

Mỗi vị trí trên một trình tự được coi là một biến ngẫu nhiên n trạng thái tùy thuộc vào loại dữ liệu (tức là n = 4 trong trường hợp dữ liệu DNA và n = 20 trong trường hợp dữ liệu protein). Khi đó, ma trận xác suất chuyển $P(t) = \{p_{ij}(t)\}$ mô tả quá trình Markov là ma trận cỡ 4×4 hoặc 20×20; trong đó $p_{ij}(t)$ biểu diễn xác suất trạng thái i chuyển thành trạng thái j sau t đơn vị thời gian. Các điều kiện đề cập ở trên của quá trình Markov được viết lại cho các phần tử trong ma trận P như sau:

(i) Tổng xác suất chuyển từ một trạng thái i sang các trạng thái khác bằng 1: $\sum_{j=1}^{n} p_{ij}(t) = 1$

(ii) Phương trình Chapman–Kolmogorov: $P(t+s) = P(t) + P(s)$

(iii) $p_{ij}(t) > 0 \; \forall \; t > 0$

(iv) Điều kiện khởi tạo: $\begin{cases} p_{ii}(0) = 1 & \forall i \\ p_{ij}(0) = 0 & \forall i \neq j \end{cases}$

Trong lân cận Δt của $t = 0$, $P(\Delta t)$ được xấp xỉ bởi công thức khai triển Taylor:

$$P(\Delta t) = P(0) + \Delta t Q \qquad (1.1)$$

Với $Q = \{q_{ij}\}$ là ma trận tốc độ thay thế tức thì, tức là q_{ij} là tốc dộ thay thế tức thì từ trạng thái i sang trạng thái j. Trong phương trình 1.1 ma trận tốc độ Q là đạo hàm cấp

1 của $P(t)$. Vì $P(t)$ là quá trình Markov đồng nhất về thời gian, đạo hàm cấp 1 của nó là Q không phụ thuộc vào thời gian.

Để thỏa mãn điều kiện (i) – tổng xác suất chuyển từ trạng thái i sang các trạng thái khác bằng 1 sau khoảng thời gian bất kỳ, tổng các phần tử trên một hàng của Q phải bằng 0:

$$\sum_{j=1}^{n} q_{ij} = 0 \tag{1.2}$$

Các phần tử trên đường chéo chính được xác định thông qua các phần tử còn lại trên cùng hàng:

$$q_{ii} = -\sum_{j \neq i} q_{ij} \tag{1.3}$$

Nếu đặt tổng số thay thế của tất cả các trạng thái trong một đơn vị thời gian là μ:

$$\mu = -\sum_{i=1}^{n} \pi_i q_{ii} \tag{1.4}$$

Thì trong một khoảng thời gian t, tổng số thay thế là $d = \mu t$. Ma trận tốc độ thay thế tức thì Q có thể được chuẩn hóa để tổng số thay thế trong một đơn vị thời gian là 1 ($\mu = 1$) mà không làm thay đổi tương quan thay thế giữa các trạng thái.

Từ phương trình Chapman–Kolmogorov, $P(t)$ được viết dưới dạng một hàm của Q và biến t:

$$P(t) = e^{tQ} \tag{1.5}$$

Để đơn giản hóa việc tính toán, quá trình P thường được giả thiết là có tính thuận nghịch; tức là, số thay đổi từ trạng thái i sang trạng thái j trong một đơn vị thời gian đúng bằng số thay đổi từ j sang i:

$$\pi_i q_{ij} = \pi_j q_{ji} \text{ hay } \frac{q_{ij}}{\pi_j} = \frac{q_{ji}}{\pi_i} \tag{1.6}$$

Ký hiệu ma trận $R = \left\{ r_{ij} = \frac{q_{ij}}{\pi_j} \right\}$ là ma trận hệ số hoán đổi. Khi đó, R là ma trận đối xứng vì $r_{ij} = \frac{q_{ij}}{\pi_j} = \frac{q_{ji}}{\pi_i} = r_{ji}$ và các phần tử trên đường chéo chính $r_{ii} = 0$. Ta viết lại ma trận Q dưới dạng tích của ma trận R và vectơ tần số của các trạng thái Π:

$$q_{ij} = \begin{cases} \pi_j r_{ij} & \text{với } i \neq j \\ -\sum_{x \neq i} q_{ix} & \text{với } i = j \end{cases} \quad (1.7)$$

Với n trạng thái, để xác định trực tiếp ma trận Q cần xác định giá trị cho $n \times (n-1)$ tham số, trong khi nếu tính Q thông qua R và Π ta chỉ cần xác định $\frac{n(n-1)}{2} - 1$ tham số cho R (do ma trận R đối xứng có đường chéo chính bằng 0 và tổng các phần tử bằng 0) và $n-1$ tham số cho Π (do vectơ tần số có tổng thành phần bằng 1); tổng số là $\frac{n(n+1)}{2} - 2$ tham số.

Với trường hợp dữ liệu DNA, mô hình thay thế cần tổng cộng 8 tham số (3 tham số cho tần suất nucleotit và 5 tham số cho ma trận hệ số hoán đổi). Do số lượng tham số khá nhỏ, việc ước lượng các tham số này thường được thực hiện cho từng bộ dữ liệu. Có một số kiểu mô hình thay thế phổ biến cho DNA [3]–[5], mỗi mô hình có số lượng tham số khác nhau. Một trong số mô hình thường dùng là mô hình GTR (General Time-Reversible Model) [6]. Với 5 tham số $a; b; c; d; e; f$ của ma trận R và $\pi_A;\ \pi_C;\ \pi_G;\ \pi_T$ tương ứng là tần suất của các nucleotit A, C, G và T, mô hình GTR xác định bởi ma trận Q:

$$Q = \begin{bmatrix} -\sum_{i \neq A} q_{Ai} & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & -\sum_{i \neq C} q_{Ci} & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & -\sum_{i \neq G} q_{Gi} & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & -\sum_{i \neq T} q_{Ti} \end{bmatrix}$$

Với dữ liệu protein, mô hình thay thế có tổng cộng 208 tham số (do số lượng là 20 axit amin nên ma trận hệ số hoán đổi có 189 tham số và vectơ tần suất axit amin có 19 tham số). Việc ước lượng 208 tham số trong mỗi lượt chạy không thể thực hiện được cho các bộ dữ liệu nhỏ và vừa do số lượng tham số quá lớn. Vì vậy các mô hình thay thế axit amin thường được ước lượng sẵn trên các bộ dữ liệu lớn [7]–[10], sau đó được sử dụng chung cho các bộ dữ liệu khác nhau với giả thiết quá trình tiến hóa của các loài sinh vật nói chung tuân theo các quy tắc tương đồng. Các mô hình đã ước lượng được chia làm hai loại: mô hình chung cho nhiều loài (ví dụ như: LG [9], JTT [10] – là các mô hình được ước lượng sử dụng lượng lớn sắp hàng thu thập từ nhiều loại sinh vật để tạo độ đa dạng cho dữ liệu, những mô hình này được dùng với giả thiết là các loài sinh vật nói chung đều có quá trình tiến hóa tương tự nhau) và mô hình cho một nhóm loài cụ thể. Dữ liệu sử dụng trong trường hợp ước lượng mô hình cho nhóm loài cụ thể tập trung vào các loài có quan hệ gần, thường là những loài có đặc điểm khác biệt so với các sinh vật nói chung (ví dụ như: HIVb, HIVw [11], FLU [12] là các mô hình ước lượng trên một loại vi rút cụ thể).

Bảng 1. 4 trình bày một ví dụ về mô hình thay thế axit amin (mô hình trong bảng là mô hình LG). 19 dòng đầu của bảng thể hiện nửa dưới của ma trận hệ số hoán đổi; dòng dưới cùng thể hiện tần suất của từng axit amin. Ma trận tốc độ thay thế tức thì của axit amin xác định bằng tích của hai thành phần này được gọi là mô hình thay thế Q, hay gọi tắt là mô hình Q

Bảng 1. 4 Ví dụ về mô hình thay thế axit amin (mô hình LG)

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A																				
R	0.425																			
N	0.277	0.752																		
D	0.395	0.124	5.076																	
C	2.489	0.535	0.529	0.063																
Q	0.970	2.808	1.696	0.523	0.085															
E	1.039	0.364	0.542	5.244	0.003	4.129														
G	2.066	0.390	1.438	0.845	0.569	0.268	0.349													
H	0.359	2.427	4.509	0.927	0.641	4.814	0.424	0.311												
I	0.150	0.127	0.192	0.011	0.321	0.073	0.044	0.009	0.109											
L	0.395	0.302	0.068	0.015	0.594	0.582	0.070	0.044	0.366	4.145										
K	0.537	6.326	2.145	0.283	0.013	3.234	1.807	0.297	0.697	0.159	0.138									
M	1.124	0.484	0.371	0.026	0.894	1.673	0.174	0.140	0.442	4.274	6.312	0.657								
F	0.254	0.053	0.090	0.017	1.105	0.036	0.019	0.090	0.682	1.113	2.593	0.024	1.799							
P	1.178	0.333	0.162	0.394	0.075	0.624	0.419	0.197	0.509	0.078	0.249	0.390	0.100	0.094						
S	4.727	0.858	4.008	1.240	2.784	1.224	0.612	1.740	0.990	0.064	0.182	0.749	0.347	0.362	1.338					
T	2.140	0.579	2.001	0.426	1.143	1.080	0.605	0.130	0.584	1.034	0.303	1.137	2.020	0.165	0.571	6.472				
W	0.181	0.594	0.045	0.030	0.670	0.236	0.078	0.268	0.597	0.112	0.620	0.050	0.696	2.457	0.095	0.249	0.141			
Y	0.219	0.314	0.612	0.135	1.166	0.257	0.120	0.055	5.307	0.233	0.300	0.132	0.481	7.804	0.090	0.401	0.246	3.152		
V	2.548	0.171	0.084	0.038	1.959	0.210	0.245	0.077	0.119	10.649	1.703	0.185	1.899	0.655	0.297	0.098	2.188	0.190	0.249	
Π	0.079	0.056	0.042	0.053	0.013	0.041	0.072	0.057	0.022	0.062	0.099	0.065	0.023	0.042	0.044	0.061	0.053	0.012	0.034	0.069

1.1.4 Sắp hàng đa trình tự tương đồng

Như đề cập ở 1.1.2, các trình tự được gọi là tương đồng nếu các trình tự này là kết quả của quá trình tiến hóa từ một trình tự tổ tiên chung trước đó [2]. Qua phân tích trình tự tương đồng của các loài, cụ thể là bằng việc so sánh các giá trị tại cùng một vị trí trên trình tự, ta có thể đánh giá mức độ quan hệ họ hàng giữa các loài là gần hay xa. Để so sánh nhiều loài cùng lúc, cần thiết lập một bảng các trình tự tương đồng trong đó các vị trí tương đồng trên mỗi trình tự được căn chỉnh thành một cột của bảng. Bảng này được gọi là sắp hàng đa trình tự tương đồng (gọi tắt là sắp hàng). Bảng có n hàng - mỗi hàng chứa nội dung một trình tự, l cột là số nucleotit/axit amin trong mỗi trình tự. l cũng được gọi là độ dài của mỗi trình tự hoặc độ dài của sắp hàng. Sắp hàng cần thể hiện được sự thay đổi nếu có tại từng vị trí trên mỗi trình tự. Với những vị trí xảy ra biến đổi điểm, độ dài các trình tự không có gì thay đổi, ta chỉ cần thay đổi giá trị tại điểm biến đổi. Còn với những vị trí bị xóa hoặc chèn, biến đổi được thể hiện bằng khoảng trống (biểu diễn bởi dấu '-'), mỗi khoảng trống ứng với một vị trí biến đổi. Lưu ý rằng, độ dài ban đầu của các trình tự có thể là khác nhau do kết quả của các biến đổi chèn và xóa, nhưng sau khi được sắp hàng, các trình tự trong sắp hàng sẽ có cùng độ dài vì tất cả thông tin biến đổi đều được thể hiện đầy đủ [13].

Ví dụ, Bảng 1. 5 thể hiện một sắp hàng axit amin gồm bốn trình tự; mỗi trình tự có độ dài là 15. Tên của các trình tự được để thể hiện ở cột đầu tiên bên trái, nội dung của trình tự ở dòng tương ứng bên phải. Mỗi vị trí trên trình tự là một ký tự thể hiện axit amin tại đó hoặc dấu gạch ngang thể hiện một phép biến đổi chèn hoặc xóa.

Bảng 1. 5 Ví dụ về sắp hàng đa trình tự tương đồng

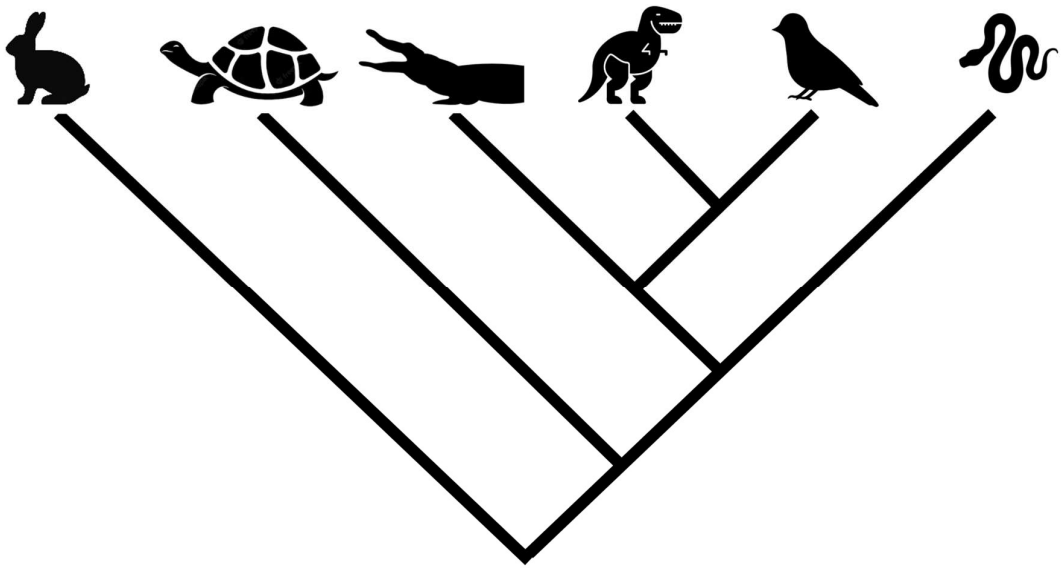
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ASVDT	E	H	D	-	N	D	E	M	C	Q	L	K	P	L	P
ASVEF	F	H	D	R	-	D	E	M	C	Q	L	K	P	L	P
ASVDP	F	G	D	R	-	D	E	M	C	Q	L	K	P	L	P
BSVDE	F	G	D	R	-	V	H	M	C	Q	L	K	P	L	P

Việc xác định sắp hàng đa trình tự tương đồng là bài toán đầu tiên cần thực hiện để phân tích một tập dự liệu. Các thuật toán sắp hàng sẽ tìm cách thêm vào các kí tự gạch ngang "-" vào các trình tự ban đầu để biểu diễn các nucleotit hoặc axit amin đã bị xóa sao cho các trình tự kết quả có cùng độ dài và các nucleotit/axit amin tại cùng vị trí là các nucleotit/axit amin tương đồng.

Một số thuật toán khác nhau đã được đề xuất để xây dựng sắp hàng cho một bộ n trình tự. Phổ biến nhất là phương pháp sắp hàng lũy tiến. Trong đó, các thuật toán thực hiện việc chèn thêm các khoảng trống vào các trình để cực tiểu hóa thiểu hàm mục tiêu tính dựa trên khoảng cách giữa các trình tự và hàm phạt tương ứng với số lượng khoảng trống đã thêm vào dữ liệu. Điển hình là MUSCLE [13] - thuật toán được sử dụng rộng rãi và được đánh giá là có độ chính xác cao. Thuật toán có thể tính toán và tạo sắp hàng cho hàng nghìn trình tự một cách nhanh chóng.

1.1.5 Cây phân loài

Xét một số đối tượng nghiên cứu (mỗi đối tượng có thể là một loài lớn hay chi tiết tới từng cá thể), cây phân loài (hay còn gọi là cây phát sinh loài, cây tiến hóa) của những đối tượng này là một dạng đồ thị dạng cây thể hiện quá trình phát sinh các đối tượng nghiên cứu và mức độ tương đồng giữa chúng. Cụ thể, đồ thị được biểu diễn dưới dạng cây nhị phân, có các nút lá tương ứng với các đối tượng được xét và các nút trong của cây tương ứng với một tổ tiên chung; mỗi nút trong có đúng hai nút con (Hình 1. 1). Độ dài của các nhánh trên cây biểu thị khoảng cách tiến hóa (số phép biến đổi) giữa hai đỉnh đầu mút.



Hình 1. 1 Ví dụ minh họa cây phát sinh loài bò sát.

Cây phân loài được chia làm hai loại: cây có gốc và cây không có gốc. Cây có gốc xác định chiều tiến hóa theo thời gian, nghĩa là gốc cây tương ứng với một tổ tiên chung từ thời điểm xa nhất, sự thay đổi của thông tin di truyền theo thời gian diễn ra theo chiều từ gốc đến lá. Tuy nhiên do không có dữ liệu về sinh vật tổ tiên; bên cạnh đó, do các mô hình thay thế nucleotit/axit amin hiện nay thường giải thiết rằng quá trình tiến hóa có tính thuận nghịch nên không thể xác định chiều của biến đổi. Do vậy, các cây xây dựng được là các cây không có gốc, những cây này không thể hiện quá trình tiến hóa theo thời gian mà chỉ xác định mức độ quan hệ tương đối giữa các đối tượng đang xét.

Mỗi nhánh trong cây chia tập các đối tượng đang xét thành hai tập con không giao nhau. Một cách tách đôi như vậy được gọi là một *phân nhánh* (*split*). Khi xét hai cấu trúc cây cụ thể, số lượng phân nhánh khác nhau giữa hai cây này được sử dụng để tính khoảng cách giữa chúng. Khoảng cách này được gọi là khoảng cách Robinson-Fould (RF) [14]. Với một bộ dữ liệu được cho dưới dạng một sắp hàng đa trình tự tương đồng có n trình tự, số lượng cấu trúc cây khác nhau tăng theo hàm mũ (Bảng 1. 6), do vậy, xác định cấu trúc cây thể hiện được quá trình phát sinh loài trong thực tế là bài toán rất khó.

Bảng 1. 6 Số lượng cây không có gốc tương ứng với số trình tự trong sắp hàng

n	**Số lượng cây phân loài không có gốc**
3	1
4	3
5	15
6	105
…	…
10	2,027,025
…	…
20	13,113,070,457,687,988,603,440,625

1.1.6 Xây dựng cây phân loài bằng phương pháp cực đại khả năng

Các thuật toán xây dựng cây phân loài có thể chia thành bốn nhóm là: phương pháp cực tiểu số lượng biến đổi, phương pháp dựa trên khoảng cách và hai phương pháp dựa trên mô hình xác suất là phương pháp Bayes và phương pháp cực đại khả năng.

Phương pháp cực tiểu số lượng biến đổi xây dựng cây phân loài dựa vào phân tích đặc điểm của các trình tự trong sắp hàng. Ý tưởng cốt lõi của phương pháp cực tiểu số lượng biến đổi là: cây mô tả quá trình tiến hóa tốt nhất là cây có thể biểu diễn dữ liệu cần ít thay đổi nhất hay chính là cây có tổng số lượng biến đổi trên các cạnh nhỏ nhất [15]. Cây tốt nhất có thể tìm theo phương pháp duyệt toàn bộ, tuy nhiên với số lượng cấu trúc cây tăng theo hàm mũ những phương pháp này chỉ thực hiện được trên các bộ dữ liệu nhỏ; trong thực tế các nghiên cuucs thường dùng phương pháp leo đồi kết hợp với các kỹ thuật tạo cây khác nhau để xây dựng cây với số lượng lên đến hàng nghìn trình tự [15]–[17] .

Xây dựng cây phân loài dựa vào khoảng cách giữa các trình tự là phương pháp được phát triển và sử dụng rộng rãi [18]–[20]. Xét một cây T có n lá tương ứng với các đối

tượng đang được nghiên cứu, khoảng cách giữa hai lá i và j được tính bằng tổng độ dài các cạnh nằm trên đường đi từ i tới j trên T. Với đầu vào là một sắp hàng n trình tự tương đồng, trước hết ta cần tính ma trận khoảng cách di truyền W giữa các cặp trình tự; đầu ra của bài toán là cây nhị phân n lá sao cho khoảng cách giữa các lá phù hợp với khoảng cách tương ứng trong ma trận khoảng cách W. Cách đơn giản nhất để sử dụng ma trận khoảng cách là bắt đầu với n đỉnh rời nhau, sau đó lần lượt gộp hai đỉnh gần nhất thành một đỉnh mới và lặp lại việc này đến khi xây dựng cây n lá hoàn chỉnh. Ưu điểm của phương pháp dựa trên khoảng cách so với các phương pháp còn lại là chỉ thực hiện tính toán trên ma trận khoảng cách giữa các trình tự, trong khi số trình tự trong sắp hàng thường nhỏ hơn độ dài sắp hàng nhiều lần. Do vậy cây phân loài được tìm ra nhanh chóng. Cây phân loài xây dựng bằng các phương pháp khoảng cách thường được sử dụng làm cây bắt đầu cho các thuật toán tối ưu cây khác, ví dụ như làm cây bắt đầu cho các thuật toán cực đại khả năng [2].

Đối với phương pháp Bayes trước tiên ta gán xác suất cho cấu trúc cây theo hiểu biết sẵn có, nếu chưa có tri thức gì về cấu trúc cây của các loài đang nghiên cứu thì ta gán xác suất bằng nhau cho các tất cả các cây. Sau đó, ta lấy mẫu dữ liệu và sử dụng một mô hình tiến hóa ngẫu nhiên cùng với định lý Bayes để tính toán khả năng xảy ra của những cây này. Các cây được xếp vào một trong hai loại là chấp nhận hoặc loại bỏ dựa trên một ngưỡng do người dùng đặt ra. Nếu một cấu trúc cây được chấp nhận, cấu trúc này được thêm vào phân phối hậu nghiệm. Sau khi lặp lại quá trình này với số lần đủ lớn, ta thu được một phân phối hậu nghiệm lớn. Khi đó, xác suất một cây được chấp nhận trên tất cả số lần lấy mẫu là xác suất hậu nghiệm mà cây đó là cây phân loài trong thực tế [2].

Cực đại khả năng (ML – maximum likehood) là phương pháp thống kê được sử dụng rộng rãi để ước lượng các tham số của mô hình xác suất tương ứng với bộ dữ liệu quan sát được. Hiện nay phương pháp ML được sử dụng phổ biến trong việc xây dựng cây phân loài vì thường cho kết quả tốt hơn các phương pháp khác [2].

Với một sắp hàng cho trước $D = \{d_1, d_2, \dots, d_n\}$ là một sắp hàng có n trình tự, mỗi trình tự d_k có độ dài l ($d_k = d_{k1}\ d_{k2} \dots d_{kl}$. Phương pháp ML xác định cây nhị phân không gốc T và mô hình thay thế nucleotit/axit amin Q làm cực đại hóa hàm khả năng $L(T|Q, D)$:

$$L(T|Q, D) = \prod_{i=1}^{l} L(T|Q, d_i) \tag{1.8}$$

Trong đó d_i là cột thứ i của sắp hàng và giá trị khả năng tại mỗi vị trí i (cột i) là một hàm tỉ lệ thuận với xác suất thu được của dữ liệu:

$$L(T|Q, d_i) \propto P(d_i|T, Q) \tag{1.9}$$

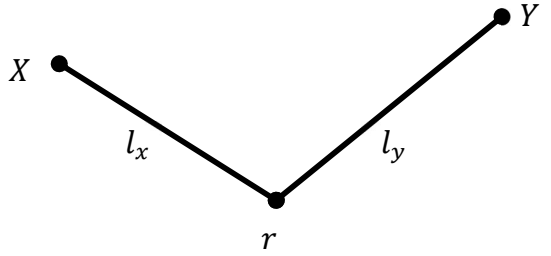
Các tham số cần được xác định ở đây là cấu trúc cây, độ dài của các nhánh và các giá trị trong mô hình tiến hóa. Cây phân loài tối ưu là cây có giá trị khả năng cao nhất hay chính là cây có xác suất cao nhất đối với bộ dữ liệu quan sát được, mô hình thay thế được dùng để tính giá trị khả năng cần được chọn là mô hình phù hợp nhất với quá trình tiến hóa trong bộ sắp hàng D.

Sau đây là ví dụ minh họa cho việc tính giá trị khả năng của cây nhị phân ở trường hợp đơn giản nhất – cây chỉ có hai lá.

Xét cây nhị phân T của sắp hàng D, có hai trình tự là $X = \{x_i\}$ và $Y = \{y_i\}$, mô hình phù hợp nhất dùng để tính toán giá trị khả năng của cây ký hiệu là Q. Ta cần tính giá trị khả năng của cây $L(T|Q, D)$ thông qua giá trị khả năng của từng vị trí $L(T|Q, d_i = x_i y_i)$

Vì mô hình Q có tính thuận nghịch theo thời gian, không mất tính tổng quát, ta thêm một đỉnh gốc giả r nằm tại vị trí bất kỳ trên cạnh và ký hiệu khoảng cách từ r tới hai lá X, Y là l_x, l_y. (Hình 1. 2 Cây có một đỉnh gốc và hai lá).

Tại vị trí i trên đỉnh gốc r, có 4 giá trị có thể nhận là các nucleotit A, T, C và



Hình 1. 2 Cây có một đỉnh gốc và hai lá

G. Khi đó $P(d_i|T,Q)$ được tính bằng tổng xác suất của cả 4 trường hợp khác nhau của nucleotit lại gốc r. Tức là:

$$\begin{aligned} P(d_i|T,Q) &= P(x_i y_i|T,Q) \\ &= \pi_A P_{x_i A}(l_x|T,Q) \times P_{y_i A}(l_y|T,Q) \\ &\quad + \pi_T P_{x_i T}(l_x|T,Q) \times P_{y_i T}(l_y|T,Q) \\ &\quad + \pi_C P_{x_i C}(l_x|T,Q) \times P_{y_i C}(l_y|T,Q) \\ &\quad + \pi_G P_{x_i G}(l_x|T,Q) \times P_{y_i G}(l_y|T,Q) \end{aligned} \tag{1.10}$$

Trong đó: $P_{uv}(l|T,Q)$ là xác suất nucleotit u biến đổi thành nucleotit v nếu có l biến đổi giữa hai nucleotit u và v.

Cách tính giá trị khả năng cho trường hợp tổng quát cho cây có n lá, được mở rộng từ trường hợp này, cần số lượng tính toán lớn. Mặc dù phương pháp ML là phương pháp có độ phức tạp tính toán cao và cần nhiều thời gian để tính toán, phương pháp này cho kết quả cây phân loài đáng tin cậy nhất và là phương pháp thường được sử dụng hiện nay. Hai bộ thư viện hỗ trợ xây dựng cây bằng phương pháp ML phổ biến nhất là: RAxML-NG [21] và IQTREE [22], ngoài ra còn có nhiều phần mềm khác, ví dụ như VeryFastTree [23] - kết hợp ML với phương pháp heuristics để tăng tốc độ, giúp xây dựng cây nhanh hơn gấp nhiều lần so với các phần mềm trên.

1.2 Bài toán ước lượng mô hình thay thế axit amin

Để tính giá trị khả năng của một cấu trúc cây ta cần xác định một mô hình thay thế. Sự khác nhau giữa các mô hình thay thế có tác động lớn đến kết quả của nhiều bài toán phân tích dữ liệu di truyền [2]. Một mô hình phản ánh đúng quá trình biến đổi giữa các nucleotit/axit amin trong dữ liệu là tiền đề giúp kết quả tính toán chính xác và gần với thực tế hơn. Ngoài ra, mô hình thay thế còn cần thiết cho nhiều tác vụ khác như tính toán khoảng cách di truyền, mô phỏng dữ liệu trình tự, phân tích các cụm gen bất biến [24]

1.2.1 Bài toán

Như đã đề cập ở các phần trên, mô hình thay thế axit amin là một ma trận cỡ 20×20 dùng để biểu diễn quá trình biến đổi xảy ra giữa 20 axit amin. Mô hình này được sử dụng để ước lượng khoảng cách di truyền giữa các cặp trình tự và tính giá trị khả năng (likelihood) trong quá trình xây dựng cây phân loài bằng phương pháp ML. Để xây dựng cây phân loài bằng phương pháp cực đại khả năng, việc chọn một mô hình thay thế phù hợp đóng vai trò quyết định tới kết quả. [9], [25], [26] đã chỉ ra rằng việc sử dụng mô hình thay thế nucleotit/axit amin không phù hợp có thể gây ảnh hưởng nghiêm trọng đến tính đúng đắn của cây phân loài được chọn..

Nhiều mô hình thay thế axit amin đã được đề xuất và sử dụng rộng rãi như PAM[7], JTT[10], WAG [8] hay LG [27]. Với các mô hình thay thế đã được đề xuất, các nhà nghiên cứu có thể lựa chọn sử dụng mô hình phù hợp nhất với dữ liệu của mình. Tuy nhiên, trong quá trình nghiên cứu, một số loài có quá trình tiến hóa mang nhiều đặc trưng khác biệt với các quá trình đã được mô tả bởi những mô hình sẵn có, dẫn đến nhu cầu cần có các mô hình thay thế đã được ước lượng sẵn phù hợp hơn để sử dụng trong những trường hợp này.

Điển hình trong số các loài có nhiều đặc điểm khác biệt là vi rút, do vậy nhiều mô hình thay thế cho các loại virus đã được đề xuất. Năm 2007, Nickle và cộng sự đề xuất mô hình thay thế axit amin cho vi rút HIV, bao gồm hai mô hình riêng biệt để mô phỏng quá trình thay thế của vi rút tại một đối tượng nhiễm bệnh - HIVw, và mô phỏng quá trình thay thế của vi rút khi lan truyền giữa các người bệnh - HIVb. Năm 2010, mô hình FLU được nhóm của Cuong Dang và cộng sự xây dựng để mô phỏng quá trình thay thế axit amin trên vi rút cúm [12]. Các kết quả thực nghiệm đều cho thấy các mô hình dành riêng cho vi rút cho kết quả tốt hơn so với mô hình thay thế chung khi phân tích trình tự protein của các loại vi rút tương ứng.

Bài toán ước lượng mô hình thay thế có thể phát biểu như sau:

Đầu vào: Một tập N sắp hàng axit amin ký hiệu là $\mathbf{D} = \{D_1, \dots, D_N\}$. Mỗi sắp hàng có thể có từ vài chục đến hàng chục nghìn trình tự.

Bài toán: Ước lượng mô hình thay thế axit amin mô tả xác suất biến đổi giữa các axit amin trong quá trình tiến hóa ứng với dữ liệu trong **D**

Đầu ra: Mô hình thay thế axit amin Q biểu diễn quá trình biến đổi axit amin trên các trình tự của bộ dữ liệu **D**.

Yêu cầu: Phương pháp ước lượng cần thu được mô hình có độ chính xác cao và thực hiện trong thời gian chấp nhận được.

Nhiều phương pháp đã được đề xuất để ước lượng mô hình thay thế axit amin, nhìn chung có thể chia làm hai hướng tiếp cận là phương pháp đếm [7], [10] và phương pháp cực đại khả năng [9], [12], [28].

1.2.2 Các phương pháp ước lượng mô hình thay thế

Phương pháp đếm

Phương pháp đến được đề xuất bởi Dayhoff và các cộng sự [7]. Các tác giả sử dụng một họ ma trận PAM (*Point Accepted Mutations matrices* or *Percent of Accepted Mutations)* là các ma trận điểm thay thế. PAM được tạo ra từ khoảng cách tiến hóa trong các trình tự được xét. Cụ thể, mỗi ma trận x-PAM là một ma trận cỡ 20×20, thể hiện xác suất thay đổi giữa các cặp axit amin với điều kiện tổng số biến đổi trung bình trên 100 axit amin là x. Giá trị x càng cao thể hiện khoảng cách tiến hóa càng lớn hay sự khác nhau của các loài trong sắp hàng lớn. Phương pháp bắt đầu bằng việc xây dựng ma trận 1-PAM sau đó ngoại suy ra các ma trận tương ứng với khoảng cách lớn hơn. Lưu ý rằng, với phương pháp ban đầu do Dayhoff và các cộng sự đề xuất, các trình tự trong các sắp hàng làm đầu vào cần có độ tương đồng cao (thường là trên 85%) để đảm bảo sự thay thế là trực tiếp. Trong nghiên cứu của mình, Jones và cộng sự đã sử dụng ý tưởng của phương pháp đếm để ước lượng mô hình JTT. Dữ liệu được sử dụng gồm nhiều bộ dữ liệu của lượng lớn các loài khác nhau, nếu một bộ dữ liệu có độ tương đồng thấp hơn, nó sẽ được gom thành các cụm để đảm bảo điều kiện

về độ tương đồng trên 85% trong mỗi cụm. Mô hình JTT được sử dụng rộng rãi và cho kết quả tốt trong các bài toán liên quan.

Phương pháp cực đại khả năng

Với đầu vào $\mathbf{D} = (D_1, \dots, D_N)$ là bộ dữ liệu gồm N sắp hàng đa trình tự tương đồng.

Gọi $\mathbf{T} = (T_1, \dots T_N)$ là tập các cây tương ứng với các sắp hàng trong $\mathbf{D}$.

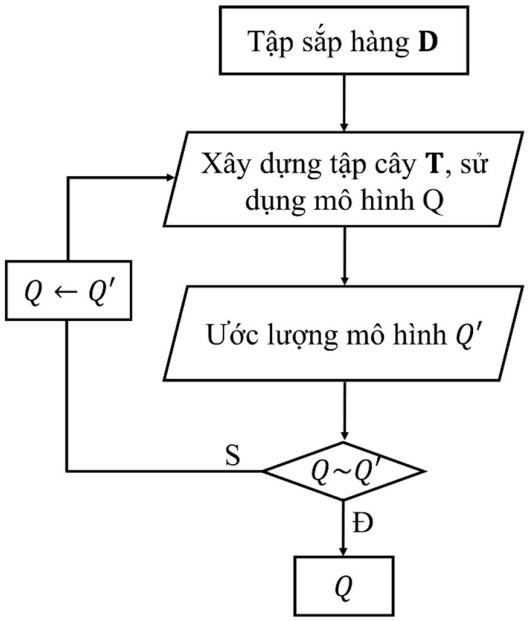
Phương pháp ước lượng cực đại khả năng xác định tập cây $\mathbf{T}$ và mô hình Q để cực đại hóa giá trị khả năng (likelihood) $L(Q, \mathbf{T}|\mathbf{D})$ tính bởi công thức:

$$L(Q, \mathbf{T}|\mathbf{D}) = \prod_{i=1}^{N} L(Q, T_i|D_i) = \prod_{i=1}^{N}\prod_{j=1}^{l_i} L\big(Q, T_i|D_{ij}\big) = \prod_{i=1}^{N}\prod_{j=1}^{l_i} P(D_{ij}|Q, T_i) \quad (1.11)$$

Trong đó:

- l_i là độ dài của sắp hàng D_i
- D_{ij} là dữ liệu tại vị trí thứ j của sắp hàng D_i
- Giá trị khả năng $L(Q, T_i|D_{ij})$ được tính tỉ lệ với xác suất có điều kiện $P(D_{ij}|\, Q, T_i)$ của vị trí D_{ij} khi biết mô hình Q và cây T_i.

Ưu điểm của phương pháp ML là không ràng buộc về tính tương đồng của các loài trong sắp hàng như phương pháp đếm nhưng việc tối ưu hóa đồng thời các tham số của mô hình Q và tham số của tập các cây $\mathbf{T}$ (bao gồm cấu trúc cây và độ dài các nhánh) là rất khó khăn. Nhiều chiến lược xấp xỉ phương pháp ML đã được đề xuất để làm giảm thời gian chạy. Điển hình trong số đó, Whelan và Goldman đã nghiên cứu và chỉ ra rằng với các cây gần tối ưu, mô hình Q ước lượng được sẽ hội tụ [8]; hệ quả là, thay vì tối ưu cùng lúc cả Q và $\mathbf{T}$, ta sẽ tìm tập cây gần tối ưu $\mathbf{T}$ trước, sau đó cố định $\mathbf{T}$ và tìm mô hình Q (Hình 1. 3).



Hình 1. 3 Sơ đồ các bước ước lượng mô hình thay thế axit amin bằng phương pháp cực đại khả năng

1.3 Tính không đồng nhất của quá trình biến đổi tại các vị trí khác nhau

Do tiến bộ trong công nghệ giải trình tự, các bộ dữ liệu rất lớn ngày càng trở nên phổ biến hơn. Để xây dựng cây phân loài cho những bộ dữ liệu như vậy cần rất nhiều thời gian để chọn lựa mô hình thay thế phù hợp và tính toán giá trị khả năng trên các cấu trúc cây. Kéo theo sự ra đời của nhiều thuật toán ML xây dựng cây phân loài cho dữ liệu lớn, ví dụ như IQPNNI [29], hay IQTREE [25]. Bên cạnh việc phát triển các thuật toán mới nhanh và hiệu quả hơn, để tăng cường tính đúng đắn của cây phân loài xây dựng được thì mô hình thay thế được sử dụng để tính toán giá trị khả năng cũng là một nhân tố quan trong ảnh hưởng đến kết quả của quá trình xây dựng và cấu trúc cây [9], [30]–[32]. Việc sử dụng mô hình thay thế không phù hợp có thể dẫn đến lỗi sai nghiêm trọng trên cấu trúc cây như đánh giá độ tin cậy cao cho những nhánh không chính xác [33].

Trong thủ tục tính giá trị khả năng của cây T ứng với sắp hàng D ở phần 1.1.6, sau khi đã xác định mô hình Q phù hợp nhất, mô hình Q sẽ được sử dụng để tính giá trị khả năng cho tất cả các vị trí trong sắp hàng, từ đó tính giá trị khả năng của sắp hàng với cây T và mô hình Q. Trong thủ tục tính giá trị khả năng này, duy nhất một mô hình Q được sử dụng cho tất cả các vị trí trong sắp hàng do quá trình tiến hóa thường được giả thiết là có tính đồng nhất - tức là tốc độ thay thế từ trạng thái này sang trạng thái khác là không đổi và giống nhau trên toàn bộ các vị trí trong sắp hàng. Tuy nhiên, nhiều nghiên cứu đã chỉ ra rằng trong thực tế, quá trình tiến hóa không giống nhau tại tất cả các vị trí; tức là quá trình quá trình biến đổi có thể khác nhau tại các vị trí khác nhau [34]–[39]. Với dữ liệu cỡ hệ gen, giả thiết về tính đồng nhất trong biến đổi càng trở nên bất hợp lý và có khả năng gây ảnh hưởng đến độ chính xác của các tính toán trong quá trình phân tích và đánh giá [40]. Do vậy nhiều nghiên cứu đã đề xuất các phương pháp khác nhau để giải quyết vấn đề này như [30], [32], [35]–[38], [41]. Mục tiêu cần đạt đến của các phương pháp là vừa có thể biểu diễn được quá trình tiến hóa không đồng nhất giữa các vị trí mà vẫn đảm bảo thực hiện được các tính toán trong thời gian hợp lý.

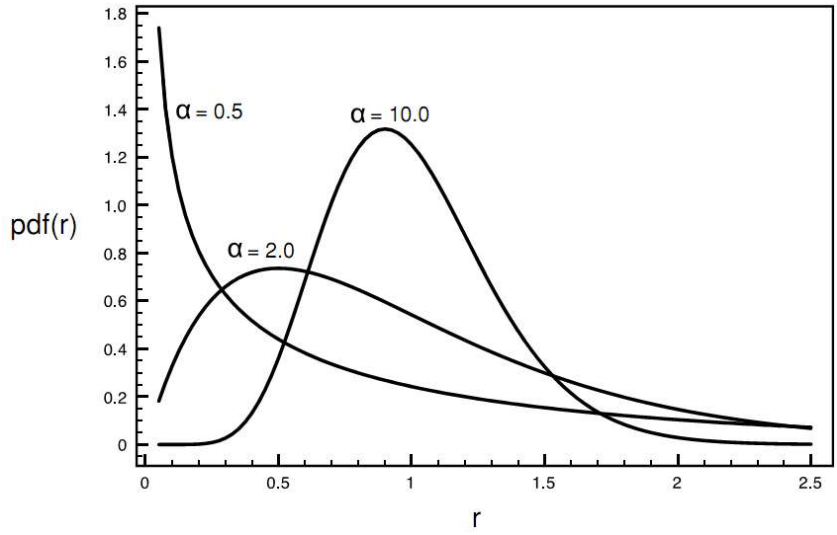
1.3.1 Mô hình tốc độ biến đổi

Mô hình tốc độ biến đổi là giải pháp đơn giản, dễ tích hợp và được sử dụng phổ biến để mô tả sự không đồng nhất về tốc độ thay thế r giữa các vị trí sử dụng một phân phối gamma (Γ) với kỳ vọng 1.0 và phương sai $1/\alpha$ ($\alpha > 0$) [35]. Mỗi vị trí trên trình tự được gán một giá trị gọi là tham số tốc độ tính theo công thức:

$$Pdf(r) = \frac{\alpha^{\alpha} r^{\alpha-1}}{e^{\alpha r}\Gamma(\alpha)} \tag{1.12}$$

Trong đó $\Gamma(\alpha) = \int_0^{\infty} e^{-t} t^{\alpha-1} dt$.

Sự thay đổi của tốc độ thay thế r được điều khiển bằng cách điều chỉnh giá trị của tham số điều khiển hình dạng α (Hình 1. 4). Khi sử dụng giá trị $\alpha > 1$, phân phối có



Hình 1. 4 Hàm mật độ xác suất (pdf) của phân phối gamma với tham số α khác nhau

dạng hình chuông, với giá trị $\alpha < 1$, phân phối có dạng hình chữ L. Giá trị α nhỏ thể hiện tính không đồng nhất về tốc độ biến đổi cao hơn so với giá trị α lớn.

Trong thực tế, hàm phân phối gamma rời rạc thường được dùng thay cho hàm phân phối liên tục để tăng tốc độ tính toán. Hàm phân phối rời rạc sử dụng k nhóm tốc độ với xác suất đều nhau. Giá trị trung bình hoặc trung vị của mỗi nhóm được sử dụng cho tất cả các tốc độ trong nhóm đó. Khi đó giá trị khả năng của sắp hàng D với cây T và mô hình Q được tính như sau:

$$L(Q,T,\alpha|D) = \prod_{i=1}^{l}\left(\frac{1}{k}\sum_{h=1}^{k} L(\Gamma(\alpha,h)Q,T|d_i)\right) \tag{1.13}$$

Theo phân tích trong [35], sử dụng bốn nhóm tốc độ là phù hợp để tính nhanh và đủ tốt để xấp xỉ mô hình phân phối liên tục. Tuy nhiên, các nhà nghiên cứu đã chỉ ra vấn đề của giải pháp này là chỉ quan tâm tới khía cạnh tốc độ chứ không bao gồm quá trình tiến hóa. Do vậy không thực sự phù hợp với những bộ dữ liệu mà các vị trí trên trình tự được kỳ vọng là tuân theo các quá trình tiến hóa khác nhau và chưa biết trước [36]. Một số giải pháp đã được đề xuất để biểu diễn không chỉ tốc độ tiến hóa khác

nhau tại các vị trí khác nhau mà cả quá trình tiến hóa cũng có thể khác nhau. Các phương pháp này được chia thành hai nhóm: sử dụng mô hình đa ma trận [36], [38] hoặc sử dụng mô hình lược đồ phân vùng [32], [34], [37].

1.3.2 Mô hình đa ma trận

Dựa trên cách tính giá trị khả năng, để thể hiện sự khác nhau về mô hình tiến hóa giữa các vị trí, thay vì chỉ sử dụng một ma trận, *mô hình đa ma trận* sử dụng nhiều ma trận thay thế Q khác nhau để tính giá trị khả năng tại mỗi vị trí.

Khi đó, giá trị khả năng của mỗi vị trí d_i trong sắp hàng chịu tác động tổng hợp của các ma trận được dùng, mỗi ma trận đi kèm với một trọng số điều khiển mức độ ảnh hưởng [42]. Tức là, mô hình đa ma trận xác định bởi hai thành phần:

- J ma trận thay thế $\mathbf{Q} = \{Q_1, Q_2, \dots, Q_J\}$
- Vectơ trọng số $\boldsymbol{w} = \{w_1, w_2, w_3, w_4 \mid \sum_j w_j = 1\}$ - w_j là trọng số của ma trận thay thế Q_j.

Với mô hình đa ma trận, giá trị khả năng của mỗi vị trí được tính riêng với từng mô hình đơn ma trận, sau đó nhân với trọng số và tổng hợp theo công thức như sau:

$$L(\{Q_1, Q_2, \dots, Q_J\}, \mathrm{T}, \{w_1, w_2, \dots, w_j\}|D) = \prod_i \left(\sum_j w_j L(Q_j, \mathrm{T}|d_i) \right) \qquad (1.14)$$

LG4M [42] là một ví dụ về mô hình đa ma trận. Mô hình này sử dụng bốn ma trận thay thế khác nhau $\mathbf{Q} = \{Q_1, Q_2, Q_3, Q_4\}$, kết hợp với mô hình phân phối tốc độ rời rạc theo công thức (2.11); mô hình phân phối tốc độ gồm 4 nhóm với trọng số mỗi nhóm bằng $1/4$. Khi đó, giá trị khả năng của sắp hàng với mô hình $\mathbf{Q}$ và cây T với tham số hình dạng α của phân phối Γ là:

$$L(\mathbf{Q}, T, \alpha|D) = \prod_{i=1}^{l} \left(\frac{1}{4} \sum_{k=1}^{4} L(\Gamma(\alpha, k) Q_k, T|d_i) \right) \qquad (1.15)$$

Mô hình LG4X [26] thay thế mô hình phân phối Γ bằng tham số tốc độ để thu được dạng tổng quát hơn. Cụ thể, mô hình bao gồm tập bốn ma trận **Q**, mỗi ma trận được gán trọng số w_j và tốc độ ρ_j; các tham số w_j, ρ_j được ước lượng từ dữ liệu và thỏa mãn hai điều kiện $\sum_{j=1}^{4} w_j = 1$ và $\sum_{j=1}^{4} w_j \rho_j = 1$. Giá trị khả năng được tính như sau:

$$L(\mathbf{Q}, T, \mathbf{P} = \{\rho_1, \rho_2, \rho_3, \rho_4\}, \mathbf{W} = \{w_1, w_2, w_3, w_4\} | D) = \prod_i \left(\sum_j w_j L(\rho_j Q_j, T | d_i) \right) \quad (1.16)$$

Kết quả thực nghiệm cho thấy các mô hình đa ma trận như LG4M và LG4X giúp xây dựng cây phân loài cực đại khả năng tốt hơn khi sử dụng mô hình đơn ma trận thông thường [26], [27]. Tuy nhiên thời gian tính toán của LG4M, LG4X nói riêng và các mô hình đa ma trận nói chung lâu hơn so với mô hình đơn. Trong mô hình đa ma trận, việc tăng J giúp tăng khả năng phân hóa của mô hình tiến hóa tổng hợp đồng thời tăng khối lượng tính toán cần thực hiện (cần tính tác động của tất cả J mô hình lên mỗi vị trí sau đó tính giá trị tổng hợp). Đây là trở ngại chính khi sử dụng phương pháp này, đặc biệt là khi tính toán cho dữ liệu lớn và rất lớn.

1.3.3 Mô hình lược đồ phân vùng

Mô hình lược đồ phân vùng, ngược lại với mô hình đa ma trận, chỉ dùng một mô hình phù hợp nhất tại mỗi vị trí, nhưng mô hình được thay đổi tùy vào vị trí khác nhau. Mặc dù ý tưởng của phương pháp là hợp lý, việc xác định mô hình phù hợp nhất từng vị trí trên sắp hàng là không khả thi do vậy mô hình lược đồ phân vùng tạo các nhóm vị trí có mô hình tiến hóa tương đồng, nghĩa là những vị trí này có thể coi là tiến hóa dưới mô hình giống nhau. Cụ thể:

Với một sắp hàng D, một "*lược đồ phân vùng*" ('partitioning scheme') của D, ký hiệu là **PS**, là một phân hoạch trên tập vị trí của sắp hàng D. Nghĩa là **PS** bao gồm các tập con khác rỗng, không giao nhau, mỗi tập con là một tập hợp các vị trí của sắp hàng sao cho mỗi vị trí thuộc một và chỉ một tập con. Mỗi tập con này còn được gọi là một phân vùng/tập con (partition/subset) vị trí.

Ký hiệu $\mathbf{PS} = \{S_1, \dots, S_k\}$ là lược đồ phân vùng của D chứa k phân vùng, trong đó phân vùng S_i có l_i vị trí; s_{ij} là giá trị của vị trí thứ j trong phân vùng S_i; mô hình thay thế tương ứng của phân vùng là Q_i và mô hình tốc độ biến đổi là V_i. Giá trị khả năng của sắp hàng được xác định lại bằng tích các giá trị khả năng của tất cả các phân vùng (**Q** và **V** là tập mô hình thay thế và tập mô hình tốc độ biến đổi cho các phân vùng trong lược đồ **PS**):

$$L(D|T, \mathbf{Q}, \mathbf{V}) = \prod_{i=1}^{k} \prod_{j=1}^{l_i} L(s_{ij}|T, Q_i, V_i) = \prod_{i=1}^{k} \prod_{j=1}^{l_i} P(s_{ij}|T, Q_i, V_i) \quad (1.17)$$

Mục tiêu của các thuật toán phân hoạch sắp hàng là tìm một lược đồ phân vùng thể hiện tốt nhất sự không đồng nhất trong quá trình biến đổi giữa các nucleotit/axit amin tại các vị trí khác nhau trên sắp hàng. Hàm mục tiêu của các mô hình được đánh giá thông qua cây phân loài cực đại khả năng cho sắp hàng D xây dựng được khi sử dụng những mô hình này.

Khi sử dụng mô hình lược đồ phân vùng, việc lựa chọn mô hình thay thế để tính giá trị khả năng cho từng vị trí trên sắp hàng sẽ bao gồm hai bước. Bước 1 tạo lược đồ phân vùng (bằng các thuật toán phân hoạch vị trí trong sắp hàng); bước 2 chọn mô hình thay thế phù hợp nhất với mỗi nhóm. Ý tưởng của mô hình lược đồ phân vùng có thể áp dụng cho trường hợp thông thường - với giả thiết rằng quá trình tiến hóa của tất cả các vị trí đều đồng nhất – khi đó ta bỏ qua bước một, chỉ thực hiện bước hai vì chỉ có một nhóm vị trí duy nhất. Sử dụng lược đồ phân vùng thỏa mãn được cả hai điều kiện:

- Thể hiện được tính không đồng nhất về quá trình biến đổi thông qua việc gán các mô hình thay thế khác nhau cho mỗi phân vùng vị trí.
- Không làm tăng khối lượng tính toán khi tính giá trị khả năng, mỗi vị trí chỉ tính một lần với các mô hình thay thế khác nhau.

Bên cạnh đó, việc tính giá trị khả năng cho sắp hàng sử dụng lược đồ phân vùng được thực hiện dễ dàng. Do vậy lược đồ phân vùng được hỗ trợ trong nhiều phần mềm phổ biến như PartitionFinder [43], RAxML-NG[21], IQTREE[22].

1.3.4 Một số thuật toán phân hoạch sắp hàng

Ban đầu, việc phân hoạch sắp hàng chủ yếu dựa trên các đặc trưng sinh học của bộ dữ liệu, ví dụ như, khi giả thiết rằng tốc độ tiến hóa trong các gen khác nhau là khác nhau, mỗi đoạn trình tự tương ứng với một đoạn gen khác nhau tạo thành một phân vùng; một cách phân hoạch nhỏ hơn là sử dụng vị trí trong codon của từng gen (tức là với các vị trí trong mỗi gen lại phân ra ba nhóm tương ứng với vị trí thứ nhất, thứ hai và thứ ba trong các codon định nghĩa gen đó). Mặc dù các nucleotit tại cùng vị trí trong codon được xác nhận là có nhiều đặc điểm chung, giả thiết rằng chúng luôn cùng mô hình tiến hóa không phải lúc nào cũng đúng đắn về mặt sinh học. Thực tế là cùng một vị trí codon, các vị trí khác nhau vẫn có thể tiến hóa với tốc độ khác nhau và biến đổi theo mẫu khác nhau nên việc phân hoạch bằng các thông tin di truyền không phải lúc nào cũng đúng đắn [32]. Một số các đặc trưng khác như cấu trúc thùy (stem-loop) của trình tự RNA hay cấu trúc bậc cao của chuỗi cũng cung cấp thông tin hữu ích để phân hoạch tập vị trí [44]–[46].

Phát triển từ ý tưởng phân hoạch bằng các thông tin sinh học, nhóm tác giả Lanfear và các cộng sự đề xuất các phương pháp sử dụng các lược đồ phân vùng như trên làm lược đồ khởi tạo, từ đó cải tiến để thu được các lược đồ tốt hơn [37], [47]. Trong các thuật toán này, người dùng định nghĩa trước các *khối* vị trí dựa trên hiểu biết của mình về dữ liệu. Tập hợp các khối này tạo thành lược đồ phân vùng khởi tạo. Sau đó, thuật toán tiến hành duyệt và nhập các khối có mô hình tiến hóa tương tự với nhau để một mặt làm giảm số khối, mặt khác, làm tăng độ tốt của lược đồ. Cách làm này được đánh giá là giúp cải thiện độ chính xác của cây [48] và được sử dụng rộng rãi trong nhiều nghiên cứu liên quan.

Tuy nhiên các phương pháp trên không phù hợp khi có ít hoặc không có thông tin về các đặc điểm sinh học và tiến hóa của dữ liệu. Ngoài ra, một số nghiên cứu cũng chỉ ra là ngay cả khi sử dụng những đặc điểm này, không phải lúc nào giả thiết các vị trí trong cùng nhóm tạo bởi một gen cũng đúng về mặt sinh học [26]. Do vậy, các phương pháp tính toán dựa trên phân tích dữ liệu sắp hàng là cần thiết để có thể xác định một

lược đồ phân vùng mà không phụ thuộc vào các thông tin này. Một số thuật toán phân hoạch tự động có thể kể đến là *k*-means lặp, RatePartition [30], [32]. Các thuật toán này sử dụng tốc độ tiến hóa tại mỗi vị trí để phân hoạch tập vị trí dựa trên giả định rằng các vị trí có tốc độ tiến hóa tương đồng nên nằm trong cùng một phân vùng (tốc độ tiến hóa sẽ được đề cập chi tiết ở phần 1.4).

Thuật toán *k*-means lặp

Một ví dụ của phương pháp phân hoạch tự động là thuật toán *k*-means lặp của nhóm tác giả Frandsen [30]. Thuật toán thực hiện lặp đi lặp lại việc phân cụm các vị trí thành các tập hợp con bằng thuật toán *k*-means dựa trên tốc độ tiến hóa tại mỗi vị trí.

Cụ thể, với sắp hàng D, thuật toán phân hoạch sắp hàng để tìm lược đồ tốt nhất (ký hiệu là **PS**) gồm 4 bước (Thuật toán 1. 1):

Bước 1: Khởi tạo cây ban đầu T cho sắp hàng D

Bước 2: Tạo lược đồ khởi tạo chỉ gồm một phân vùng toàn bộ D và chọn mô hình thay thế phù hợp nhất cho phân vùng đó.

Bước 3: Tính giá trị hàm mục tiêu cho lược đồ phân vùng hiện tại.

Bước 4: Với mỗi phân vùng trong lược đồ phân vùng hiện tại: kiểm tra xem có nên chia nhỏ hay không:

- Tính tốc độ tiến hóa cho từng vị trí của phân vùng
- Sử dụng thuật toán *k*-means để chia đôi tập vị trí dựa trên tốc độ
- Chọn mô hình phù hợp nhất cho các phân vùng mới
- Tính hàm mục tiêu cho lược đồ phân vùng tạm thời (là lược đồ được tạo ra bằng cách thay phân vùng trước khi chia bằng hai phân vùng sau khi chia). Hàm mục tiêu được sử dụng ở đây là độ đo AICc được trình bày cụ thể trong phần sau.
- Nếu hàm mục tiêu tốt hơn thì đánh dấu phân vùng đang xét là cần chia.

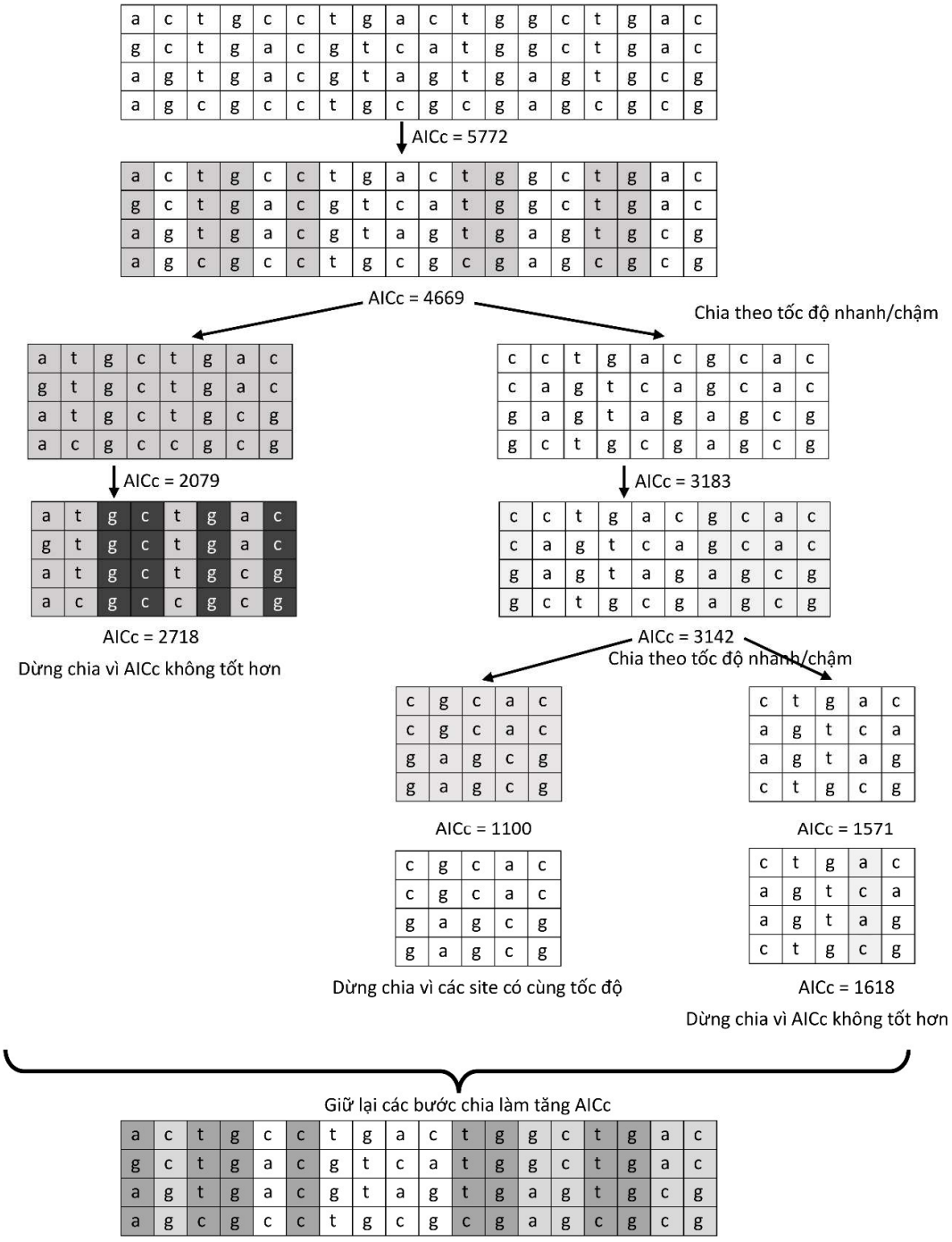
Bước 5: Thuật toán kết thúc khi không còn phân vùng nào cần chia. Ngược lại, tạo ra lược đồ phân vùng mới, trong đó mỗi phân vùng được đánh dấu là cần chia sẽ được thay bằng hai phân vùng chia được tương ứng trong bước 4 và quay lại bước 4.

Hình 1. 5 minh họa một ví dụ cụ thể của thuật toán *k*-means vừa trình bày. Mặc dù ý tưởng của thuật toán rất hay, nghiên cứu của Baca và cộng sự trên một số cách phân hoạch khác nhau cho bộ dữ liệc các loài bọ cánh cứng thủy sinh đã chỉ ra rằng cách phân hoạch như trên tạo ra một phân vùng chứa tất cả các vị trí bất biến (là các vị trí mà trên đó giá trị của nucleotit/axit amin không biến đổi trên tất cả các trình tự) gọi là phân vùng bất biến. Phân vùng bất biến có thể rất lớn và gây ảnh hưởng đến tính đúng đắn của quá trình xây dựng cây phân loài [49]. Đây là vấn đề nghiêm trọng của thuật toán vì một vị trí có thể là bất biến trong tập dữ liệu được nghiên cứu, nhưng sẽ là vị trí có biến đổi khi xét trên tập dữ liệu lớn hơn. Hệ quả là phân vùng này có thể tăng giá trị khả năng của cây nhưng đồng thời cũng tăng khả năng cây bị sai lệch vì sử dụng sắp hàng con có các trình tự giống hệt nhau [32], [49]. Do đó, kể từ năm 2017, thuật toán *k*-means được khuyến nghị không tiếp tục sử dụng cho dữ liệu DNA và axit amin.

Algorithm Phương pháp $k-means$ lặp để phân hoạch sắp hàng

input: Sắp hàng D
output: Lược đồ phân vùng của **PS** của D
procedure ITERATIVE $k-means(D)$
    Tạo cây ban đầu T cho D // Khởi tạo
    Phân vùng khởi tạo $\mathbf{PS} = \{D\}$
    Tìm mô hình thay thế phù hợp nhất cho D
    Tính AICc(**PS**)
    Khởi tạo danh sách nhóm có thể chia $LS = \{D\}$
    while $LS \neq \emptyset$ **do**
        for each $S_i \in LS$ **do**
            $LS = LS \backslash \{S_i\}$
            Tính tốc độ r_j cho vị trí j trong S_i
            $S_i \xrightarrow[k-means]{r_i} S_{i1}; S_{i2}$
            Tìm mô hình phù hợp nhất M_1 cho (S_1, T) và M_2 cho (S_2, T)
            $\mathbf{PS'} = \mathbf{PS} \backslash \{S_i\} \cup \{S_{i1}\} \cup \{S_{i2}\}$
            Tính AICc(**PS'**)
            if AICc(**PS'**)<AICc(**PS**) **then**
                $LS = LS \cup \{S_{i1}\} \cup \{S_{i2}\}$
                $\mathbf{PS} = \mathbf{PS'}$
            end if
        end for
    end while
end procedure

Thuật toán 1. 1 Thuật toán $k-means$ lặp để phân hoạch sắp hàng.



Hình 1. 5 Ví dụ về thuật toán k-means lặp tìm lược đồ phân vùng tự động [30]

Algorithm Thuật toán RatePartition để phân hoạch sắp hàng

input: Danh sách tốc độ tiến hóa của từng vị trí trong sắp hàng
$\mathbf{R} = \{r_1, ..., r_l\}$, hệ số phân chia $d > 1$.
output: Phân hoạch của tập vị trí $\{S_1, S_2, ...\}$.
procedure RatePartition($\mathbf{R}, d$)
    Khởi tạo: $i = 1; rMin = min\{\mathbf{R}\}; riSlow = max\{\mathbf{R}\}; riFast = rMin$
    while Còn $\geq 10\%$ số vị trí chưa thuộc nhóm nào **do**
        $S_i = \emptyset$
        if $i == 1$ **then**
            $riFast = riSlow - \frac{riSlow - rMin}{d}$
        else
            $riSlow = riFast$
            $riFast = riSlow - \frac{riSlow - rMin}{d + i \times 0.3}$
        end if
        for each j **do**
            if $(r_j > riFast) \&\& (r_j \leq riSlow)$ **then**
                $S_i = S_i \cup \{i\}$
            end if
        end for
        $i = i + 1$
    end while
    Tất cả các vị trí còn lại tạo thành nhóm cuối cùng của phân hoạch.
end procedure

Thuật toán 1. 2 Thuật toán RatePartition để phân hoạch tập vị trí

Thuật toán RatePartition

Để tránh việc tạo phân vùng bất biến trong lược đồ, tác giả Rota và các cộng sự đã đề xuất thuật toán RatePartition [32]. Thuật toán gồm hai bước:

Bước 1: Tính tốc độ tiến hóa cho từng vị trí trong sắp hàng

Bước 2: Phân hoạch vị trí theo tốc độ

- Sử dụng một công thức đơn giản để phân hoạch tốc độ tiến hóa đã tính ở trên.
- Lược đồ phân vùng thu được là tập các vị trí có tốc độ tương ứng với các đoạn tốc độ trong phân hoạch tốc độ.

Công thức phân hoạch tốc độ tiến hóa của RatePartition được thiết kế để thêm các vị trí có tốc tiến hóa thấp vào nhóm bất biến. Công thức chia khoảng tốc độ được điều chỉnh bởi hệ số phân chia $d \in R, d > 1$. Với d càng lớn, số nhóm chia được càng nhiều.

Cho trước danh sách tốc độ tiến hóa tại từng vị trí, việc phân hoạch tập vị trí trong RatePartition được thực hiện tại bước 2, cách tính cụ thể như trong Thuật toán 1. 2.

Trong cả hai thuật toán, thông tin được sử dụng để phân hoạch sắp hàng là tốc độ biến đổi tại mỗi vị trí. Theo các các giả, có thể sử dụng bất kỳ loại tốc độ nào trong tính toán, tuy nhiên phương pháp tính tốc độ không sử dụng cây cụ thể (như TIGER - Tree Independent Generation of Evolutionary Rates) sẽ tốt hơn những phương pháp sử dụng cây.

1.4 Tốc độ tiến hóa tại mỗi vị trí trên sắp hàng

Một cách tổng quát, tốc độ tiến hóa là tốc độ mà các loài sinh vật thay đổi và thích nghi với môi trường thể hiện qua quá trình tiến hóa. Với phạm vi dữ liệu trong một sắp hàng, tốc độ tiến hóa là tương đối giữa các vị trí; một số vị trí có thể phát triển với tốc độ nhanh hơn trong khi có vị trí phát triển chậm hơn. Ví dụ, nghiên cứu [50] đã chỉ ra rằng xét trên một đoạn gen bất kỳ, trong phần lớn trường hợp, ba vị trí của

nucleotit trên các codon có tốc độ tiến hóa tuân theo quy luật là vị trí thứ hai tiến hóa chậm hơn vị trí thứ nhất, và vị trí thứ nhất lại tiến hóa chậm hơn vị trí thứ ba.

Tốc độ tiến hóa có ứng dụng trong nhiều bài toán liên quan tới đánh giá dữ liệu như xác định các vị trí có xu hướng tiến hóa tương đồng; xác định vị trí nào trên sắp hàng là vị trí có thể quan trọng trong phân tích phát sinh loài, hoặc sử dụng để loại bỏ các vị trí có khả năng là nhiễu. Do vậy có nhiều nghiên cứu liên quan đến bài toán ước lượng tốc độ tiến hóa cho từng vị trí [51]–[55]. Các phương pháp này thường có điểm chung là cần sử dụng một cây phân loài của sắp hàng đang xét [51], [53], [54]. Tuy nhiên, tốc độ ước lượng được bị ảnh hưởng bởi cấu trúc cây được sử dụng; tức là, khi cấu trúc cây hay độ dài nhánh thay đổi, giá trị tốc độ của các vị trí cũng thay đổi theo. Do vậy phương pháp ước lượng tốc độ không sử dụng cây là cách tính được khuyên dùng trong các nghiên cứu phương pháp phân hoạch sắp hàng [30], [32]. Điển hình trong số này là TIGER [52], thuật toán ước lượng tốc độ bằng cách phân tích sự giống nhau của các mẫu trên từng vị trí. Thuật toán bao gồm 3 bước như trong Thuật toán 1. 3.

- Bước 1: Với mỗi vị trí trong sắp hàng, TIGER tạo một phân hoạch trình tự, trong mỗi tập con của phân hoạch là các trình tự có cùng trạng thái trong vị trí đang xét của sắp hàng. Ví dụ, vị trí i có nội dung là xâu "GGCGAA" thì phân hoạch vị trí của nó $P(i) = \{\{1, 2, 4\}, \{3\}, \{5, 6\}\}$ vì các kí tự thứ 1, 2 và 4 trong xâu là G, tương tự như vậy, ký tự thứ 3 là C và hai kí tự thứ 5 và 6 là A.
- Bước 2: Với mỗi vị trí i, thực hiện tính điểm "tương đồng phân hoạch" với từng vị trí j khác trong sắp hàng:

$$pa(i,j) = \frac{\sum_{x \in P(j)} a\big(x, P(i)\big)}{|P(j)|} \qquad (1.18)$$

Với $a\big(x, P(i)\big) = \begin{cases} 1 & \text{nếu } \exists A \in P(i) | x \subseteq A \\ 0 & \text{ngược lại} \end{cases}$

Algorithm Thuật toán TIGER ước lượng nhanh tốc độ tiến hóa tại mỗi vị trí không phụ thuộc cây

input: Sắp hàng D có n trình tự, độ dài l
output: Danh sách tốc độ tiến hóa r_i của vị trí thứ i trong sắp hàng
procedure TIGER(D)
    for each vị trí i **do**
        Tạo $p(i)$ phân hoạch tập trình tự có cùng trạng thái
    end for
    for each $(i \neq j)$ **do**
        Tính giá trị phân hoạch tương đồng của hai vị trí: $pa(i,j)$
    end for
    for each vị trí i **do**

$$r_i = \frac{\sum_{i \neq j} pa(i,j)}{n-1}$$

    end for
end procedure

Thuật toán 1. 3 Thuật toán TIGER ước lượng quá trình tiến hóa vị trí

Ở đây giá trị tương đồng phân hoạch giữa hai vị trị i và j - $pa(i,j)$ được tính bằng xác suất một tập trong phân hoạch trình tự của vị trí j xuất hiện trong một tập phân hoạch trình tự của vị trí i; do vậy, $pa(i,j)$ và $pa(j,i)$ là độc lập và thường nhận giá trị khác nhau.

Ví dụ: Hai vị trí i và j có nội dung: "GGCGAA" và "AAACCC" thì

$P(i) = \{\{1,2,4\},\{3\},\{5,6\}\}$; $P(j) = \{\{1,2,3\},\{4,5,6\}\}$

Khi đó $a(\{1,2,4\}, P(j)) = 0$; $a(\{3\}, P(j)) = 1$; và $a(\{5,6\}, P(j)) = 1$

Trong khi $a(\{1,2,3\}, P(i)) = 0$ và $a(\{4,5,6\}, P(i)) = 0$

Do vậy $pa(i,j) = \frac{0}{2} = 0$ còn $pa(j,i) = \frac{2}{3}$

- Bước 3: Tính giá trị tốc độ tiến hóa r_i cho mỗi vị trí i là trung bình điểm tương đồng phân hoạch của vị trí i với tất cả các vị trí khác:

$$r_i = \frac{\sum_{j \neq i} pa(i,j)}{l-1} \quad (1.19)$$

Với mỗi vị trí, TIGER cần tính điểm phân hoạch tương đồng với tất cả các vị trí còn lại trong sắp hàng, do vậy độ phức tạp của phương pháp là hàm bình phương độ dài $O(\mathrm{n} \times l^2)$. Vì vậy việc tính tốc độ tiến hóa cho những sắp hàng có độ dài lên tới hàng triệu vị trí là vô cùng tốn kém về mặt thời gian.

1.5 Các phương pháp đánh giá mô hình

Mục tiêu cốt lõi của việc đưa ra các mô hình mới (trong phạm vi luận án là mô hình thay thế axit amin và mô hình phân hoạch sắp hàng) là để xây dựng cây phân loài chính xác hơn. Do vậy, để so sánh các mô hình, trước hết cần xây dựng cây phân loài cho cùng một bộ dữ liệu bằng các mô hình cần so sánh. Sau đó so sánh giá trị khả năng hay sử dụng một số độ đo trên các cây phân loài đã xây dựng; ngoài ra, so sánh cấu trúc giữa các cây được thực hiện để đánh giá tác động của các mô hình [12], [27], [56].

1.5.1 So sánh giá trị khả năng của cây phân loài xây dựng bằng phương pháp cực đại khả năng

Gọi M_1, M_2 là hai mô hình thay thế nucleotit/axit amin hoặc mô hình phân hoạch sắp hàng cần so sánh. Với sắp hàng D, phương pháp cực đại khả năng được sử dụng để xây dựng cây T_1, T_2 tương ứng với mô hình M_1 và M_2. Giá trị khả năng của hai cây tương ứng là $L(T_1)$ và $L(T_2)$ – chính xác là lôgarít tự nhiên của giá trị khả năng để so sánh vì giá trị khả năng thường rất nhỏ. Cách đơn giản nhất để so sánh hai cây, hay chính là so sánh hai mô hình là so sánh giá trị khả năng:

- Nếu $L(T_1) > L(T_2)$: mô hình M_1 tốt hơn mô hình M_2.
- Ngược lại, nếu $L(T_1) < L(T_2)$: mô hình M_2 tốt hơn mô hình M_1.

Trong thực tế, giá trị khả năng có thể có sai lệch do các tham số ngẫu nhiên trong quá trình tính toán. Dẫn đến việc một cây có giá trị khả năng cao hơn không hoàn toàn

chắc chắn là cây tốt hơn. Để kiểm tra xem một cây có giá trị khả năng cao hơn có chắc chắn là cây tốt hơn hay không, ta có thể thực hiện các kiểm tra thống kê như KH [57] hay SH [58]. Các kiểm tra thống kê này sẽ thực hiện lấy mẫu lại có thay thế trên sắp hàng ban đầu nhiều lần để tạo ra các sắp hàng gọi là sắp hàng bootstrap; sau đó xây dựng cây phân loài cho các sắp hàng bootstrap này theo cùng cách xây dựng cây phân loài cho sắp hàng ban đầu và tính toán xem cây nào trong số các cây được kiểm tra là cây tốt hơn. Trong số các phương pháp kiểm tra, phương pháp gần đúng (AU – approximately unbiased) [59] được đánh giá là phương pháp cho kết quả tốt nhất.

1.5.2 Các độ đo AIC và BIC

Gọi **M** là tập hợp các mô hình đã sử dụng để xây dựng cây phân loài. **M** có thể bao gồm mô hình thay thế nucleotit, axit amin; mô hình tốc độ biến đổi; mô hình đa ma trận; mô hình lược đồ phân vùng … Giống như các công trình nghiên cứu khác [12], [32], [47]; luận án dùng ba độ đo thường dùng là AIC [60], AICc [61] và BIC [62] để đánh giá độ phù hợp của mô hình được sử dụng từ đó xác định xem mô hình nào có độ phù hợp cao nhất. Ba tiêu chuẩn có điểm chung là đều được tính dựa trên giá trị khả năng. Tuy nhiên do mỗi mô hình có số tham số tự do khác nhau; nếu chỉ xét giá trị khả năng thì việc lựa chọn sẽ có xu hướng chọn mô hình có nhiều tham số nhất. Hệ quả là làm phức tạp quá trình tính toán. Với cả ba độ đo, mô hình tốt hơn có giá trị độ đo thấp hơn.

Độ đo AIC (Akaike Information Criterion) của mô hình **M** được tính như sau:

$$\mathrm{AIC}(\mathbf{M}|D) \;=\; 2k \,-\, 2ln\big(L(T|M,D)\big) \tag{1.20}$$

Với k là số tham số của mô hình và $L(T|M,D)$ là giá trị khả năng của cây T được xây dựng với mô hình **M** và sắp hàng D.

Tuy nhiên, theo [61] AIC có xu hướng chọn mô hình nhiều tham số hơn, có thể là nguyên nhân của hiện tượng quá khớp (over-fiting); các tác giả đã đưa ra một phiên bản mới của AIC, là độ đo AICc (Corrected Akaike Information Criterion), được hiệu chỉnh cho dữ liệu cỡ nhỏ:

$$AIC_c = AIC + \frac{2k^2 + 2k}{n - k - 1} \quad (1.21)$$

Khi kích thước mẫu n nhỏ và số lượng tham số k lớn, giá trị AICc sẽ cao hơn, do vậy độ đo ưu tiên chọn mô hình có ít tham số hơn. Tuy nhiên, độ đo AICc bản chất là độ đo AIC bổ sung thêm một thành phần phạt là hàm của biến n và k. Do vậy, nếu các mô hình cần so sánh có cùng cỡ dữ liệu và số lượng tham số thì tương quan về độ lớn của hai độ đo trên các mô hình là giống nhau. Ngoài ra, nếu số lượng dữ liệu nhỏ hơn số lượng tham số (giá trị phạt âm) hoặc lượng dữ liệu rất lớn so với số tham số ($n \gg k$, giá trị phạt xấp xỉ 0) thì việc sử dụng AICc cũng không có ý nghĩa. Nhìn chung giá trị AICc được sử dụng để đánh giá độ chính xác của một mô hình thống kê khi mô hình có số lượng quan sát ít.

Độ đo BIC (Bayesian Information Criterion) dựa trên lý thuyết tiên nghiệm Bayesian. BIC đánh giá mức độ phù hợp của một mô hình dựa trên khái niệm "độ phức tạp" của mô hình. BIC giả định rằng mô hình đúng nhất sẽ có tham số ít nhất và có thể giải thích dữ liệu tốt nhất. Giá trị BIC được tính bởi công thức:

$$BIC(M|D) = 2kln(n) - 2ln\big(L(T|M,D)\big) \quad (1.22)$$

Khi so sánh các mô hình thay thế (trong chương 2), ngoại trừ mô hình đa ma trận LG4X cần thêm 6 tham số cho các tốc độ tự do và trọng số, với các mô hình còn lại việc xây dựng cây đều có cùng các tham số là độ dài nhánh trong cây và một tham số α cho phân phối gamma do vậy các so sánh sử dụng độ đo AIC. Trong chương 3 và chương 4, khi sử dụng các lược đồ khác nhau sẽ có số lượng phân vùng khác nhau do vậy số tham số trong mỗi mô hình là khác nhau. Trong hai chương này các thực nghiệm sử dụng độ đo AICc và BIC.

1.5.3 So sánh cấu trúc cây

Trên dữ liệu thực, việc so sánh cấu trúc cây không đưa ra kết luận mô hình nào là tốt hơn mà đánh giá tác động của mô hình tiến hóa tới kết quả xây dựng cây phân loài.

Hai cây có cấu trúc càng giống nhau thể hiện hai mô hình dùng để xây dựng cây có độ tương đồng càng cao.

Sự khác nhau về cấu trúc giữa hai cây được tính bằng khoảng cách Robinson-Fould giữa hai cây đó [14], [56]. Khoảng cách ban đầu được định nghĩa:

$RF(T_1, T_2) =$ số phân nhánh thuộc một cây; không thuộc cây còn lại (1.23)

Tuy nhiên, giá trị này chỉ thể hiện được sự khác nhau mà không thể hiện được tỉ lệ khác nhau giữa hai cây (ví dụ cùng khác nhau 4 phân nhánh, nhưng trên cây chỉ có 5 nhánh thì hai cây rất khác nhau, còn nếu cây có hàng trăm nhánh thì cấu trúc hai cây về cơ bản là giống nhau). Do vậy trong [34], các tác giả đề xuất sử dụng khoảng cách Robinson-Fould được chuẩn hóa:

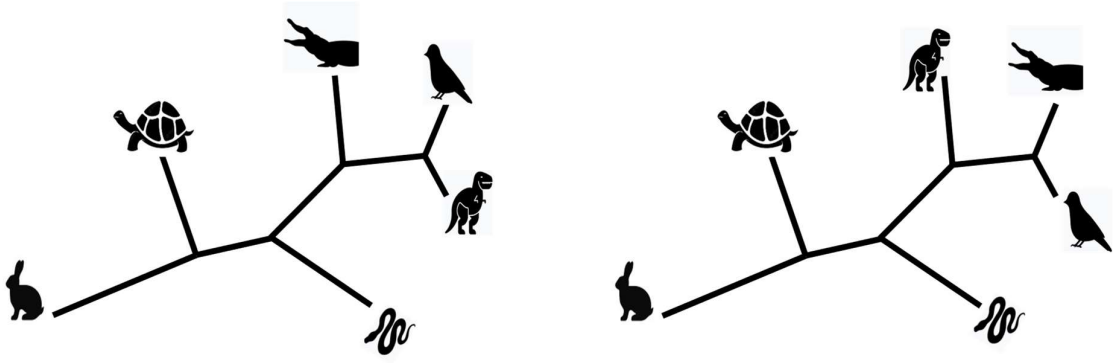
$$nRF(T_1, T_2) = \frac{RF(T_1, T_2)}{2(n-3)} \quad (1.24)$$

Trong công thức (2.17) n là số trình tự của sắp hàng, $2(n-3)$ là tổng số nhánh của hai cây (không có gốc) có thể khác nhau. Cụ thể, mỗi cây không có gốc có n lá sẽ có $2n-3$ phân nhánh, trong đó n phân nhánh bên ngoài tương ứng với cạnh nối với lá và $n-3$ phân nhánh trong tương ứng với cạnh trong. Các phân nhánh ngoài luôn có mặt ở cả hai cây, chỉ các phân nhánh trong có thể khác nhau. Khoảng cách Robinson-Foulds chuẩn hóa nhận giá trị từ 0 đến 1, biểu thị tỉ lệ giữa số phân nhánh chỉ có ở một trong hai cây trên tổng số phân nhánh của hai cây. Giá trị càng gần 0 thể hiện hai cây càng giống nhau, giá trị bằng 1 khi hai cây hoàn toàn khác nhau.

Lưu ý: khoảng cách Robinson-Fould chỉ tính toán trên các cây có cùng đối tượng nghiên cứu (trùng tên và số lượng); bên cạnh đó, khoảng cách không xét độ dài của các nhánh mà chỉ xét trên cấu trúc – mối quan hệ họ hàng giữa các loài trong cây.

Ví dụ, Hình 1. 6 minh họa hai cây phân loài không có gốc với cấu trúc khác nhau trên cùng tập 6 taxa. Với mỗi cạnh trong của cây, ta có một cách phân nhánh tập taxa. Tập phân nhánh của hai cây như trong Bảng 1. 7. Ta thấy hai phân nhánh thứ 3 của mỗi

cây không xuất hiện trong cây còn lại. Như vậy khoảng cách Robinson-Fould giữa hai cây là 2/6.



Hình 1. 6 Hai cây phân loài không gốc có cấu trúc khác nhau trên cùng tập taxa

Bảng 1. 7 Danh sách phân nhánh trên hai cây trong Hình 1. 6

Cây bên trái		**Cây bên phải**	
STT	***Phân nhánh***	***STT***	***Phân nhánh***
1	(thỏ, rùa),(rắn, cá sấu, khủng long, chim)	1	(thỏ, rùa),(rắn, khủng long, cá sấu, chim)
2	(thỏ, rùa, rắn),(cá sấu, khủng long, chim)	2	(thỏ, rùa, rắn),(khủng long, cá sấu, chim)
3	(thỏ, rùa, rắn, cá sấu),(khủng long, chim)	3	(thỏ, rùa, rắn, khủng long),(cá sấu, chim)

1.6 Kết luận

Dữ liệu trình tự DNA và axit amin chứa những thông tin quan trọng, việc nghiên cứu nguồn dữ liệu phong phú này nhằm xác định mối quan hệ giữa các loài trong tự nhiên là mục tiêu trung tâm của nhiều nghiên cứu. Trong quá trình xác định mối quan hệ giữa các loài thông qua việc xây dựng cây phân loài bằng phương pháp cực đại khả năng, việc lựa chọn mô hình thay thế phù hợp nhất có ảnh hưởng lớn đến kết quả tính toán và cây phân loài cực đại khả năng xây dựng được. Do vậy nhiều mô hình thay

thế đã được đề xuất để các nhà khoa học có thể lựa chọn được mô hình biểu diễn tốt nhất quá trình tiến hóa trên dữ liệu được nghiên cứu, phổ biến nhất là sử dụng ma trận có kích thước 4x4 hoặc 20x20 thể hiện sự thay thế giữa các cặp nucleotit hoặt axit amin. Các mô hình thay thế mới, các cách biểu diễn mới vẫn tiếp tục được nghiên cứu và đề xuất nhằm nâng cao khả năng biểu diễn dữ liệu so với các mô hình đã có.

Với những bộ dữ liệu lớn, với độ dài mỗi trình tự lên đến hàng trăm nghìn hay hàng triệu vị trí, vấn đề tiến hóa không đồng nhất tại các vị trí khác nhau trên trình tự cần được tính đến khi xây dựng cây, cụ thể là trong bước tính giá trị khả năng. Việc sử dụng lược đồ phân vùng để mô tả các nhóm vị trí có mô hình tiến hóa khác nhau là phương pháp hiệu quả về mặt tính toán để nâng cao tính đúng đắn của cây phân loài xây dựng được. Lược đồ phân vùng có thể xác định dựa trên hiểu biết về dữ liệu hoặc sử dụng các thuật toán để phân hoạch sắp hàng. Một số thuật toán tự động phân hoạch sắp hàng tìm lược đồ phân vùng đã được đề xuất và cho kết quả thực nghiệm tốt. Tuy nhiên các thuật toán này chỉ sử dụng tốc độ tiến hóa ước lượng tại mỗi vị trí để phân hoạch do vậy cần được nghiên cứu và cải tiến thêm.

Chương 2: Mô hình thay thế axit amin FLAVI cho Flavivirus

Nội dung chương 2 trình bày quá trình ước lượng mô hình thay thế axit amin FLAVI cho chi Flavivirus – là mô hình biểu diễn xác suất biến đổi giữa các axit amin trong dữ liệu thông tin di truyền của các loài trong chi Flavivirus - và kết quả thực nghiệm so sánh hiệu quả của mô hình FLAVI với một số mô hình được sử dụng rộng rãi hiện nay (bao gồm ba mô hình chung và bốn mô hình ước lượng riêng cho vi rút). Nội dung của chương được tổng hợp từ hai công trình nghiên cứu [CT1] và [CT4].

2.1 Giới thiệu

Mặc dù có ý kiến cho rằng việc lựa chọn mô hình tiến hóa không có ảnh hưởng quá nhiều đến kết quả xây dựng cây phân loài [63], nhiều nghiên cứu đã khẳng định việc lựa chọn mô hình thay thế axit amin phù hợp với quá trình tiến hóa của dữ liệu có ảnh hưởng lớn đến kết quả xây dựng cây phân loài bằng phương pháp cực đại khả năng [2], [9], [64]. Do vậy, các nghiên cứu đề xuất mô hình thay thế mới phù hợp với các nhóm sinh vật có đặc điểm tiến hóa khác nhau là cần thiết [65], [66].

Trong một vài năm trở lại đây, nhiều dịch bệnh liên tục bùng phát. Dịch bệnh thường xuyên gây ảnh hưởng nghiêm trọng tới sức khỏe người dân, y tế công cộng, và gián tiếp ảnh hưởng tới nền kinh tế [67]. Một trong những dịch bệnh xuất hiện hàng năm tại nhiều nơi trên thế giới là dịch sốt xuất huyết, tập trung chủ yếu ở khu vực nhiệt đới và cận nhiệt đới. Nguyên nhân gây ra bệnh sốt xuất huyết là do vi rút Dengue - một vi rút thuộc chi Flavivirus - gây ra. Vi rút này không lây trực tiếp từ người sang người mà thông qua vật chủ trung gian là muỗi vằn Aedes. Hiện nay có trên 100 quốc gia ghi nhận sốt xuất huyết là dịch. Theo ước tính của WHO, trên thế giới có 2.5 tỉ người sống trong vùng dịch, hàng năm có khoảng 50 triệu người mắc bệnh với khoảng 25 nghìn người chết vì vi rút Dengue [68]. Bên cạnh vi rút Dengue, một số vi rút khác

thuộc Flavivirus như vi rút Tây sông Nin (West Nile) hay vi rút Zika cũng nổi lên là những dịch bệnh nguy hiểm trong thời gian dần đây.

Tuy đã xuất hiện từ lâu, đến nay vẫn chưa có vắc xin hay thuốc đặc hiệu cho các bệnh gây ra bởi các vi rút trên. Do vậy, luận án đề xuất mô hình thay thế axit amin FLAVI cho nhóm vi rút trong chi Flavivirus với mục tiêu tăng cường hiểu biết về quá trình tiến hóa, tạo điều kiện cho các nghiên cứu về những vi rút này và góp phần mở ra triển vọng giảm thiểu tác hại của những dịch bệnh trên.

Mô hình được ước lượng bằng phương pháp cực đại khả năng, kết hợp với phương pháp chia nhỏ sắp hàng dựa trên cấu trúc cây FASTMG/ [69] để giảm bớt thời gian. Mô hình sau khi ước lượng được so sánh với một số mô hình thay thế khác (bao gồm cả mô hình chung và mô hình thay thế axit amin trên các vi rút khác). Các kết quả thực nghiệm chỉ ra rằng FLAVI cho kết quả tốt hơn hẳn các mô hình hiện có khi phân tích trình tự protein của các vi rút thuộc chi này.

2.2 Phương pháp

Như đã đề cập trong 1.2 , với đầu vào là một tập N sắp hàng axit amin ký hiệu là $\mathbf{D} = \{D_1, \dots, D_N\}$. Đầu ra là: Mô hình thay thế axit amin Q gồm ma trận hệ số hoán đổi R là ma trận thể hiện tốc độ biến đổi từ axit amin này sang axit amin khác và vectơ tần số Π biểu diễn quá trình thay thế axit amin trên các trình tự của bộ dữ liệu $\mathbf{D}$.

Phương pháp ước lượng mô hình thay thế bằng phương pháp cực đại khả năng sẽ xác định tập cây $\mathbf{T}$ và mô hình Q để cực đại hóa giá trị khả năng (likelihood) $L(Q, \mathbf{T}|\mathbf{D})$. Tuy nhiên, do sự không đồng nhất trong quá trình tiến hóa tại các vị trí khác nhau, bên cạnh mô hình Q, mô hình tốc độ biến đổi $\mathbf{V}$ được sử dụng để mô tả sự không đồng nhất này. Giá trị khả năng khi đó trở thành $L(Q, \mathbf{T}, \mathbf{V}|\mathbf{D})$.

Hai thành phần mô hình $(Q, \mathbf{V})$ và tập cây $\mathbf{T}$ có thể ước lượng độc lập với nhau [15] do vậy bài toán ước lượng được chia thành hai bước: xác định tập cây gần tối ưu và ước lượng mô hình thay thế [8], [27]. Quá trình được lặp lại cho đến thi thu được mô

hình đủ tốt bằng việc đánh giá giá trị khả năng của cây phân loài xây dựng bằng mô hình thu được đến thời điểm hiện tại và độ tương đồng của các mô hình ước lượng được trước và sau mỗi bước.

Thời gian thực hiện ước lượng chủ yếu tập trung ở bước xác định tập cây gần tối ưu, đặc biệt là khi các sắp hàng có số lượng trình tự lớn. Để giảm thời gian của bước xây dựng cây, luận án sử dụng phương pháp ước lượng nhanh FastMG [69] trong đó, sắp hàng ban đầu được chia thành các sắp hàng con, mỗi sắp hàng con sẽ gồm một số trình tự của sắp hàng ban đầu. Việc chia tách thực hiện như trong Thuật toán 2. 1.

Thuật toán chia tách một sắp hàng thành các sắp hàng con có đầu vào là một sắp hàng có n trình tự và giá trị k xác định số lượng trình tự tối thiểu trong một sắp hàng con. Đầu ra của thuật toán là các sắp hàng có số lượng trình tự từ k tới $2k$.

Cách chia được thực hiện theo tư tưởng của thuật toán BIONJ [18]. Ban đầu mỗi trình tự được coi là một nhóm. Tại mỗi vòng lặp, hai nhóm có khoảng cách di truyền nhỏ nhất (tính theo thuật toán BIONJ) được gộp lại với nhau nếu tổng số trình tự của hai nhóm nhỏ hơn $2k$. Trong trường hợp tổng số lượng trình tự trong hai nhóm lớn hơn $2k$, nhóm nào có nhiều hơn k trình tự sẽ được tách thành sắp hàng con riêng. Theo [69], giá trị $k \geq 8$ là phù hợp.

Quy trình ước lượng mô hình FLAVI được mô tả trong Thuật toán 2. 2.

Trong bước đầu tiên, phần mềm fastMG được sử dụng để chia các sắp hàng ban đầu thành các sắp hàng nhỏ hơn theo cấu trúc cây. Mô hình Q được ước lượng trên tập sắp hàng con mới (mỗi sắp hàng có từ k đến $2k$ trình tự).

Quá trình ước lượng mô hình là một vòng lặp để cập nhật các giá trị trong ma trận của mô hình. Ký hiệu Q là mô hình ước lượng được trong lần lặp cuối và Q' là mô hình ước lượng được trong lần lặp hiện tại, vòng lặp được thực hiện đến khi Q và Q' hội tụ (tức là hệ số tương quan giữa hai mô hình xấp xỉ bằng một). Mỗi lần lặp gồm hai bước:

- Lựa chọn mô hình thay thế, mô hình tốc độ và xây dựng cây T cho mỗi sắp hàng. Với mỗi sắp hàng, một mô hình phù hợp nhất được chọn từ một tập mô hình khởi tạo **M**. Mô hình phù hợp nhất trong trường hợp này là mô hình có điểm BIC lớn nhất trên một cây khởi tạo sử dụng thuật toán ModelFinder [70]. Việc sử dụng tập mô hình khởi tạo **M** để khoanh vùng những mô hình có khả năng phù hợp nhất với bộ dữ liệu giúp giảm thời gian xây dựng cây. Trong trường hợp không chắc chắn về các mô hình có khả năng phù hợp, ta không sử dụng tập **M**, khi đó mô hình phù hợp nhất được tìm trong tất cả các mô hình có thể có. Tiếp theo, ước lượng tham số điều khiển hình dạng α của mô hình tốc độ biến đổi và sử dụng hai mô hình này để sử dụng xây dựng cây cực đại khả năng cho sắp hàng đang xét (xem thêm mục 1.1.6 và 1.3.1 để biết thêm chi tiết về thuật toán).
- Cập nhật cho các giá trị trong ma trận thuật toán cực đại kỳ vọng [9] dựa trên tập mô hình và cây cho mỗi sắp hàng con.

2.3 Thực nghiệm

2.3.1 Dữ liệu

Chi Flavivirus có các vi rút phổ biến như West Nile, Dengue, Zika và một vài vi rút khác. Đây là các vi rút chứa một sợi ARN, cấu trúc đối xứng hình khối, có một vỏ bọc ngoài nucleocapsid. Bộ gen của vi rút dài khoảng 9500 đến 12500 nucleotit, gồm 3 gen mã hóa các protein cấu trúc (là gen E – vỏ bọc, M - màng, và C - capsid) và 7 gen mã hóa các protein không cấu trúc (NS) [72].

Để ước lượng mô hình thay thế axit amin của các vi rút này, luận án sử dụng dữ liệu trình tự protein được NCBI (https://www.ncbi.nlm.nih.gov/genomes/virusvariation/) [67] cung cấp. Dữ liệu sau khi tải về được phân tách theo từng loại protein như trên Hình 2. 1 và loại bỏ các trình tự trùng lặp. Sau khi xử lý, dữ liệu **D** thu được gồm 11392 trình tự (bao gồm 693 trình tự vi rút Zika, 2091 trình tự vi rút West Nile và 8698 trình tự vi rút Dengue), mỗi trình tự là một protein của một trong ba loại vi rút.

Algorithm Phương pháp chia sắp hàng dựa trên cấu trúc cây

input: Sắp hàng $D = \{d_1; d_2; ...; d_n\}$ và số nguyên dương k
output: Tập các sắp hàng con $\mathbf{L} = \{D_1; D_2; ...; D_m\}$; D_i có từ $k \to 2k$ trình tự.
procedure TreeBasedSplit(D, k)
    Khởi tạo $\mathbf{G} = \{G_i = d_i\}$; $\mathbf{L} = \emptyset$
    while Còn > 1 nhóm trong **G do**
        for all $i \neq j$ **do**
            Tính khoảng cách giữa G_i và G_j sử dụng thuật toán BIONJ [18]
        end for
        Tìm hai nhóm có khoảng cách nhỏ nhất, giả sử là G_1 và G_2, số trình tự trong mỗi nhóm là m_1 và m_2
        if $m_1 + m_2 \leq 2k$ **then**
            Gộp G_2 vào G_1
        else
            if $m_1 > k$ **then**
                $\mathbf{G} = \mathbf{G} \backslash G_1$
                $\mathbf{L} = \mathbf{L} \cup G_1$
            else
                $\mathbf{G} = \mathbf{G} \backslash G_2$
                $\mathbf{L} = \mathbf{L} \cup G_2$
            end if
        end if
    end while
    if Số trình tự trong nhóm còn lại > 2 **then**
        Coi nhóm còn lại là một sắp hàng con và thêm vào **L**
    else
        Nhập nhóm còn lại vào nhóm gần nhất trong **L**
    end if
end procedure

Thuật toán 2. 1 Phương pháp chia tách sắp hàng dựa trên cấu trúc cây BIONJ [69]

Algorithm Phương pháp ML ước lượng mô hình thay thế axit amin

input: Tập n sắp hàng $\mathbf{D} = \{D_1; D_2; ...; D_n\}$ và tập mô hình khởi tạo $\mathbf{M}$
output: Mô hình thay thế ước lượng được Q
procedure FLAVI(**D**,**M**)
    for each $D_i \in \mathbf{D}$ **do**
        Sử dụng FastMG, chia D_i thành các tập sắp hàng con $D_{i1}; D_{i2}; ...; D_{ik}$ dùng ma trận khoảng cách và cấu trúc cây
    end for
    Đặt $Q = \emptyset; Q' = \emptyset$
    while $(Q == \emptyset) \vee (Q \neq Q')$ **do**
        $Q = Q'; \mathbf{M} = \mathbf{M} \cup Q$
        for each D_{ij} **do**
            Lựa chọn mô hình Q' phù hợp nhất trong tập $\mathbf{M}$
            Tối ưu tham số α của mô hình tốc độ $\mathbf{V}$
            Xây dựng cây ML cho D_{ij} sử dụng mô hình Q' và $\mathbf{V}$
        end for
        Ước lượng mô hình Q' từ tập sắp hàng và tập cây tương ứng
    end while
end procedure

Thuật toán 2. 2 Ước lượng nhanh mô hình thay thế axit amin cho bộ dữ liệu **D**

Tổng quan về dữ liệu được mô tả trong Bảng 2. 1. Tập các protein **D** được chia ngẫu nhiên thành hai phần bằng nhau, một phần để ước lượng mô hình và một phần để đánh giá mô hình. Những trình tự cùng loại protein của mỗi loài trong mỗi phần ước lượng và đánh giá được sắp hàng bằng phần mềm MUSCLE [13]. Với 10 loại protein của ba loại vi rút, mỗi bộ dữ liệu huấn luyện và dữ liệu kiểm tra có 30 sắp hàng.

2.3.2 Tham số cài đặt

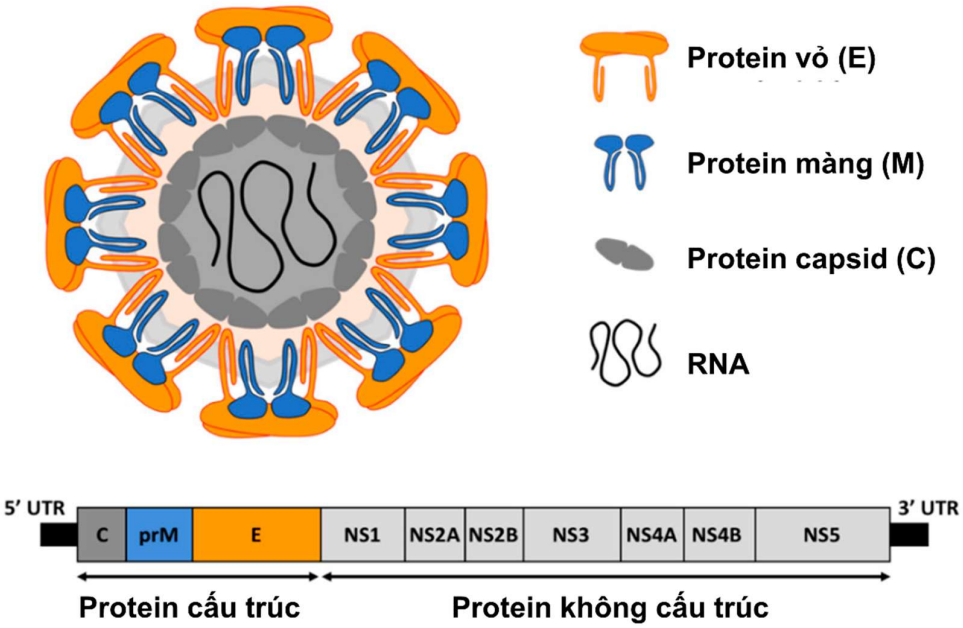
Theo bài báo gốc, ngưỡng khuyên dùng mà tác giả đưa ra là ≥ 8. Để đảm bảo kết quả thu được trong thời gian đủ nhanh mà vẫn giữ được nhiều thông tin trong mỗi gióng hàng con do dữ liệu ước lượng là virut có tốc độ phát triển nhanh, giá trị tham số k trong thuật toán ước lượng nhanh mô hình thay thế FastMG [69] được đặt bằng 25.

Nghĩa là mỗi gióng hàng con có từ 25 tới 50 trình tự, trừ gióng hàng cuối cùng khi số trình tự còn lại quá ít sẽ được gộp vào gióng hàng phù hợp nhất. Với k = 25, thuật toán thực hiện trong thời gian chấp nhận được (chưa đến 24 giờ) và có số lượng trình tự trong mỗi sắp hàng con nhiều hơn, thể hiện được nhiều thông tin hơn so với việc chia nhỏ hơn nữa.

Thuật toán ModelFinder [70] được sử dụng để chọn mô hình Q phù hợp nhất khi thực hiện xây dựng cây.

Tập mô hình ban đầu **M** gồm ba mô hình thay thế chung cho các loài JTT [10], WAG [8] và LG [27] và 3 mô hình thay thế axit amin dành riêng cho các loại vi rút HIVb, HIVw [11] và FLU [12].

Giá trị ngưỡng để xác định hai mô hình Q và Q' tương đương là 0.999, tức là thuật toán sẽ kết thúc khi hệ số tương quan giữa mô hình ước lượng ở hai vòng lặp liên tiếp lớn hơn hoặc bằng 0.999.



Hình 2. 1: Cấu trúc virion và bộ gen của vi rút Zika [99]

Bảng 2. 1 Tổng quan về dữ liệu được sử dụng. Số trình tự là số trình tự trong sắp hàng protein của vi rút tương ứng, độ dài là số axit amin có trong sắp hàng.

	Zika		**West Nile**		**Dengue**	
	Số trình tự	Độ dài	Số trình tự	Độ dài	Số trình tự	Độ dài
C	51	117	240	118	756	111
M	41	75	109	75	438	74
E	36	97	109	97	738	193
NS1	116	352	449	351	1497	354
NS2A	67	219	184	186	1142	207
NS2B	32	126	142	126	496	127
NS3	31	151	113	150	542	152
NS4A	24	145	59	143	530	144
NS4B	72	243	334	247	663	242
NS5	133	639	352	641	1896	652

2.3.3 Thực nghiệm

Các thực nghiệm được thực hiện nhằm:

- Đánh giá tính bền vững của mô hình hay độ phù hợp của dữ liệu để ước lượng mô hình nhiều tham số.
- Đánh giá sự khác biệt của mô hình mới.
- So sánh hiệu quả khi xây dựng cây phân loài.

Cụ thể:

Để đánh giá tính bền vững của mô hình, dữ liệu sau khi tiền xử lý **D** được chia ngẫu nhiên thành hai nửa **D1** và **D2**. Sau đó áp dụng quy trình ước lượng của FastMG đã giới thiệu trong 2.2 để ước lượng hai mô hình FLAVI-1 cho **D1** và FLAVI-2 cho **D2**. Tính giá trị tương quan giữa hai mô hình FLAVI-1 và FLAVI-2. Giá trị tương quan được sử dụng để đánh giá mức độ ổn định của mô hình kết quả và đưa ra kết luận về độ phù hợp của dữ liệu. Mô hình là bền vững (hay dữ liệu đủ lớn) nếu giá trị tượng quan giữa FLAVI-1 và FLAVI-2 gần gằng 1.

Trong các mô hình đã được đề xuất, sáu mô hình phổ biến được chọn để so sánh với FLAVI bao gồm mô hình thay thế chung JTT [10], LG [9] và bốn mô hình thay thế dành riêng cho các loài vi rút, cụ thể là mô hình FLU cho vi rút cúm [12], HIVb/HIVw cho vi rút HIV [11], và rtREV cho retrovirus [73]. Sự khác biệt của FLAVI với các mô hình đã có được đánh giá thông qua hệ số tương quan giữa các cặp mô hình. Mô hình chung LG và mô hình HIVb được sử dụng để so sánh giá trị tương ứng của từng tham số với mô hình FLAVI.

Các cây phân loài cực đại khả năng được xây dựng cho bộ dữ liệu kiểm tra với từng mô hình thay thế (trong thực nghiệm này, mô hình đa ma trận LG4X được sử dụng bên cạnh các mô hình trong bước trên). Hiệu quả của các mô hình được đánh giá dựa trên giá trị khả năng và độ đo AIC, bên cạnh đó thuật toán kiểm tra thống kê AU (Approximately Unbiased) [59] cũng được sử dụng để so sánh các cây được tạo ra bởi hai mô hình tốt nhất trên cùng một bộ dữ liệu.

Các thuật toán lựa chọn mô hình, tính giá trị khả năng, cây phân loài cực đại khả năng và tính độ đo AIC của cây được thực hiện bằng phần mềm IQTREE[22].

Thuật toán kiểm tra thống kê AU được thực hiện bằng phần mềm CONSEL [74].

Mã nguồn của chương trình ước lượng tham số cho mô hình thay thế axit amin có thể được tham khảo tại https://github.com/thulekm/flavi

2.4 Kết quả

2.4.1 Tính bền vững của mô hình

Với hai mô hình FLAVI-1 và FLAVI-2 được ước lượng từ hai nửa dữ liệu D1 và D2, hệ số tương quan Pearson giữa FLAVI-1 và FLAVI-2 trên hai ma trận hệ số hoán đổi là 0.994 và trên hai vectơ tần số axit amin là 0.991. Hệ số tương quan xấp xỉ 1 cho thấy lượng dữ liệu sử dụng để huấn luyện có độ lớn phù hợp, đảm bảo mô hình ước lượng được là bền vững.

Mô hình FLAVI được sử dụng là mô hình FLAVI-1.

2.4.2 Phân tích và đánh giá mô hình

Bảng 2. 2 tổng hợp hệ số tương quan của các cặp mô hình, nửa trên của bảng thể hiện hệ số tương quan của các cặp ma trận hệ số hoán đổi trong khi nửa dưới thể hiện hệ số tương quan của các vectơ tần số axit amin. Hệ số tương quan Pearson thấp giữa các cặp mô hình thể hiện sự khác nhau đáng kể giữa các mô hình.

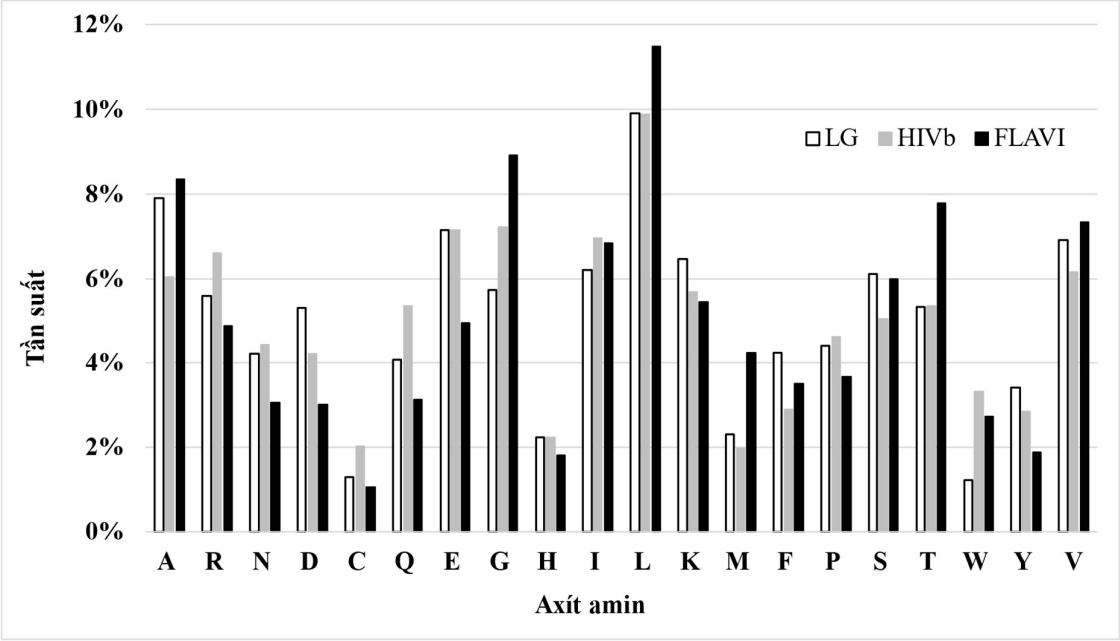
Hệ số tương quan của ma trận hệ số hoán đổi của FLAVI với các mô hình còn lại có giá trị dao động từ 0.67 (tương quan với rtREV) tới 0.92 (tương quan với FLU); trong khi hệ số tương quan của vectơ tần số thay đổi từ 0.59 (HIVw) tới 0.90 (JTT). Các giá trị này chứng tỏ FLAVI có nhiều điểm khác so với các mô hình đã có. Khi quan

Bảng 2. 2 Hệ số tương quan Pearson giữa FLAVI và các mô hình khác. Giá trị ở nửa trên là hệ số tương quan của ma trận hệ số hóa đổi, nửa dưới là tương quan của các vectơ tần số

	FLAVI	**JTT**	**LG**	**HIVw**	**HIVb**	**FLU**	**rtREV**
FLAVI	-	0.88	0.77	0.89	0.91	0.92	0.67
JTT	0.90	-	0.91	0.80	0.90	0.87	0.83
LG	0.85	0.96	-	0.65	0.80	0.81	0.95
HIVw	0.59	0.58	0.55	-	0.85	0.84	0.54
HIVb	0.86	0.88	0.89	0.66	-	0.86	0.71
FLU	0.73	0.80	0.72	0.84	0.72	-	0.75
rtREV	0.80	0.86	0.87	0.59	0.90	0.67	-

sát tương quan của ma trận hệ số hoán đổi trong mô hình FLAVI với hai mô hình chung (JTT và LG) ta thấy các giá trị này thấp hơn so với các mô hình thay thế axit amin của vi rút trừ mô hình rtREV. Xu hướng này là hợp lý vì những mô hình này được đề xuất để biểu diễn quá trình tiến hóa của các loài nói chung nên khó có thể biểu diễn tốt quá trình biến đổi diễn ra trên vi rút. Riêng mô hình rtREV được ước lượng cho retrovirus từ bộ dữ liệu khá nhỏ - chỉ có 33 trình tự, do vậy có thể có nhiều điểm khác các mô hình vi rút còn lại.

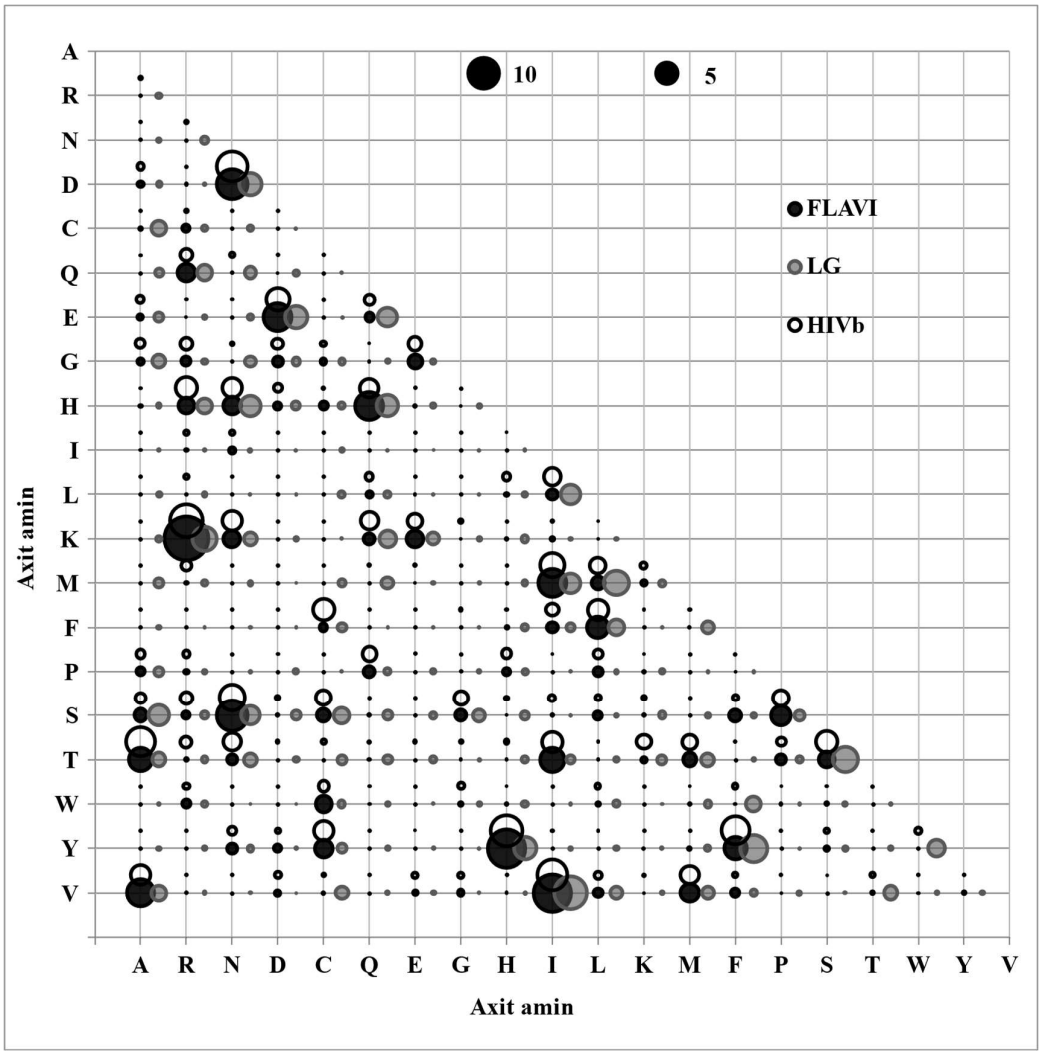
Khi so sánh từng tham số trên ba mô hình LG, HIVb và FLAVI, bên cạnh các đặc điểm tương đồng có nhiều điểm khác biệt về giá trị thành phần trên mỗi mô hình - xem Hình 2. 2, Hình 2. 3. Hình 2. 2 biểu diễn tần số của mỗi axit amin trong ba vectơ tần số của ba mô hình, trong đó một số axit amin có tần số gần bằng nhau như I (Isoleucine), L (Leucine) hay K (Lysine) trong khi một số axit amin có tần số khác nhau đáng kể như tần suất của M (Methionine) trong FLAVI là 4% xấp xỉ gấp đôi trong hai mô hình còn lại; tần suất của W (Tryptophan) trong FLAVI gấp khoảng ba lần giá trị tương ứng của LG. Hình 2. 3 mô tả hệ số hoán đổi của ba ma trận tương



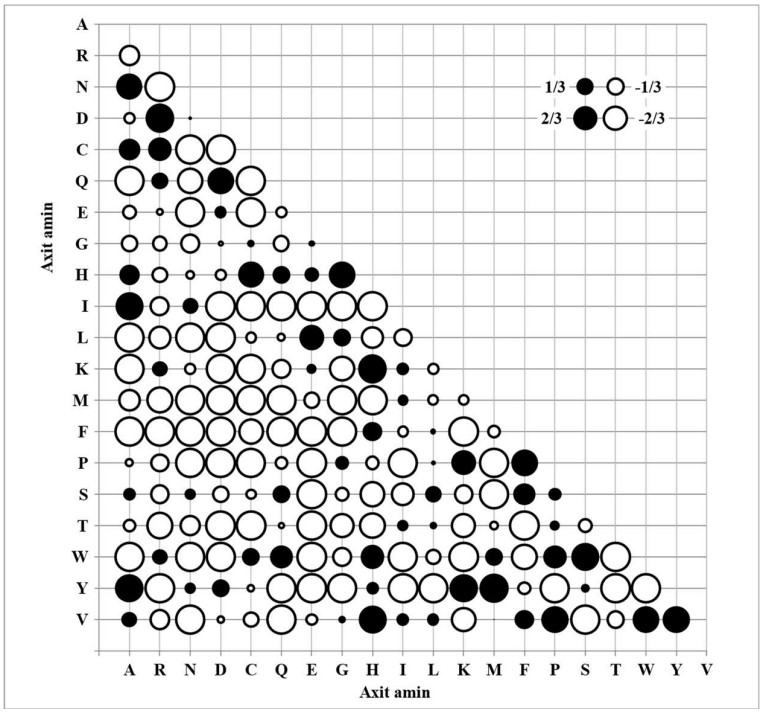
Hình 2. 2 Tần số của 20 axit amin trong ba mô hình FLAVI, LG và HIVb.

ứng trong các mô hình FLAVI, HIVb, và LG. Về cơ bản, sự hoán đổi xảy ra trên cả ba mô hình phù hợp một cách tương đối, tuân theo quy luật tiến hóa chung thể hiện ở các cụm vòng tròn thường có kích thước tương đồng về mức độ lớn/nhỏ. Tuy nhiên có một số khác biệt dễ thấy giữa các mô hình, ví dụ như các mẫu thay thế trên hàng T (Threonine), cột C (Systeine), Q (Glutamine) điển hình là tỉ lệ thay thế giữa T và I của FLAVI lớn gấp nhiều lần của LG, hay tỉ lệ của thay thế giữa T và K, N (Asparagine) của HIVb lớn hơn nhiều FLAVI.

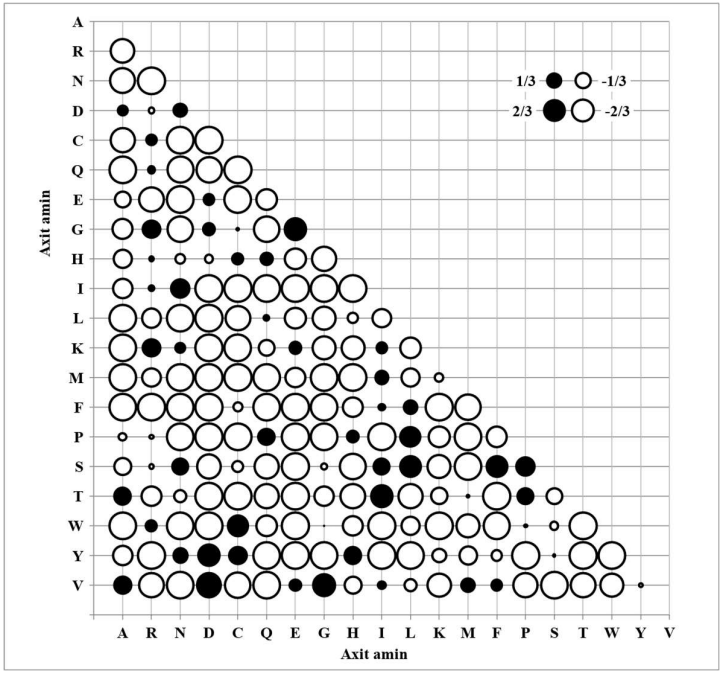
Khi phân tích sự khác biệt giữa từng cặp hệ số hoán đổi, ta thấy có sự khác biệt lớn thể hiện qua số lượng vòng tròn lớn chiếm đa số trên Hình 2. 4. Cụ thể, có 89 trong số 189 cặp hệ số của ma trận hoán đổi trong mô hình FLAVI và HIVb sai khác nhau trên 5 lần, con số này là 99 đối với mô hình LG (Bảng 2. 3). Kết quả cho thấy mô hình FLAVI khác biệt với các mô hình hiện có, đặc biệt là khác các mô hình chung, do vậy sử dụng mô hình chung để phân tích dữ liệu Flavivirus là không phù hợp.



Hình 2. 3 Ma trận hệ số hoán đổi của ba mô hình FLAVI, HIVb và LG. Vòng tròn màu đen, xám và trắng tại hàng X, cột Y thể hiện hệ số hoán đổi giữa axit amin X và Y của mô hình FLAVI, LG và HIVb



a. So sánh hệ số hoán đổi trên hai mô hình FLAVI và HIVb



b. So sánh hệ số hoán đổi trên hai mô hình FLAVI và LG

Hình 2. 4 So sánh tương quan các hệ số hoán đổi giữa FLAVI với HIVb (Hình a) và LG (Hình b). Các hình tròn hiển thị sự khác biệt tương đối giữa hệ số hoán đổi trong FLAVI với X (HIVb hoặc LG) giá trị được tính bằng (FLAVI_XY – HIVb_XY)/(FLAVI_XY + HIVb_XY) . Các hình tròn màu đen thể hiện hệ số của FLAVI lớn hơn X, màu trắng thể hiện hệ số của X lớn hơn FLAVI. Giá trị 1/3 hoặc 2/3 có nghĩa hệ số của FLAVI lớn hơn X 2 hoặc 5 lần. Giá trị -1/3 hoặc -2/3 có nghĩa hệ số của X lớn hơn FLAVI 2 hoặc 5 lần.

Bảng 2. 3 Sự khác nhau trên từng phần tử trong ma trận hoán đổi. Giá trị ở hàng "gấp đôi", cột "FLAVI>HIVb" là số hệ số trong ma trận FLAVI lớn hơn hoặc bằng hai lần hệ số ở vị trí tương ứng trong ma trận HIVb. Các giá trị khác trong bảng có ý nghĩa tương tự.

	FLAVI > HIVb	**HIVb > FLAVI**	**FLAVI > LG**	**LG > FLAVI**
Gấp đôi	31	96	24	120
Gấp năm	18	71	7	92

2.4.3 So sánh hiệu quả của FLAVI

Sau khi xây dựng cây cực đại khả năng xây dựng được cho 30 bộ dữ liệu kiểm tra với từng mô hình, giá trị khả năng và chỉ số AIC của các cây ứng với mỗi sắp hàng được so sánh để xem mô hình nào giúp xây dựng cây tốt hơn. Kết quả tổng hợp được thể

Bảng 2. 4 Kết quả so sánh giá trị khả năng của các cây phân loài được xây dựng bằng tám mô hình. Giá trị tại hàng X, cột Y là số cây được xây dựng bởi mô hình X có giá trị khả năng cao thứ Y trong số các cây xây dựng được.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
FLAVI	28	2	0	0	0	0	0	0
JTT	0	1	7	15	7	0	0	0
LG	0	0	0	0	4	7	9	10
FLU	0	0	0	2	0	7	13	8
HIVw	0	3	13	5	3	3	2	1
HIVb	2	15	7	4	2	0	0	0
LG4X	0	0	2	3	8	2	3	7
RtREV	0	9	1	1	6	11	1	1

Bảng 2. 5 Bảng so sánh giá trị log-likelihood và AIC trung bình tương ứng với các mô hình trên 30 cây cực đại khả năng trong dữ liệu kiểm tra. Lưu ý: giá trị log-likelihood cao hơn hoặc giá trị AIC nhỏ hơn tương ứng với mô hình tốt hơn.

	Log-likelihood/vị trí	**AIC / sắp hàng**	**AIC /vị trí**
FLAVI	-15.5	6649	34.8
HIVb	-15.7	6722	35.2
FLU	-15.8	6745	35.3
JTT	-15.8	6771	35.5
HIVw	-15.9	6799	35.6
LG4X	-16	6861	35.9
LG	-16.1	6889	36.1
rtREV	-16.3	6964	36.5

hiện trên Bảng 2. 4. Giá trị log-likelihood trung bình, điểm AIC trung bình tại mỗi vị trí và điểm AIC trung bình cho mỗi sắp hàng được thể hiện trên Bảng 2. 5.

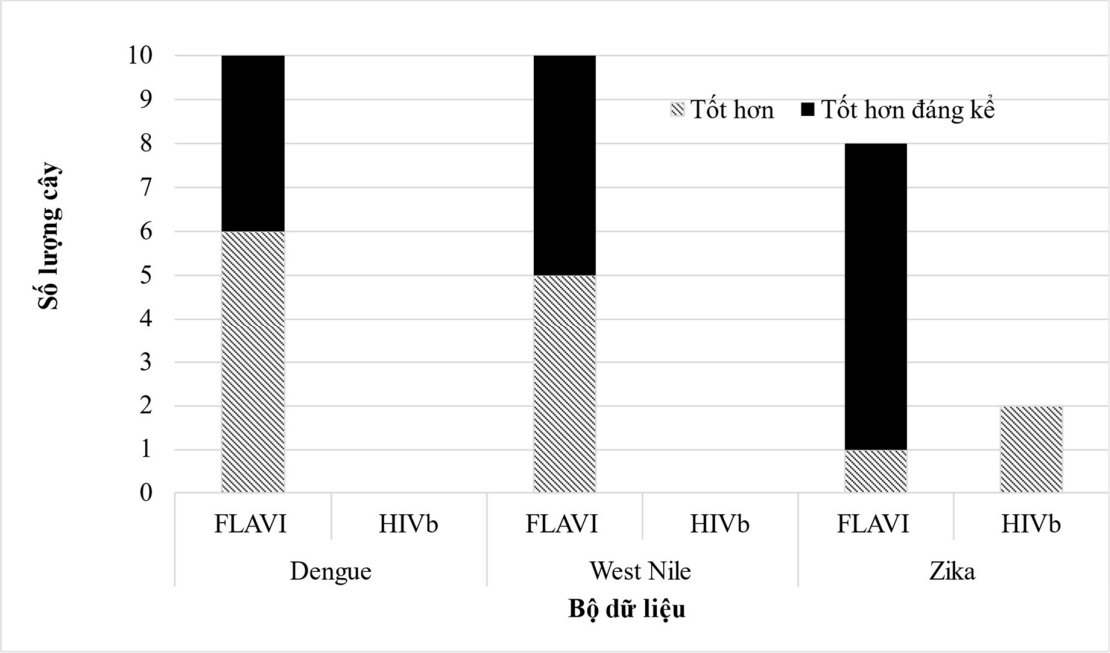
Nhìn vào Bảng 2. 4, dễ dàng nhận thấy FLAVI tốt hơn các mô hình còn lại, thể hiện ở giá trị khả năng của 28 trong 30 cây là cao nhất, chỉ có 2 cây có giá trị khả năng xếp thứ 2. Mặc dù dữ liệu của virus Dengue chiếm 76.4% trong khi virus West Nile và Zika chỉ chiếm tương ứng là 18.3% và 5.3%, mô hình FLAVI cho kết quả tốt trên các bộ dữ liệu kiểm tra của cả ba loại. Chứng tỏ các virus trong chi Flavivirus có thể có nhiều điểm tương đồng trong quá trình phát triển. Trong các mô hình còn lại HIVb là mô hình tốt nhất với hai cây cực đại khả năng tốt nhất cho hai sắp hàng protein của Zika, đồng thời có 15 cây trong số 28 cây còn lại xếp thứ 2 (chỉ sau FLAVI). Ở chiều ngược lại, mô hình rtREV cho kết quả kém nhất. Hiệu suất kém của rtREV cảnh báo rằng các mô hình thay thế axit amin nên được ước lượng từ các bộ dữ liệu có kích thước đủ lớn.

Dựa trên giá trị cụ thể trong Bảng 2. 5, khi sử dụng mô hình FLAVI, giá trị trung bình tăng lên 0.2 điểm log-likehood (1.27%) so với mô hình xếp thứ hai và tăng 0.3 điểm log-likehood (1.9%) so với mô hình thay thế chung tốt nhất.

Khi so sánh cấu trúc cây xây dựng bằng các mô hình khác nhau cho cùng một sắp hàng bằng khoảng cách Robinson-Foulds chuẩn hóa giữa mỗi cặp cây. Trung bình khoảng cách của tất cả các cặp cây đã xây dựng là 0.6 thể hiện sự khác biệt đáng kể trong cấu trúc giữa các cây. Sự khác biệt lớn này là do mối liên hệ giữa các trình tự khá gần dẫn đến việc khó khăn trong xác định rõ ràng cây nào tốt hơn nếu chỉ dựa trên giá trị khả năng; ngay cả khi một cây có giá trị khả năng lớn hơn khá nhiều so với một cây khác.

Để xác định xem liệu cấu trúc của cây tốt nhất tính theo giá trị khả năng (tức là cây có giá trị khả năng cao nhất) có thực sự là cây tốt nhất hay không, kiểm tra thống kê AU [59] được lựa chọn để thực hiện. AU được đánh giá là thuật toán kiểm tra tốt vì giải quyết được vấn đề có thể gây sai lệch tăng giá trị khả năng của các cây dẫn đến kết luận sai trong một số trường hợp như các thuật toán đề xuất trước như [57], [58]. Thuật toán thực hiện so sánh các cặp cây xây dựng bởi mô hình FLAVI và mô hình HIVb là mô hình xây dựng tập cây tốt thứ hai. Kết quả so sánh với độ tin cậy 0.95 được thể hiện trong Hình 2. 5. Trong 28 sắp hàng kiểm tra mà cây FLAVI có giá trị khả năng cao hơn cây HIVb, 16 cây là tốt hơn đáng kể, 12 trường hợp còn lại tuy cây FLAVI có giá trị khả năng cao hơn, có thể do ảnh hưởng của quá trình lấy mẫu và thống kê.

Khi phân tích cấu trúc hai cây được xây dựng bởi hai mô hình FLAVI và HIVb trên toàn bộ dữ liệu (nghĩa là gộp dữ liệu của cả ba loại vi rút thành một sắp hàng duy nhất) chúng tôi thấy rằng các trình tự thuộc cùng một loại vi rút đều được gom và tách thành nhóm riêng. Các nhóm này có khoảng cách lớn hơn, trong khi trong mỗi nhánh, độ dài các nhánh con rất ngắn. Tập các nhánh ngắn này (độ dài trung bình chỉ 0.006) là nguyên nhân của khoảng cách lớn giữa các cây.



Hình 2. 5 Kết quả so sánh cây ML của 30 sắp hàng Dengue, West Nile và Zika trong tập dữ liệu kiểm tra bằng thuật toán AU. Cây Tốt hơn là cây có giá trị khả năng cao hơn, nhưng không thực sự đáng kể theo thuật toán kiểm tra AU; cây Tốt hơn đáng kể là cây có giá trị khả năng cao hơn thực sự đáng kể theo thuật toán.

Trong tất cả các cây được xây dựng bởi tám mô hình, cây xây dựng bởi LG4X là cây có chiều dài lớn nhất (tổng độ dài của các cây là 65.1). Trong các cây còn lại, tổng độ dài cạnh của các cây FLAVI là 63.4 là cây có chiều dài lớn nhất khi sử dụng ma trận đơn. Phát hiện này cho thấy, khi sử dụng mô hình đơn ma trận, cây FLAVI thể hiện được nhiều biến đổi ẩn trong quá trình tiến hóa của chi Flavivirus hơn so với các mô hình đã có.

Cuối cùng, chúng tôi đánh giá khả năng sử dụng mô hình FLAVI cho vi rút khác trong chi Flavivirus ngoài ba vi rút Zika, West Nile và Dengue đã xuất hiện trong dữ liệu huấn luyện. Ở đây, chúng tôi thu thập 261 trình tự protein của vi rút sốt vàng da từ National Center for Biotechnology Information và xử lý giống như đã trình bày trong 2.3.1 để tạo 10 sắp hàng tương ứng với 10 loại protein của vi rút sốt vàng da. Khi so sánh cây ML xây dựng từ tám mô hình đã kể ở trên, cây FLAVI cho kết quả

tốt nhất trong 6 trường hợp (LG4X cho cây tốt nhất trong hai trường hợp, JTT và FLU mỗi loại tốt nhất cho một trường hợp).

2.5 Kết luận

West Nile, Dengue, và Zika là ba trong số các vi rút thuộc chi Flavivirus gây ra các dịch bệnh nguy hiểm trong thời gian gần đây. Mô hình thay thế axit amin FLAVI cho chi Flavivirus (Bảng 2. 6) được ước ượng dựa trên dữ liệu của ba vi rút trên. Kết quả của các thực nghiệm cho thấy dữ liệu được sử dụng để ước lượng mô hình FLAVI là đủ lớn, đảm bảo mô hình ước lượng được là ổn định. Mô hình FLAVI có một số đặc trưng khác biệt với các mô hình đã có, thể hiện quá trình thay thế khác biệt của chi Flavivirus. Sử dụng mô hình FLAVI xây dựng cây có giá trị khả năng tốt hơn so với cây sử dụng các mô hình khác. Kết quả này thu được không chỉ đối với ba vi rút được sử dụng để ước lượng mô hình mà cho cả vi rút khác trong cùng chi. Do vậy, để tiết kiệm thời gian, các nhà khoa học có thể sử dụng FLAVI như mô hình mặc định khi phân tích trình tự protein của các vi rút trong chi Flavivirus. Tuy nhiên, các kết quả thực nghiệm cũng chỉ ra rằng trong một số trường hợp FLAVI không phải là mô hình cho khả năng cao nhất, do vậy, chúng ta có thể sử dụng thuật toán lựa chọn mô hình phù hợp nhất như ModelFinder [70] để chọn trong một tập giới hạn các mô hình có khả năng cao.

Bảng 2. 6 Mô hình thay thế axit amin FLAVI cho chi Flavivirus

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A																				
R	0.077																			
N	0.078	0.000																		
D	0.551	0.089	8.801																	
C	0.115	0.573	0.000	0.000																
Q	0.090	3.857	0.093	0.184	0.000															
E	0.561	0.021	0.000	7.603	0.000	1.058														
G	0.478	1.281	0.148	1.360	0.221	0.000	2.331													
H	0.169	2.704	3.326	0.543	0.241	8.959	0.025	0.033												
I	0.000	0.151	0.538	0.000	0.000	0.000	0.000	0.000	0.037											
L	0.003	0.114	0.000	0.000	0.085	0.790	0.024	0.038	0.376	1.571										
K	0.000	19.413	2.764	0.000	0.000	1.546	2.993	0.028	0.000	0.253	0.039									
M	0.000	0.151	0.000	0.000	0.000	0.000	0.031	0.009	0.000	8.538	2.895	0.340								
F	0.000	0.000	0.000	0.000	0.773	0.000	0.000	0.000	0.093	1.369	6.297	0.000	0.000							
P	1.256	0.292	0.000	0.000	0.000	1.571	0.000	0.030	0.723	0.000	1.183	0.098	0.000	0.035						
S	2.104	0.947	9.494	0.128	1.516	0.141	0.000	1.736	0.080	0.212	1.009	0.131	0.074	1.822	4.417					
T	6.315	0.200	1.330	0.000	0.000	0.057	0.000	0.045	0.144	6.768	0.033	0.567	2.228	0.000	1.543	3.291				
W	0.000	0.791	0.000	0.000	2.306	0.000	0.000	0.324	0.207	0.000	0.176	0.000	0.173	0.148	0.074	0.154	0.000			
Y	0.044	0.000	1.248	0.772	4.086	0.000	0.030	0.000	16.469	0.000	0.000	0.072	0.229	5.908	0.000	0.431	0.000	0.000		
V	8.031	0.000	0.000	0.440	0.150	0.000	0.346	0.583	0.067	16.913	1.074	0.007	3.856	1.177	0.037	0.053	0.134	0.000	0.234	
П	0.078	0.054	0.034	0.035	0.014	0.030	0.055	0.086	0.018	0.063	0.104	0.060	0.040	0.034	0.037	0.061	0.076	0.030	0.020	0.071

Chương 3: Phương pháp phân hoạch sắp hàng sử dụng mô hình tiến hóa

Chương 3 trình bày thuật toán mPartition, là thuật toán được đề xuất để biểu diễn sự không đồng nhất trong quá trình biến đổi tại các vị trí khác nhau trong sắp hàng thông qua việc phân hoạch sắp hàng ban đầu thành các sắp hàng con, mỗi sắp hàng con chứa các vị trí có cùng quá trình tiến hóa. Thuật toán sử dụng mô hình tiến hóa phù hợp nhất tại mỗi vị trí trong thủ tục đánh giá sự tương đồng của quá trình tiến hóa tại các vị trí khác nhau. Nội dung chương tổng hợp từ các công trình [CT2], [CT3] và [CT5].

3.1 Giới thiệu

Như đã đề cập trong chương 1, các thuật toán phân hoạch sắp hàng có đầu vào là một sắp hàng và đầu ra là một lược đồ phân vùng mà trong đó các vị trí trên sắp hàng có quá trình thay thế giống nhau thuộc cùng một phân vùng. Các thuật toán này thường sử dụng tốc độ tiến hóa là đặc trưng để phân hoạch, điển hình trong số đó là thuật toán *k*-means lặp và thuật toán RatePartition (xem thêm phần 1.3 để biết thêm chi tiết về thuật toán). Tốc độ tiến hóa nói chung thể hiện tốc độ mà các loài sinh vật thay đổi và thích nghi với môi trường trong quá trình tiến hóa. Với phạm vi dữ liệu trong một sắp hàng, tốc độ tiến hóa được tính cho từng vị trí trên sắp hàng và có giá trị thể hiện tốc nhanh/chậm tương đối giữa các vị trí. Trong số các loại tốc độ tiến hóa, TIGER là cách tính thường được sử dụng do giá trị tốc độ được tính hoàn toàn bằng dữ liệu, không bị ảnh hưởng bởi cây khởi tạo cho quá trình tính toán (xem thêm phần 1.3 để biết thêm chi tiết về thuật toán).

So với thuật toán *k*-means lặp, thuật toán RatePartition đã giải quyết được vấn đề tạo phân vùng bất biến trong lược đồ phân vùng. Tuy nhiên độ chính xác của RatePartition bị ảnh hưởng bởi hệ số phân chia *d*. Tham số này tác động tới số lượng phân vùng trong lược đồ phân vùng kết quả - khi *d* tăng, số lượng phân vùng tăng

lên; khi d giảm, số lượng phân vùng cũng giảm. Do đó, việc chọn giá trị d ảnh hưởng đáng kể đến độ chính xác của thuật toán RatePartition.

Bên cạnh đó, mặc dù tốc độ tiến hóa của các vị trí khác nhau trên cùng một sắp hàng cung cấp các thông tin hữu ích cho quá trình phân hoạch các vị trí; việc chỉ dùng tốc độ tiến hóa để đại diện cho toàn bộ quá trình thay thế của các trạng thái (nucleotit hoặc axit amin) là không phù hợp vì cách thức biến đổi tại các vị trí cũng có thể là tiêu chí cần xét đến khi đánh giá độ tương đồng giữa các vị trí [36]. Từ đó, ý tưởng của thuật toán mPartition là sử dụng cả hai loại thông tin, bao gồm tốc độ tiến hóa được ước lượng tại mỗi vị trí trên sắp hàng và mô hình thay thế giữa các nucleotit hoặc các axit amin để phân hoạch tập vị trí thành các tập con sao cho các vị trí trong cùng một tập con có tốc độ tiến hóa và mô hình thay thế tương tự nhau.

Mục tiêu của lược đồ phân vùng là gom các vị trí có quá trình tiến hóa tương đồng thành nhóm, từ đó tăng giá trị khả năng trong quá trình tính toán hay nói cách khác là tăng tính chính xác của cây phân loài xây dựng được. Do vậy việc so sánh các lược đồ phân vùng thường được thực hiện bằng cách so sánh độ đo lý thuyết thông tin trên cây cực đại khả năng tương ứng. Cụ thể là so sánh độ đo AICc (corrected Akaike information criterion[61]) hay BIC (Bayesian information criterion [62], [75]) vì số lượng tham số của các lược đồ khác nhau (số nhóm/phân vùng khác nhau).

3.2 Phương pháp

Đầu vào của bài toán là một sắp hàng đa trình tự tương đồng $D = \{d_1, \dots, d_l\}$ độ dài l, với d_i là một cột của ma trận dữ liệu D.

Đầu ra là một lược đồ phân vùng $\mathbf{PS} = \{P_1, \dots, P_k\}$ là một phân hoạch của tập các vị trí từ 1 đến l.

Thuật toán mPartition thực hiện tìm lược đồ phân vùng sao cho cây cực đại khả năng xây dựng cho sắp hàng D sử dụng lược đồ phân vùng tìm được có điểm BIC nhỏ nhất. mPartition bắt đầu bằng lược đồ khởi tạo chỉ có một phân vùng, chứa tất cả các vị trí của sắp hàng. Với lược đồ hiện tại, thuật toán chia nhỏ các phân vùng trong đó thành

các phân vùng nhỏ hơn để thu được lược đồ mới tốt hơn (giảm điểm BIC của cây cực đại khả năng). Thuật toán kết thúc khi không còn phân vùng nào có thể chia nhỏ.

Để chia nhỏ một phân vùng, trước hết mPartition chia các vị trí trong phân vùng đang xét thành ba tập con tương ứng với ba nhóm theo tốc độ tiến hóa TIGER: nhóm tiến hóa chậm gồm các vị trí có giá trị tốc độ lớn, tương tự là nhóm tiến hóa nhanh có giá trị tốc độ nhỏ, còn lại là nhóm tiến hóa trung bình. Sau đó tìm mô hình phù hợp nhất cho các nhóm và điều chỉnh lại tập vị trí trong mỗi phân vùng theo mô hình phù hợp nhất và hàm ánh xạ giá trị khả năng của các cặp vị trí – mô hình tiến hóa [76].

Thuật toán mPartition gồm bốn bước chính, được thực hiện như trong Thuật toán 3.1

Chi tiết các bước như sau:

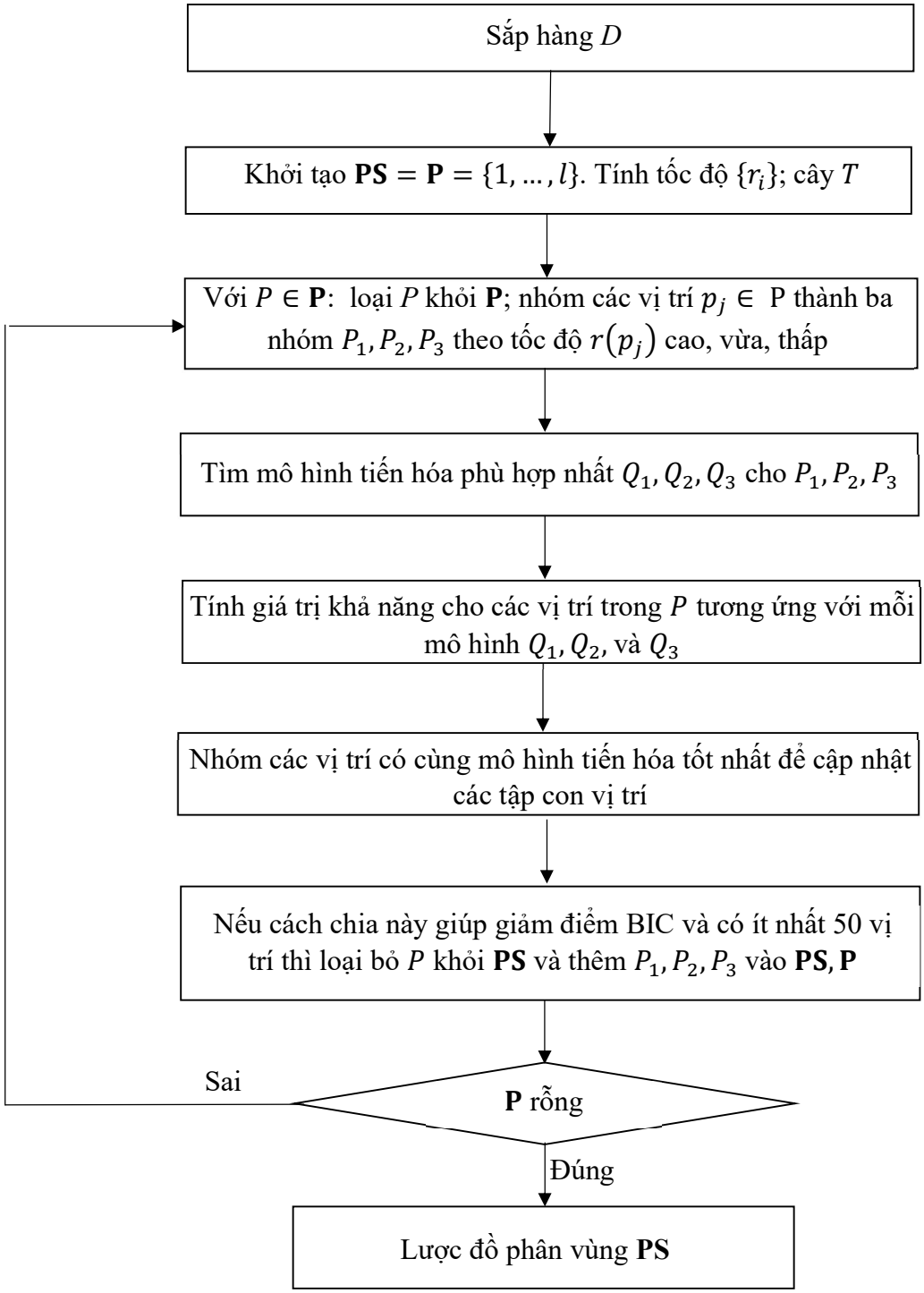
1. **Khởi tạo** (dòng 4 - 6): Nếu một phân vùng trong **PS** chưa thực hiện bước kiểm tra để chia nhỏ thì phân vùng được gắn nhãn là "đang phân vùng" (tức là đang trong quá trình chia nhỏ thành các phân vùng con). Các phân vùng ở trạng thái đang phân vùng được lưu trong danh sách **P**.

Đặt $\mathbf{PS} = \mathbf{P} = \{1, \dots, l\}$ – tức là lược đồ chỉ gồm duy nhất một phân vùng chứa tất cả vị trí của D. Tiếp theo, tính tốc độ tiến hóa cho tất cả các vị trí của sắp hàng D bằng thuật toán TIGER (xem mục 1.4 để biết thêm chi tiết về cách thực hiện thuật toán TIGER); và xây dựng cây cực đại khả năng T cho D.

2. **Phân vùng bằng tốc độ biến đổi tại các vị trí** (dòng 10)

   Với P là một tập vị trí của **P**. Gọi $r(p_j)$ là giá trị tốc độ của mỗi vị trí p_j trong P và r_{max} và r_{min} là tốc độ tiến hóa cao nhất và thấp nhất của các vị trí trong P. Lưu ý rằng, tốc độ TIGER biến thiên trong khoảng từ 0 đến 1; các vị trí biến đổi nhanh nhất có giá trị tốc độ nhỏ nhất và ngược lại.

   Các vị trí trong P được chia thành ba cụm khác nhau tương ứng với các vị trí có tốc độ cao (P_1), cụm tốc độ vừa (P_2), và cụm tốc độ thấp (P_3). Vị trí p_j được phân vào nhóm $g(p_j)$ theo công thức như sau:



Hình 3. 1 Quy trình phân hoạch sắp hàng D bằng phương pháp mPartition

Algorithm Thuật toán mPartition phân hoạch sắp hàng tự động

input: Sắp hàng D
output: Lược đồ phân vùng $\mathbf{PS} = \{S_1; ...; S_k\}$
procedure mPartition(A)
    $\mathbf{S} = \{D\}; \mathbf{P} = \{D\}$ // Khởi tạo
    Tính tốc độ r_i cho vị trí i
    Xây dựng cây ML T cho D
    while $\mathbf{P} \neq \emptyset$ **do** // Lặp đến khi không còn phân vùng nào cần chia
        for each $P \in \mathbf{P}$ **do**
            $\mathbf{P} = \mathbf{P} \setminus P$
            Chia P thành 3 nhóm $P_1; P_2; P_3$
            Xác định mô hình $Q_1; Q_2; Q_3$ phù hợp nhất cho $P_1; P_2; P_3$
            Tính giá trị khả năng cho các vị trí trong P tương ứng với $Q_1; Q_2; Q_3$
            Cập nhật $P_1; P_2; P_3$ là tập các vị trí có cùng mô hình tiến hóa tốt nhất
            $S' := S \setminus P \cup \{P_1; P_2; P_3\}$
            Tính BIC của lược đồ S'
            if $BIC(S') < BIC(S)$ **then**
                $S = S'$
                $P = P \cup \{P_1; P_2; P_3\}$
            end if
        end for
    end while
    Nhập các tập con P_i chứa toàn vị trí bất biến nếu có.
end procedure

Thuật toán 3.1 Thuật toán phân hoạch sắp hàng mPartition.

$$g(p_j) = \begin{cases} P_1: r(p_j) < r_{min} + k \\ P_2: r_{min} + k \leq r(p_j) < r_{min} + 2 \times k \\ P_3: r_{min} + 2 \times k \leq r(p_j) \end{cases} \quad (1.18)$$

Với $k = \frac{r_{max} - r_{min}}{3}$. Tức là khoảng từ r_{max} tới r_{min} được chia thành ba khoảng đều nhau tương ứng với ba nhóm.

3. **Điều chỉnh lại các tập vị trí dựa trên mô hình tiến hóa** (dòng 11-19)

Với mỗi tập vị trí P_i $(i = 1, 2, 3)$, cặp mô hình tiến hóa và mô hình tốc độ biến đổi (Q_i, V_i) phù hợp nhất được lựa chọn bằng thuật toán ModelFinder [70]. Sau khi xác định các cặp mô hình tiến hóa phù hợp nhất, giá trị khả năng tại mỗi vị trí p_j trong P đối với ba cặp mô hình lần lượt là: $L(p_j|T, Q_1, V_1)$, $L(p_j|T, Q_2, V_2)$ và $L(p_j|T, Q_3, V_3)$.

Các vị trí trong phân vùng P được sắp xếp lại dựa trên các giá trị khả năng vừa tính. Cụ thể, tập vị trí có biến đổi trong phân vùng P được chia thành ba nhóm P_1, P_2, P_3 , trong đó nhóm P_x chứa những vị trí có giá trị khả năng cao nhất khi sử dụng mô hình tiến hóa Q_x. Với tập vị trí bất biến, mỗi vị trí được gán cho một trong ba nhóm P_x với xác suất p_x tỉ lệ với giá trị khả năng của mô hình tương ứng với nhóm:

$$p_x = L(p_j|T, Q_x, V_x) / \sum_{v=1...3} L(p_j|T, Q_v, V_v) \quad (1.19)$$

Dễ thấy rằng việc sử dụng xác suất trên các vị trí bất biến giúp phân tán các vị trí bất biến trong phân vùng ban đầu vào ba phân vùng con mới được tạo. Trong một số trường hợp, khi một trong các tập con P_x quá nhỏ (dưới 50 vị trí – ngưỡng được chọn trong bài báo [77]), các vị trí trong tập sẽ được gán lại vào hai tập còn lại và loại bỏ tập con P_x khỏi lược đồ phân vùng.

Sau khi thu được các tập con $P_1, P_2,$ và P_3 mới, giá trị BIC được tính lại cho lược đồ mới. Nếu giá trị BIC của cây dùng lược đồ sau khi chia tốt hơn so với trước

khi chia thì thêm $P_1, P_2,$ và P_3 vào danh sách "đang phân vùng" **P** và thay phân vùng P trong **PS** bằng ba phân vùng mới là $P_1, P_2,$ và P_3. Trường hợp lược đồ mới không tốt hơn so với trước khi chia thì ta chỉ loại bỏ P khỏi **P** và giữ nguyên các phân vùng của lược đồ **PS**.

4. **Hiệu chỉnh phân hoạch** (dòng 22)

   Trong khi vẫn còn các tập con ở trạng thái "đang phân vùng" trong **P**, quay lại bước 2 để thực hiện chia nhỏ các tập này. Ngược lại, nếu không còn tập con nào ở trạng thái "đang phân vùng" có nghĩa là tất cả các tập hợp vị trí không thể thay đổi để cải thiện giá trị BIC, vòng lặp kết thúc. Lúc này, trong lược đồ phân vùng **PS** có thể có một số phân vùng chỉ chứa các vị trí bất biến. Các phân vùng này được gộp thành một phân vùng duy nhất chứa các vị trí bất biến. Thuật toán kết thúc và trả về kết quả là lược đồ phân vùng **PS**.

Thuật toán có thể chia làm hai phần rời nhau: tính tốc độ tiến hóa, xác định cây cực đại khả năng cho sắp hàng và thực hiện phân chia tập vị trí. Thời gian tính tốc độ tiến hóa là giống nhau cho các *k*-means lặp, RatePartition và mPartition và việc chọn thuật toán tính tốc độ đảm bảo tính đúng đắn và tốc độ tính toán nhanh là cần thiết. Sau khi đã có tốc độ tiến hóa, RatePartition có độ phức tạp kà $O(l)$, do thuật toán chỉ duyệt qua mỗi vị trí một lần để kiểm tra tốc độ tại vị trí đó và xác định vào khoảng tốc độ phù hợp. Trong khi đó mPartition và *k*-means lặp cần thực hiện xây dựng một cây cực đại khả năng cho sắp hàng D. Bài toán tối ưu cây phân loài là một bài toán NP-khó [78], các thuật toán tìm kiếm gần đúng thường được sử dụng để giảm bớt thời gian [21], [22].

Trong tác vụ phân chia tập vị trí, hai thuật toán sử dụng đệ quy, trong trường hợp xấu nhất, mPartition phải chạy $\frac{l-inv}{100}$ mức (inv là số vị trí bất biến) nếu mỗi lần chia tạo ra hai phân vùng có độ dài tối thiểu (độ dài 50 vị trí có biến đổi) và một phân vùng chứa toàn bộ các vị trí còn lại, *k*-means lặp có thể chạy tới l vòng lặp do không không chế về độ dài tối thiểu của mỗi phân vùng. Với mỗi phân vùng con, thuật toán *k*-means lặp chạy thuật toán *k*-means trên tập tốc độ ($O(l^2)$) của các vị trí tạo phân vùng

mới và tính AICc của phân vùng mới để so sánh với phân vùng cũ trong khi mPartition thực hiện chia nhóm theo tốc độ ($O(l)$), tìm mô hình tiến hóa phù hợp và tính giá trị khả năng với từng mô hình tìm được, tạo phân vùng mới và tính giá trị AICc để so sánh giống với thuật toán *k*-means lặp. Việc chọn mô hình tiến hóa phù hợp là một bài toán NP-khó và tốn nhiều thời gian do vậy bước phân chia tập vị trí của thuật toán mParition có độ phức tạp cao hơn so với thuật toán *k*-means lặp.

Tuy nhiên, mPartition thực hiện phân hoạch tập vị trí dựa trên hai thông tin, bao gồm tốc độ tiến hóa tương đối và mô hình tiến hóa tại mỗi vị trí, do vậy xét trên phương diện sinh học, kết quả đưa ra có khả năng phù hợp với quá trình tiến hóa tại các vị trí hơn so với các thuật toán chỉ sử dụng tốc độ là *k*-means lặp hay RatePartition. Hệ quả là mô hình lược đồ phân vùng thu được giúp xác định cấu trúc cây có khả năng cao hơn. Bên cạnh đó, việc sử dụng hàm phân phối mật độ xác suất để tính xác suất một vị trí bất biết thuộc một trong các phân vùng của lược đồ tránh được tình trạng đưa tất cả các vị trí vào cùng một phân vùng, điều này là không hợp lý vì, như đã đề cập ở mục 1.1.2, nucleotit/axit amin quan sát được có thể giống nhau nhưng thực tế đã xảy ra các quá trình biến đổi phức tạp.

3.3 Thực nghiệm

Các thực nghiệm khác nhau được thực hiện để so sánh lược đồ phân vùng tạo bởi thuật toán mPartition và thuật toán RatePartition trên hai loại dữ liệu mô phỏng và dữ liệu thực. Trong đó các thực nghiệm trên dữ liệu mô phỏng so sánh khả năng xây dựng lại cây gốc (cây dùng để tạo dữ liệu mô phỏng) thông qua khoảng cách nRF tới cây gốc; với dữ liệu thực, các thực nghiệm so sánh các lược đồ phân vùng bằng chỉ số AICc và BIC (không có cây gốc). Chi tiết các thực nghiệm được trình bày trên từng bộ dữ liệu.

3.3.1 Dữ liệu

Các thực nghiệm được thực hiện trên ba bộ dữ liệu khác nhau bao gồm dữ liệu DNA mô phỏng, dữ liệu DNA thực và dữ liệu protein thực. Trong đó dữ liệu DNA được

lấy từ nghiên cứu về phương pháp RatePartition; dữ liệu protein được trích từ dữ liệu của một số công trình đã được công bố [79]–[83]. Các thực nghiệm đều được thực hiện bằng phần mềm IQTREE [22]

a. Dữ liệu mô phỏng DNA

Dữ liệu mô phỏng DNA (có tại: *10.5281/zenodo.1251684*) được mô phỏng trên một cây 17 lá. Hai cấu trúc cây được mô phỏng là cây đối xứng và bất đối xứng với các thiết lập độ dài nhánh khác nhau. Mô hình tiến hóa được sử dụng là F81 và GTR cùng với mô hình tốc độ biến đổi. Với mỗi cấu trúc cây, các tác giả sử dụng 14 bộ tham số khác nhau (bao gồm các tham số độ dài nhánh và mô hình tiến hóa) và thực hiện mô phỏng 10 lần để thu được 280 bộ sắp hàng. Mỗi sắp hàng có độ dài 4000, gồm bốn phân vùng có độ dài bằng nhau, mỗi phân vùng có 1000 nucleotit được mô phỏng theo cùng một mô hình thay thế (F81 hoặc GTR), khác vectơ tần số và tỷ lệ thay thế nucleotit. Ngoài ra, để kiểm tra kết quả phân vùng trên dữ liệu không đầy đủ, các tác giả đã thực hiện loại bỏ một phân vùng trên bốn cặp tham số để tạo thêm các bộ sắp hàng thiếu dữ liệu (chi tiết về các bộ dữ liệu có trong bài báo RatePartition [32]).

Bảng 3. 1 Tám bộ dữ liệu DNA thực dùng để so sánh các phương pháp phân hoạch sắp hàng. Độ dài là số nucleotit trong sắp hàng tương ứng

Dữ liệu	**#Số trình tự**	**#Số gen**	**Độ dài**
Arctiina [84]	113	8	5809
Calisto [85]	90	6	5297
Choreutidae [86]	41	8	6293
Coenonymphina [87]	69	5	4435
Geometridae [88]	164	8	5998
Morpho [89]	31	8	6372
Noctuidae [90]	78	8	6365
Pieridae [91]	110	8	6247

b. Dữ liệu DNA thực

Bộ dữ liệu DNA thực bao gồm tám sắp hàng DNA, mỗi sắp hàng bao gồm một gen ty thể (COI) và bốn đến bảy gen nhân thường thấy trong phát sinh loài của bộ cánh vẩy (lepidopteran). Các sắp hàng có độ dài từ 4435 đến 6372 và số lượng trình tự trong mỗi sắp hàng từ 31 đến 164. Sắp hàng có kích thước từ nhỏ đến lớn tương ứng với các cỡ quần thể khác nhau. Các sắp hàng trong bộ dữ liệu bao gồm cả dữ liệu bị thiếu. Tỷ lệ thiếu dữ liệu của các sắp hàng dao động trong khoảng 9,6% đến 39,1%. Để đảm bảo không gây sai lệch khi xây dựng cây cực đại khả năng, dữ liệu được tiền xử lý để loại bỏ những trình tự có thiếu nhiều hơn 80% độ dài, trừ trường hợp tập dữ liệu có không quá 1% vị trí như vậy [32]. Thông tin về các sắp hàng DNA thực có trong Bảng 3. 1.

Dữ liệu protein thực được trích từ các nghiên cứu đã được công bố về nhện và nhện biển [79], động vật có vú [83], thực vật có hạt [82] và động vật có xương hàm [80], [81]. Trong các bộ dữ liệu gốc đã được công bố, số lượng trình tự của sắp hàng dao động từ 38 đến 100, mỗi trình tự có hàng nghìn gen và có độ dài lên tới với hàng triệu vị trí. Khối lượng tính toán là rất lớn nếu kiểm tra tất cả các lược đồ phân vùng trên các sắp hàng đầy đủ. Do vậy các sắp hàng nhỏ hơn được ghép nối từ 10, 20, 30 và 40 gen lựa chọn ngẫu nhiên từ mỗi sắp hàng ban đầu để dễ dàng thực hiện các thuật toán. Tổng cộng, có 20 sắp hàng protein thực được sử dụng để thử nghiệm hai thuật toán mPartition và RatePartition. Tỉ lệ thiếu dữ liệu trong các sắp hàng con dao động từ 1% tới 32%; và tỉ lệ vị trí bất biến từ 20% tới 45%. Lưu ý rằng việc lựa chọn gen của mỗi sắp hàng đều là ngẫu nhiên, do vậy sắp hàng lớn hơn không bao gồm dữ liệu trong sắp hàng nhỏ hơn (ví dụ sắp hàng 40 gen không bao gồm sắp hàng 10, 20 hoặc 30 gen). Thông tin về các bộ dữ liệu có trong Bảng 3. 2.

c. Dữ liệu protein thực

Bảng 3. 2 Sắp hàng protein thực sử dụng để so sánh các lược đồ phân vùng

Dữ liệu	**Nhánh**	**#Số trình tự**	**#Số gen**	**Độ dài**
Ballesteros10 [79]	Nhện biển	53	10	4046
Ballesteros20 [79]	Nhện biển	53	20	8575
Ballesteros30 [79]	Nhện biển	53	30	17045
Ballesteros40 [79]	Nhện biển	53	40	21998
Chen10 [79]	Động vật có xương hàm	58	10	3967
Chen20 [79]	Động vật có xương hàm	58	20	6403
Chen30 [79]	Động vật có xương hàm	58	30	13267
Chen40 [79]	Động vật có xương hàm	58	40	15278
Irissari10 [80]	Động vật có xương hàm	100	10	6027
Irissari20 [80]	Động vật có xương hàm	100	20	8509
Irissari30 [80]	Động vật có xương hàm	100	30	13183
Irissari40 [80]	Động vật có xương hàm	100	40	15698
Ran10 [82]	Thực vật có hạt	38	10	3062
Ran20 [82]	Thực vật có hạt	38	20	6897
Ran30 [82]	Thực vật có hạt	38	30	10443
Ran40 [82]	Thực vật có hạt	38	40	14749
Wu10 [83]	Động vật có vú	90	10	5148
Wu20 [83]	Động vật có vú	90	20	12225
Wu30 [83]	Động vật có vú	90	30	18088
Wu40 [83]	Động vật có vú	90	40	24423

3.3.2 Thực nghiệm

a. Dữ liệu mô phỏng DNA

Với các sắp hàng DNA mô phỏng, chúng tôi so sánh các phương pháp phân vùng bao gồm: (1) lược đồ phân vùng gốc (bốn phân vùng bằng nhau, là lược đồ phân vùng dùng để mô phỏng dữ liệu), (2) lược đồ kết quả của thuật toán RatePartitions sử dụng tốc độ tiến hóa TIGER với hệ số phân chia $d = 2, 3, 4,$ và 5 và (3) lược đồ phân vùng do phương pháp mPartition tạo ra. Đầu tiên chúng tôi xây dựng cây ML cho mỗi cặp sắp hàng và các lược đồ phân vùng tương ứng. Sau đó tính khoảng cách Robinson-Fould của các cây được xây dựng với cây gốc (là cây được sử dụng để mô phỏng dữ liệu [14]).

b. Dữ liệu DNA thực

Theo kết quả do nhóm tác giả Rota và các cộng sự đã thực hiện, RatePartition cho kết quả tốt hơn so với các phương pháp phân vùng dựa trên đặc trưng sinh học như ranh giới gen và/hoặc vị trí codon trên các sắp hàng DNA thực [32]. Cũng theo các tác giả, hệ số phân chia $d = 4$ và $d = 5$ là thiết lập cho kết quả tốt nhất của thuật toán RatePartition trên dữ liệu DNA thực. Do đó, các so sánh được thực hiện trên bốn lược đồ phân vùng: ba lược đồ tạo bởi mPartition, RatePartition với $d = 4$ và $d = 5$ (kí hiệu tương ứng là RP4 và RP5) và khi không phân vùng (kí hiệu là NP – no-partition).

Các thực nghiệm bao gồm:

- So sánh điểm AICc [61] và điểm BIC [62] của các cây cực đại khả năng tương ứng với mỗi lược đồ để xác định độ tốt của lược đồ
- So sánh khoảng cách nRF giữa các cây để xác định xem các lược đồ có ảnh hưởng tới cấu trúc cây hay không
- Đánh giá số lượng phân vùng trong các lược đồ kết quả sự phân phối của các vị trí bất biến trong lược đồ phân vùng thu được từ phương pháp mPartition.

c. Dữ liệu protein thực

Đối với các sắp hàng protein thực, thuật toán mPartition chọn mô hình phù hợp nhất cho mỗi tập hợp con từ ba mô hình thay thế axit amin chung phổ biến là WAG [8],

LG [9], và JTT [10]. Vì thuật toán RatePartition chưa được nghiên cứu trên dữ liệu protein thực, các hệ số phân chia d khác nhau được sử dụng trong tính toán thử nghiệm bao gồm $d = 2$ $(RP2), d = 3$ $(RP3), d = 4$ $(RP4)$ và $d = 5$ ($RP5$).

Các thực nghiệm trên sắp hàng protein được thực hiện tương tự như đối với dữ liệu DNA thực - bao gồm so sánh các lược đồ, đánh giá cấu trúc cây, và đánh giá phân bố của các vị trí bất biến.

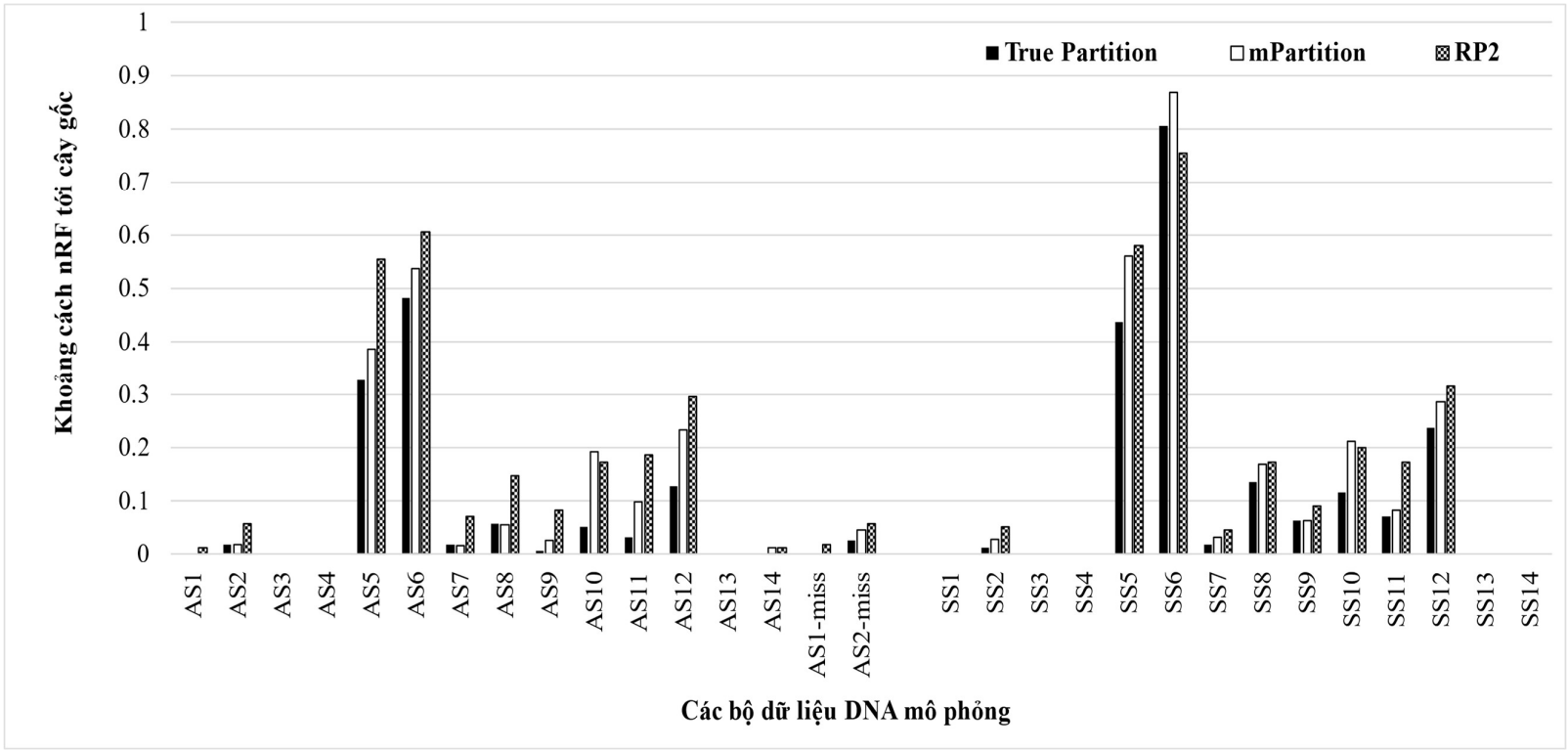
3.4 Kết quả

3.4.1 Dữ liệu DNA mô phỏng

Trung bình khoảng cách nRF giữa cây thật và các cây được xây dựng từ bốn loại lược đồ phân vùng: lược đồ gốc (TruePartition), lược đồ phân vùng tạo bởi RatePartition và lược đồ phân vùng tạo bởi mPartition được tổng kết trong Bảng 3. 3. Cụ thể, khoảng cách nRF giữa các cây thật với các cây được xây dựng bằng lược đồ phân vùng TruePartition, RatePartition và mPartition lần lượt là 0.095, 0.141 và 0.115. Lưu ý rằng khoảng cách nRF đến cây gốc nhỏ hơn tức là cây đó có ít phân nhánh không thuộc cây gốc hơn và là cây tốt hơn, tương ứng với lược đồ phân vùng tốt hơn. Dựa trên kết quả thu được, có thể kết luận lược đồ phân vùng của mPartition giúp xây dựng cây tốt hơn so với lược đồ phân vùng của RatePartition. Đối với RatePartition, kết quả đồng đều khi dùng hệ số phân chia d khác nhau trên bộ dữ liệu mô phỏng (nRF xấp xỉ 0,14 cho tất cả RP2, RP3, RP4 và RP5).

Bảng 3. 3 Trung bình khoảng cách nRF giữa cây gốc với cây ML thu được khi sử dụng các lược đồ phân vùng khác nhau trên dữ liệu mô phỏng

	TruePartition	mPartition	RP2	RP3	RP4	RP5
Khoảng cách nRF trung bình	0.095	0.115	0.141	0.140	0.141	0.142
Trung bình số tập con trong lược đồ	4	9.6	6.8	9.8	13.2	16.1



Hình 3. 2 Khoảng cách nRF trung bình giữa cây gốc và cây ML được xây dựng trên các bộ sắp hàng cùng tham số mô phỏng. AS là các bộ dữ liệu mô phỏng sử dụng cấu trúc cây bất đối xứng; SS là có bộ dữ liệu mô phỏng sử dụng cấu trúc cây đối xứng; miss: các bộ có dữ liệu mất mát. RP2: kết quả khi sử dụng lược đồ tạo bởi RatePartition với $d = 2$

Bảng 3. 4 Giá trị AICc và BIC của tám sắp hàng DNA thực khi sử dụng các lược đồ phân vùng khác nhau. Giá trị AICc (BIC) nhỏ hơn tương ứng với lược đồ phân vùng tốt hơn; giá trị tốt nhất được in đậm. Ký hiệu: NP: Không phân vùng; RP4 (RP5): RatePartition với hệ số phân chia $d = 4$ ($d = 5$); mPar: mPartition

Dữ liệu	**AICc**				**BIC**			
	NP	**RP4**	**RP5**	**mPar**	**NP**	**RP4**	**RP5**	**mPar**
Arctiina	102857	101410	101225	**99680**	104417	103641	103533	**101376**
Calisto	86492	85122	85162	**83037**	87733	86869	87060	**84394**
Choreutidae	121888	115505	115381	**113260**	122486	117065	117211	**114351**
Coenonymphina	128991	125154	125319	**121272**	129946	126699	126999	**122507**
Geometridae	384948	377148	377117	**375178**	387217	380356	380482	**377589**
Morpho	58872	55862	55756	**52633**	59351	57095	57122	**53539**
Noctuidae	206920	203296	203031	**192260**	208080	205614	205719	**193705**
Pieridae	277253	271196	271271	**268750**	278805	274178	274553	**270421**

Khi xét số lượng phân vùng trong các lược đồ, mPartition có trung bình 9,6 phân vùng trên mỗi sắp hàng; nhiều hơn so với thuật toán RatePartition với $d = 2$ và ít hơn trong các trường hợp khác của RatePartition.

Khoảng cách nRF trung bình của 10 sắp hàng ứng với cùng một bộ tham số mô phỏng được biểu diễn trong Hình 3. 2. Hình vẽ chỉ thể hiện kết quả của các lược đồ RP2 vì thuật toán RatePartition cho kết quả tương đương với hệ số phân chia khác nhau.

Khoảng cách nRF trong Hình 3. 2 cho thấy các nhánh trong có độ dài quá ngắn trong cây gốc có ảnh hưởng lớn đến độ chính xác của cây xây dựng được trên dữ liệu mô phỏng. Cụ thể, tất cả các cách phân hoạch đều giúp xây dựng cây phân loài gần giống cây gốc đối với các bộ dữ liệu được mô phỏng trên cây có các nhánh trong độ dài vừa phải (≥ 0.01 – các bộ dữ liệu AS1-AS4, AS13, AS14, SS1-SS4, SS13 và SS14). Ngược lại, khi chiều dài nhánh nhỏ, độ chính xác của các cây cực đại khả năng xây dựng được giảm đáng kể, như trong trường hợp các bộ AS6 và SS6 – là các sắp hàng được mô phỏng trên các cây có chiều dài nhánh trong chỉ 0,001 (tức là trung bình giữa hai đỉnh đầu mút chỉ có 0.001 biến đổi). Với các bộ dữ liệu không đầy đủ (AS1-miss, AS2-miss, SS1-miss, SS2-miss) được mô phỏng trên cây có độ dài nhánh 0.01, thông tin trên sắp hàng còn lại là đủ để kết quả thu được tương tự các bộ dữ liệu đầy đủ.

Ngoài ra, cấu trúc của các cây gốc cũng có một số tác động đến các kết quả của các thuật toán như ta thấy trên Hình 3. 2 - khoảng cách của các cây gốc có cấu trúc đối xứng cao hơn một chút so với cấu trúc bất đối xứng trong hầu hết các trường hợp và thậm chí là cao hơn nhiều với các bộ AS6 và SS6.

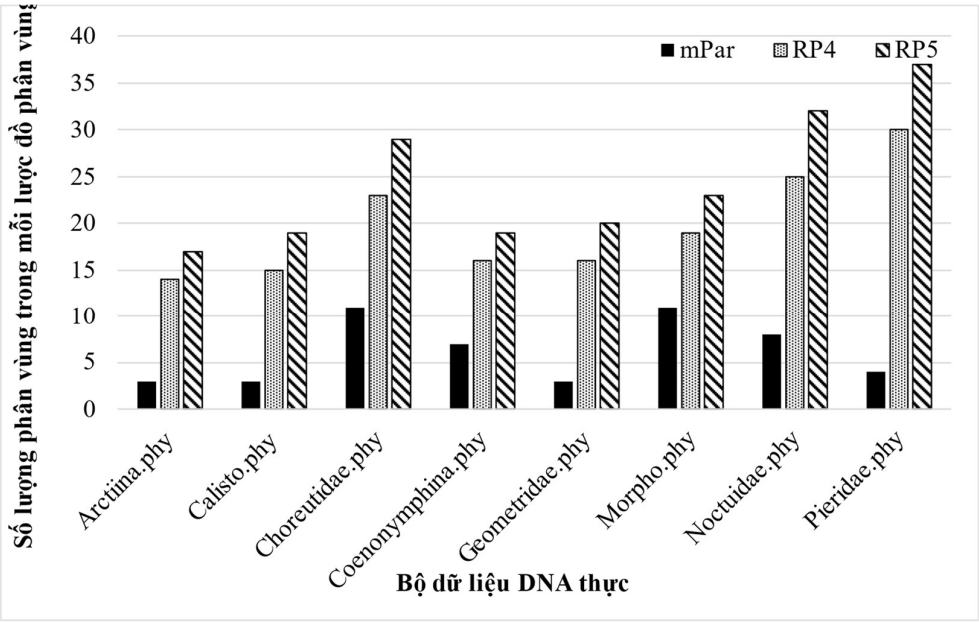
3.4.2 Dữ liệu DNA thực

Điểm AICc và BIC của cây cực đại khả năng xây dựng cho tám sắp hàng DNA thực khi sử dụng các chiến lược phân vùng bao gồm không phân vùng, sử dụng lược đồ của RatePartition và mPartition được tổng kết trong Bảng 3. 4. Kết quả cho thấy mPartition tốt hơn so với các phương pháp khác xét trên cả hai tiêu chuẩn; RatePartition đứng thứ hai và phương pháp không phân vùng cho kết quả kém nhất

trong cả tám trường hợp. Giá trị AICc và BIC của các cây có sử dụng lược đồ phân vùng thấp hơn đáng kể chứng tỏ các phương pháp phân hoạch tập vị trí đã giúp cải thiện độ chính xác của giá trị khả năng được tính toán, tương ứng là độ chính xác của cây được xây dựng.

Bảng 3. 5 Trung bình khoảng cách nRF trên các cặp cây xây dựng bằng các lược đồ phân vùng khác nhau.

	mPar	NP	RP4	RP5
mPar	-	0.080	0.076	0.089
NP	0.080	-	0.068	0.113
RP4	0.076	0.068	-	0.044
RP5	0.089	0.113	0.044	-



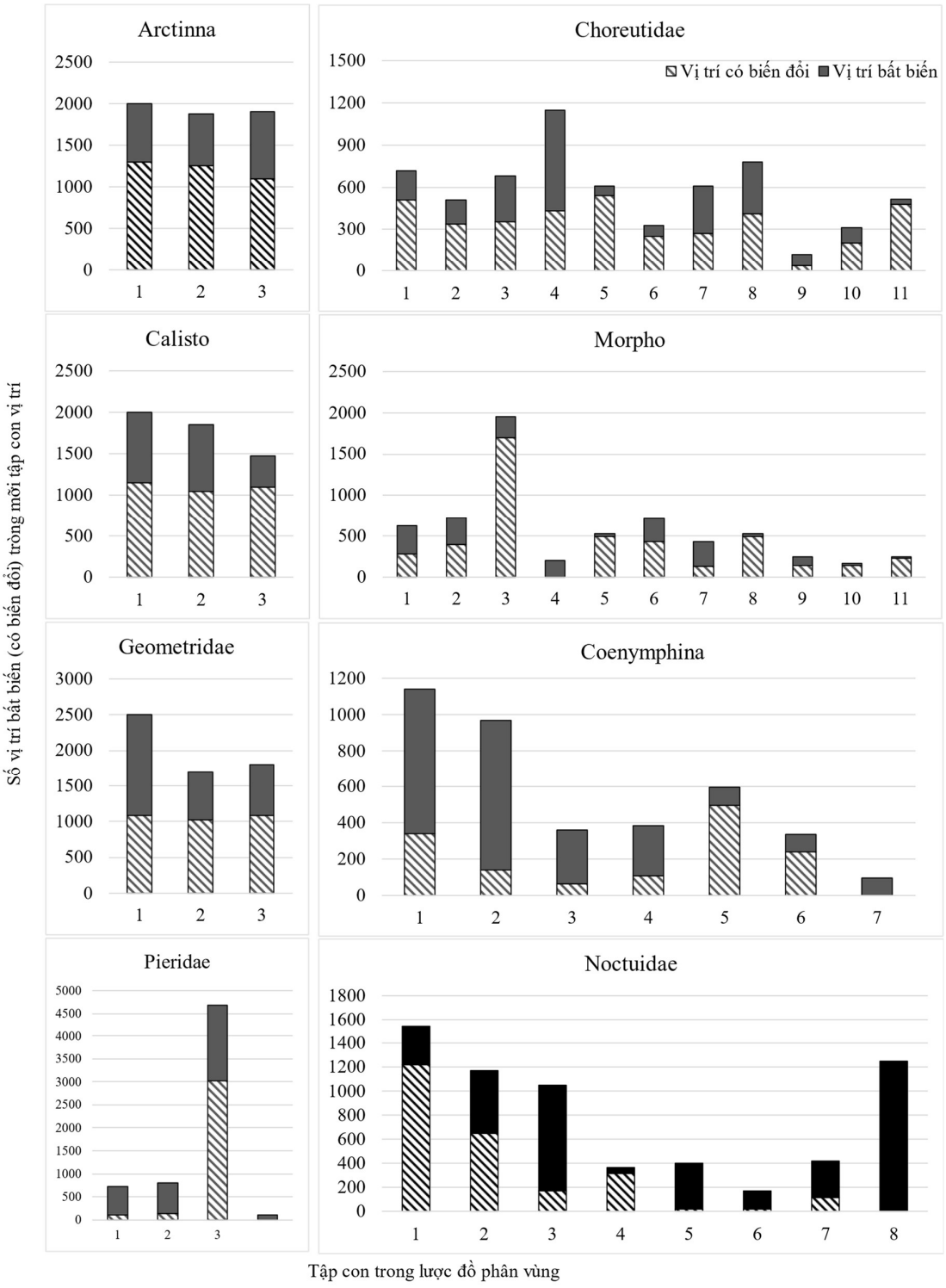
Hình 3. 3 Số lượng phân vùng trong mỗi lược đồ phân vùng tạo bởi thuật toán mPartition, RP4 và RP5

Trong số tám bộ dữ liệu, ngoại trừ các cây phân loài được xây dựng cho bộ dữ liệu Morpho có cấu trúc giống nhau với tất cả các lược đồ được sử dụng; cấu trúc cây của các sắp hàng còn lại đều có sự khác nhau giữa các lược đồ. Trung bình khoảng cách Robinson-Foulds trên tất cả các cặp cây được xây dựng bằng các lược đồ phân vùng khác nhau được tổng kết trong Bảng 3. 5. Có thể thấy, các lược đồ phân vùng được tạo ra bởi các chiến lược khác nhau có ảnh hưởng đến cấu trúc cây xây dựng được.

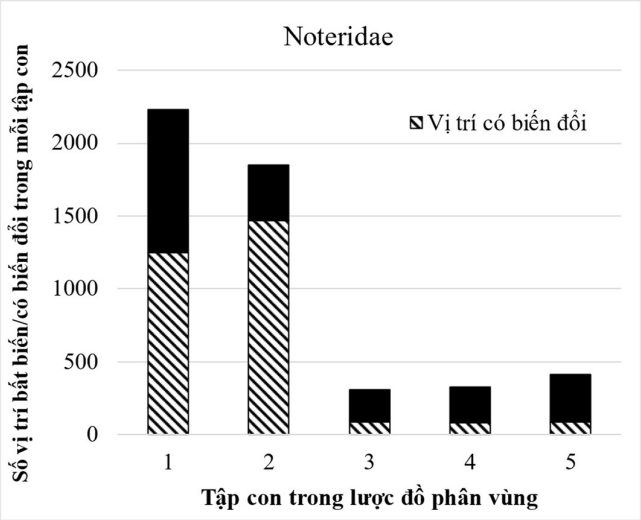
Số lượng phân vùng trong mỗi lược đồ phân vùng RP4, RP5 và mPartition được thể hiện trong Hình 3. 3. Khi d tăng, khoảng chênh lệch giữa tốc độ tiến hóa nhanh nhất và chậm nhất trong một nhóm giảm xuống, dẫn đến số lượng phân vùng của RatePartition tăng lên. Do vậy, số lượng phân vùng trong lược đồ RP5 luôn cao hơn lược đồ RP4 khoảng 20%. Trong khi đó, tương tự như kết quả trên dữ liệu mô phỏng, mPartition luôn tạo ra ít phân vùng hơn so với RP4 và RP5. Đặc biệt, chênh lệch về số lượng phân vùng nhiều nhất ở hai sắp hàng Geometridae và Pieridae. Điều này có thể là do các vị trí trong sắp hàng phù hợp với các mô hình tiến hóa ở mức độ tập trung cao, vì vậy số lượng các cụm vị trí ít hơn.

Khi xem xét sự phân bố của các vị trí bất biến trong các lược đồ phân vùng tạo bởi thuật toán mPartition, tất cả các phân vùng đều chứa vị trí bất biến. Hơn nữa, hầu hết các phân vùng đều chứa lượng lớn vị trí có biến đổi (Hình 3. 4). Đồng thời, các vị trí bất biến của cùng một loại nucleotit cũng được chia vào các tập hợp con khác nhau. Tức là, thuật toán mPartition đã giải quyết được nhược điểm của các phương pháp phân hoạch vị trí hoàn toàn dựa trên tốc độ tiến hóa (tất cả các vị trí bất biến nằm trong một phân vùng).

Thuật toán mPartition cũng được thử nghiệm trên bộ dữ liệu họ bọ cánh cứng thủy sinh Noteridae; đây là bộ dữ liệu đã được sử dụng để phân tích các phương pháp phân vùng khác nhau và phát hiện ra yếu điểm của phương pháp k-means lặp [49]. Sắp hàng Noteridae có 76 trình tự, gồm 53 loài bọ cánh cứng thủy sinh và một số chi và họ gần. Mỗi trình tự có độ dài 5011với khoảng 60% các vị trí trong sắp hàng là bất biến. Thuật toán mPartition đã chia tập các vị trí trong sắp hàng thành năm tập hợp

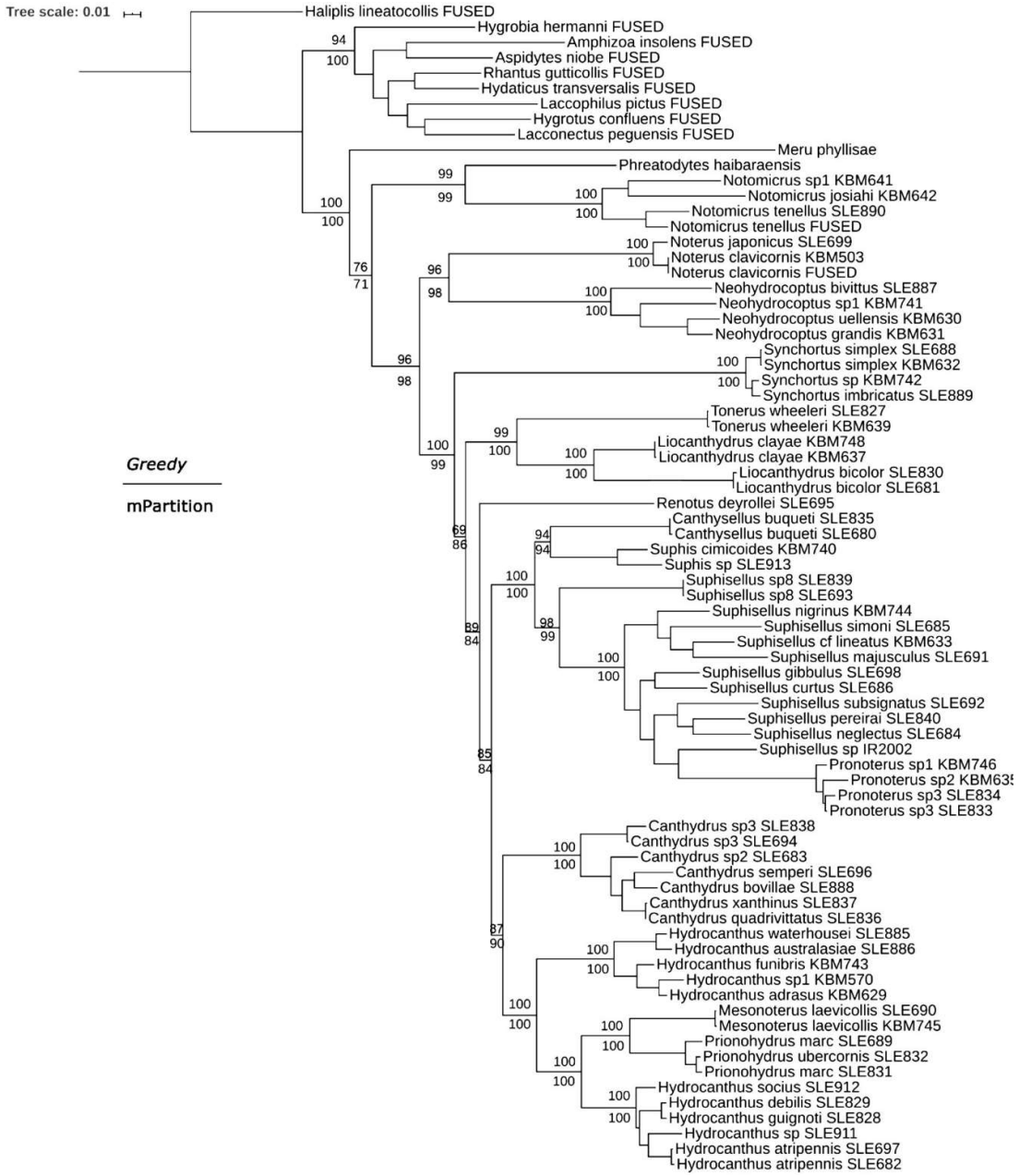


Hình 3. 4 Sự phân bố các vị trí bất biến và vị trí có biến đổi trong các lược đồ tạo bởi phương pháp mPartition



Hình 3. 5 Sự phân phối của các vị trí bất biến và vị trí có biến đổi trong các phân vùng của lược đồ phân vùng cho sắp hàng bọ cánh cứng thủy sinh Noteridae tạo bởi mPartition

con với kích thước từ 274 đến 2205 vị trí trong mỗi tập con. Các vị trí bất biến được phân phối vào tất cả các tập hợp con, mỗi tập hợp con chứa tất cả bốn loại bất biến khác nhau (Hình 3. 5 Sự phân phối của các vị trí bất biến và vị trí có biến đổi trong các phân vùng của lược đồ phân vùng cho sắp hàng bọ cánh cứng thủy sinh Noteridae tạo bởi mPartition). Thuật toán UFboot2 [33] được sử dụng để xây dựng cây bootstrap cho sắp hàng và lược đồ phân vùng mPartition (xem Hình 3. 6). Cây bootstrap sử dụng lược đồ mPartition nhìn chung nhất quán với cây bootstrap được trình bày trong [49], tức là, tất cả các chi được nhóm chính xác thành các nhánh đơn ngành với giá trị độ tin cậy nằm trong khoảng từ trung bình đến cao.



Hình 3. 6 Cây bootstrap của sắp hàng bọ cánh cứng thủy sinh Noteridae. Cây sử dụng lược đồ phân vùng thu được bằng phương pháp mPartition. Giá trị X và Y ở mỗi nhánh là giá trị độ tin cậy của nhánh sử dụng lược đồ phân vùng bằng phương pháp tham lam [31] (trong đó đó khoảng gen và vị trí nucleotit trong codon được định nghĩa sẵn) và mPartition.

3.4.3 Dữ liệu protein thực

Kết quả điểm AICc và BIC của các cây cực đại khả năng xây dựng bằng lược đồ mPartition và RatePartition cho 20 bộ dữ liệu protein thực được trình bày trong Bảng 3. 6 và Bảng 3. 7. Trong bảng, giá trị AICc (BIC) tốt nhất được in đậm để đánh dấu. Với 17 trong tổng số 20 trường hợp cho giá trị AICc/BIC nhỏ nhất, nhìn chung mPartition tạo ra các lược đồ phân vùng tốt hơn so với RatePartition. Trong ba sắp hàng mà lược đồ của RatePartition tốt hơn đều là các sắp hàng vừa và nhỏ - trong đó hai sắp hàng được ghép nối từ 10 gen (Chen 10 và Ran 10 đều có dưới 4000 axit amin) và một sắp hàng được ghép nối từ 20 gen (độ dài 6897 axit amin, tỉ lệ vị trí bất biến cao trên 40%); không có sắp hàng nào trong số các sắp hàng này chứa 30 và 40 gen. Điều này cho thấy mPartition có thể tạo lược đồ phân vùng tốt hơn RatePartition cho dữ liệu lớn, với các dữ liệu này, nhiều vị trí biến đổi có thể có quá trình tiến hóa phức tạp và chịu nhiều ảnh hưởng của nhiều nhân tố trong khi tỉ lệ vị trí bất biến cao đồng nghĩa với phân vùng bất biến lớn có thể dẫn đến giá trị khả năng cao hơn [49].

Khi phân tích sự phân bố của các vị trí bất biến, kết quả thu được tương tự như trên dữ liệu DNA thực, tức là, các vị trí bất biến được phân bổ vào các phân vùng khác nhau, mỗi phân vùng (trừ phân vùng bất biến) đều chứa vị trí bất biến của nhiều axit amin khác nhau, đồng thời chứa vị trí có biến đổi. Bảng 3. 8 trình bày ví dụ cho thấy sự phân bố của các vị trí bất biến trong lược đồ phân vùng của bốn sắp hàng Irissari.

Khi quan sát số lượng tập hợp con trong các sắp hàng, đặc điểm chung trên các bộ dữ liệu là số lượng tập con phân hoạch được tăng tương ứng với độ dài sắp hàng (tức là, sắp hàng càng dài thì lược đồ phân vùng càng có nhiều phân vùng). Khoảng cách Robinson-Fould nRF trung bình giữa các cây cực đại khả năng xây dựng bằng lược đồ của mPartition với các cây còn lại là 0.04 cho cả bốn trường hợp của RatePartition trong khi khoảng cách giữa các cây khi xây dựng bằng lược đồ phân vùng của thuật toán RatePartition thay đổi trong khoảng từ 0.011 tới 0.018. Chứng tỏ việc sử dụng lược đồ phân vùng khác nhau có ảnh hưởng tới cấu trúc cây, và các lược đồ của mPartition tạo ra khác biệt nhiều hơn so với giữa các lược đồ RatePartition với nhau.

Bảng 3. 6 Giá trị AICc của các cây được xây dựng sử dụng lược đồ tạo bởi mPartition và RatePartition trên các sắp hàng protein. RP2, RP3, RP4, và RP5 là RatePartition với $d = 2, 3, 4$ và 5.

Dữ liệu	mPartition	RP2	RP3	RP4	RP5
Ballesteros10	**191471**	195214	195194	194927	195024
Ballesteros20	**388710**	392348	392096	391497	391688
Ballesteros30	**776551**	789700	789219	788988	788905
Ballesteros40	**1156283**	1168122	1168039	1168049	1168168
Chen10	140998	139918	139795	139581	**139485**
Chen20	**261474**	263092	262892	264136	262680
Chen30	**543271**	547649	547676	547732	547589
Chen40	**713599**	720539	719695	719504	719491
Irissari10	**319393**	319990	320045	319949	319930
Irissari20	**410737**	416335	416108	416036	416072
Irissari30	**597394**	606780	606723	606285	606482
Irissari40	**765490**	782545	782584	782565	782568
Ran10	111426	**110952**	111000	110962	110978
Ran20	281988	281236	280911	**280853**	280893
Ran30	**409972**	411720	411531	411625	411730
Ran40	**593330**	599479	596830	596778	596721
Wu10	**189243**	190759	190621	190572	190640
Wu20	**586036**	590926	590380	590211	590114
Wu30	**755940**	766541	766258	766046	766239
Wu40	**1297141**	1304664	1304252	1303724	1303901

Bảng 3. 7 Giá trị BIC của các cây được xây dựng sử dụng lược đồ tạo bởi mPartition và RatePartition trên các sắp hàng protein. RP2, RP3, RP4, và RP5 là RatePartition với $d = 2, 3, 4$ và 5.

Dữ liệu	**mPartition**	**RP2**	**RP3**	**RP4**	**RP5**
Ballesteros10	**192705**	196112	196029	195972	196161
Ballesteros20	**389882**	393471	393254	392746	393020
Ballesteros30	**779124**	791613	791117	790771	790894
Ballesteros40	**1157577**	1169877	1170367	1170671	1170448
Chen10	141862	140732	140640	140457	**140361**
Chen20	**262655**	263939	263786	265036	263621
Chen30	**545438**	548933	549049	548860	548948
Chen40	**715922**	721613	720875	720616	720679
Irissari10	**321191**	321566	321529	321498	321513
Irissari20	**412975**	417985	417834	417776	417875
Irissari30	**600039**	608642	608504	608073	608314
Irissari40	**768843**	784612	784688	784897	785029
Ran10	112285	**111460**	111543	111528	111585
Ran20	282892	282086	281694	**281670**	281750
Ran30	**411766**	412645	412499	412637	412641
Ran40	**595342**	600450	597710	597733	597698
Wu10	**190485**	192026	191958	191897	192028
Wu20	**587996**	592526	592054	591943	591985
Wu30	**767489**	768338	768124	768059	768206
Wu40	**1297295**	1306733	1306434	1305946	1306042

Bảng 3. 8 Số lượng vị trí bất biến (Inv) và vị trí biến đổi (Var) trong các lược đồ của bốn sắp hàng Irissari

Irissari 10		**Irissari 20**		**Irissari 30**		**Irissari 40**	
#Inv	**#Var**	**#Inv**	**#Var**	**#Inv**	**#Var**	**#Inv**	**#Var**
696	1806	36	385	91	1030	727	0
601	1632	189	1531	487	1027	569	2062
648	644	282	301	1650	1532	287	1398
		484	710	476	2137	533	497
		405	328	651	1122	541	721
		528	804	944	904	407	1161
		970	1556	505	627	667	561
						127	652
						27	823
						1049	945
						486	1458

3.5 Kết luận

Để tăng độ chính xác của cây phân loài cực đại khả năng, bên cạnh cách làm truyền thống là sử dụng một mô hình biến đổi duy nhất cho tất cả các vị trí, hiện nay có hai hướng khác phổ biến khác là sử dụng mô hình đa ma trận và mô hình lược đồ phân vùng. Mô hình lược đồ phân vùng cho các sắp hàng lớn có lợi thế về tính toán so với phương pháp sử dụng mô hình đa ma trận và thường được sử dụng trong phân tích phát sinh loài bằng phương pháp cực đại khả năng. Xây dựng cây bằng các lược đồ khác nhau có thể ảnh hưởng đến cấu trúc cây và độ dài nhánh trong cây, việc xác định

một cách phân hoạch tốt có thể giúp cải thiện độ chính xác của cây cực đại khả năng xây dựng được. Luận án đã đề xuất thuật toán mPartition thực hiện việc phân hoạch tập vị trí của một sắp hàng để tạo lược đồ phân vùng cho một sắp hàng bất kỳ dựa trên tốc độ tiến hóa của các vị trí và mô hình thay thế tại mỗi vị trí. Thuật toán áp dụng được cho nhiều loại dữ liệu, kết quả của thuật toán là một lược đồ phân vùng được sử dụng trong quá trình xây dựng cây phân loài nhằm tăng tính chính xác của cây phân loài xây dựng được.

Các thực nghiệm được thực hiện trên cả hai loại dữ liệu trình tự DNA và protein. Kết quả cho thấy nhìn chung mPartition tạo ra các lược đồ phân vùng tốt hơn so với phương pháp chỉ sử dụng tốc độ biến đổi là RatePatition. Cụ thể, các cây được xây dựng bằng lược đồ phân vùng mPartition gần với cây gốc hơn RatePartition trên bộ dữ liệu mô phỏng so với các cây được xây dựng bằng lược đồ của RatePartition; trên các bộ dữ liệu thực, sử dụng lược đồ phân vùng tạo bởi mPartition giúp xây dựng cây ML tốt hơn so với RatePartition xét trên hai độ đo AICc và BIC.

Hơn nữa, các phương pháp dựa trên tốc độ tiến hóa có xu hướng nhóm các vị trí bất biến chỉ trong một vài phân vùng và điều này có thể dẫn đến sai lệch cấu trúc cây. Trong khi đó mPartition phân phối các vị trí bất biến vào tất cả các tập hợp con trong lược đồ phân vùng. Hầu hết tập hợp con này chứa một tỷ lệ lớn các vị trí có biến đổi giúp quá trình xây dựng cây tránh được những sai lệch tiềm ẩn trong cấu trúc cây. Một ưu điểm khác là mPartition không tạo ra những phân vùng/tập con quá nhỏ (dưới 50 vị trí) do vậy mỗi tập con có đủ thông tin để lựa chọn mô hình tiến hóa phù hợp. Kết quả này là một lợi thế vì tránh được việc phải ước lượng quá nhiều tham số trong quá trình xây dựng cây.

Chương 4: Phương pháp phân hoạch sắp hàng cho dữ liệu hệ gen

Chương 4 trình bày thuật toán phân hoạch sắp hàng gPartition. Thuật toán được thiết kế để phân hoạch nhanh dữ liệu lớn cỡ hệ gen. Tương tự như mPartition, bên cạnh tốc độ biến đổi, gPartition sử dụng mô hình tiến hóa phù hợp nhất để đánh giá sự tương đồng trong quá trình tiến hóa tại các vị trí khác nhau. Sự khác biệt giữa hai thuật toán là gPartition sử dụng phương pháp ước lượng nhanh và chỉ tính lại giá trị khả năng một lần. Nội dung chương được tổng hợp từ hai công trình [CT6] và [CT8].

4.1 Mở đầu

Thuật toán mPartition được trình bày trong chương 3 thực hiện việc phân hoạch dữ liệu và biểu diễn tốt quá trình tiến hóa không đồng nhất tại các vị trí khác nhau trong sắp hàng. Tuy nhiên, thuật toán chỉ thực hiện được trên các bộ dữ liệu cỡ nhỏ và vừa và trở nên tốn rất nhiều thời gian khi áp dụng cho những bộ dữ liệu mà trong đó mỗi trình tự có độ dài lên đến hàng trăm nghìn vị trí. Ví dụ, thuật toán mPartition không thể hoàn thành việc phân hoạch một sắp hàng có 187 trình tự, mỗi trình tự hơn 183 nghìn vị trí trong vòng 72 giờ. Trong khi đó, các bộ dữ liệu chứa tới hàng nghìn gen, với độ dài lên đến hàng triệu nucleotit/axit amin ngày càng trở nên phổ biến. Do vậy, cần có những phương pháp thực hiện việc phân hoạch cho những bộ dữ liệu có kích thước rất lớn trong khoảng thời gian ngắn nhất.

Đối với dữ liệu lớn, ta có thể sử dụng phương pháp phân cụm phân cấp [47]. Phương pháp này có đầu vào là sắp hàng với các khối dữ liệu được định nghĩa sẵn, mỗi khối dữ liệu là một tập hợp các vị trí; và đầu ra là lược đồ phân vùng mà mỗi phân vùng chứa một hay nhiều khối dữ liệu ban đầu. Trong quá trình thực hiện, mỗi khối dữ liệu (được định nghĩa từ đầu hay được tạo mới) được đại diện bởi một số thông tin bao gồm một số thông tin như tổng chiều dài các nhánh trong khối; một số tham số liên quan đến mô hình tiến hóa của khối như tần số của mỗi nucleotit/axit amin, tham số α mô tả phân phối Γ của mô hình tốc độ biến đổi. Độ tương đồng của các cụm tham

số này được tính toán và sử dụng để nhập các khối dữ liệu có độ tương đồng cao nhằm tạo ra lược đồ phân vùng tốt nhất.

Mặc dù thuật toán phân cụm phân cấp cải tiến rất nhiều so với phương pháp không phân vùng tuy nhiên thuật toán vẫn dựa trên các khối dữ liệu được xác định trước do người dùng định nghĩa. Thông tin này yêu cầu kiến thức chuyên môn nên không có sẵn trong nhiều trường hợp. Do vậy, luận án tập trung vào các phương pháp tìm lược đồ phân vùng dựa trên các đặc điểm của chính dữ liệu như RatePartition [32] hay mPartition. Thuật toán RatePartition chỉ thực hiện việc chia khoảng tốc độ tiến hóa, thời gian chạy của thuật toán khi đã có tốc độ biến đổi là hàm tuyến tính theo độ dài của sắp hàng, do vậy tổng thời gian chạy chủ yếu phụ thuộc vào thời gian tính tốc độ biến đổi tại mỗi vị trí.

Như đã đề cập trong 1.4 nhiều phương pháp được đề xuất để ước lượng tốc độ tiến hóa. Các nghiên cứu về phân hoạch dữ liệu khuyến khích sử dụng những cách ước lượng tốc độ tốc độ không sử dụng cây để đảm báo không gây ảnh hưởng đến mức độ tương đồng của các vị trí. Điển hình trong số này là phương pháp TIGER (Tree Independent Generation of Evolutionary Rates), phương pháp được đánh giá là ước lượng tốc độ ít sai lệch hơn và đã được sử dụng trong một các thuật toán phân hoạch sắp hàng [30], [32]. Nhược điểm của TIGER là có độ phức tạp là hàm bậc 2 của độ dài, do vậy không phù hợp để sử dụng cho dữ liệu cỡ hệ gen.

Để phân hoạch những sắp hàng cỡ hệ gen, có độ dài lên đến hàng triệu nucleotit; luận án đề xuất thuật toán gPartition và thuật toán ước lượng tốc độ fastTIGER. Thuật toán fastTIGER tính tốc độ dựa trên sự tương đồng giữa nội dung tại mỗi vị trí trong sắp hàng. fastTIGER được sử dụng kết hợp với mô hình tiến hóa để tự động phân hoạch sắp hàng cỡ hệ gen mà không yêu cầu xác định khối dữ liệu như cách tiếp cận phân cụm phân cấp.

4.2 Phương pháp

4.2.1 Thuật toán ước lượng nhanh tốc độ tiến hóa

Đầu vào: Sắp hàng đa trình tự tương đồng D có n trình tự, mỗi trình tự có độ dài l.
Đầu ra: Tốc độ tiến hóa tương đối cho từng vị trí trên sắp hàng

Tương tự như TIGER [52], thuật toán FastTIGER so sánh mẫu trong các vị trí với nhau. Nhưng thay vì so sánh nội dung của từng cặp vị trí, thuật toán sử dụng sự giống nhau giữa hai trình tự dựa trên số lượng vị trí có cùng trạng thái tại một vị trí trong hai trình tự. Ý tưởng của thuật toán là: với một cặp trình tự rất giống nhau, trạng thái của chúng phải giống hệt nhau ở các vị trí phát triển chậm, còn nếu hai trạng thái khác nhau tại vị trí i thì tốc độ tiến hóa r_i tại đó có khả năng cao hơn các vị trí khác. Như vậy nếu hai kí tự trong một vị trí có cùng trạng thái thuộc về các trình tự rất giống nhau, tốc độ sẽ giảm nhiều hơn so với trường hợp hai trình tự khác nhau.

Với sắp hàng D cho trước, phương pháp ước lượng nhanh fastTIGER gồm hai bước như sau:

- Bước 1: Tính ma trận tương đồng giữa các cặp trình tự. Là số lượng vị trí giống nhau giữa hai cặp trình tự bất kỳ.
- Bước 2: Tính tốc độ tại mỗi vị trí bằng công thức trong Thuật toán 4. 1

Theo cách tính này vị trí có tốc độ tiến hóa nhanh hơn sẽ có với giá trị lớn hơn và ngược lại; vị trí có tất cả các trạng thái cùng một giá trị có tốc độ bằng 0.

Ví dụ: cho một sắp hàng gồm 3 trình tự A_1, A_2, A_3, mỗi trình tự có 9 nucleotit như sau

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
A_1	A	A	C	T	G	A	T	C	A
A_2	A	G	C	G	G	G	-	C	G
A_3	T	A	C	G	G	G	A	G	C

Algorithm Thuật toán fastTIGER ước lượng nhanh tốc độ tiến hóa tại mỗi vị trí

input: Sắp hàng D có n trình tự, độ dài l
output: Danh sách tốc độ tiến hóa r_i của vị trí thứ i trong sắp hàng
procedure FASTTIGER(D)
    for each vị trí $i : D_i = d_{i1}d_{i2}\dots d_{il}$ **do** // Tính ma trận tương đồng
        for each $(j \neq iD_j = d_{j1}d_{j2}\dots d_{jl})$ **do**
            $f(i,j) = 0$;
            for $k = 1 \to l$ **do**
                if $d_{ik} == d_{jk}$ **then**
                    $f(i,j) = f(i,j) + 1$
                end if
            end for
        end for
    end for
    for $k = 1 \to l$ **do** // Tính tốc độ tại mỗi vị trí

$$r_i = 1 - \frac{\sum_{d_{ik}=d_{jk}} f(i,j)}{\sum_{i,j} f(i,j)}$$

    end for
end procedure

Thuật toán 4. 1 fastTIGER ước lượng nhanh tốc dộ tiến hóa tại mỗi vị trí

Khi ước lượng tốc độ tiến hóa bằng phương pháp TIGER ta thực hiện như sau:

$P(1) = \{\{1,2\},\{3\}\}$; $P(2) = \{\{1,3\},\{2\}\}$; $P(3) = \{\{1,2,\ 3\}\}$

$P(4) = \{\{1\},\{2,3\}\}$; $P(5) = \{\{1,2,3\}\}$; $P(6) = \{\{1\},\{2,3\}\}$

$P(7) = \{\{1\},\{3\}\}$; $P(8) = \{\{1,2\},\{3\}\}$; $P(9) = \{\{1\},\{2\},\{\ 3\}\}$

Sau đó tính điểm tương đồng phân hoạch:

$pa(1,2) = \frac{1}{2}$ vì trong hai tập $\{1,3\}$ và $\{2\}$ trong tập $P(2)$, chỉ có $\{2\} \subset \{1,2\} \in P(1)$

Tương tự:

$pa(1,3) = 0$; $pa(1,4) = \frac{1}{2}$; $pa(1,5) = 0$;

$pa(1,6) = \frac{1}{2}$; $pa(1,7) = 1$; $pa(1,8) = 1$; và $pa(1,9) = 1$

Do vậy, tốc độ tiến hóa của vị trí đầu tiên:

$$r_1 = {\frac{1}{2} + 0 + \frac{1}{2} + 0 + \frac{1}{2} + 1 + 1 + 1}\Big/{8} = \frac{9}{16}$$

Thực hiện tương tự ta có tốc độ tiến hóa của các vị trí còn lại trong sắp hàng.

Với phương pháp fastTIGER, đầu tiên ta xây dựng ma trận tương đồng giữa các cặp trình tự:

	4	3
4		4
3	4	

$$\Rightarrow \sum_{1 \le i < j \le 3} f(i,j) = 11$$

Sau đó tính giá trị tốc độ của từng vị trí

Vị trí đầu tiên có nội dung "AAT" có cùng trạng thái A tại vị trí 1 và 2

$$\Rightarrow r_1 = 1 - \frac{f(1,2)}{11} = \frac{7}{11}$$

Vị trí thứ hai có nội dung "AGA" có cùng trạng thái A ở vị trí 1 và 3

$$\Rightarrow r_2 = 1 - \frac{f(1,3)}{11} = \frac{8}{11}$$

Thực hiện tương tự ta có tốc độ của các vị trí còn lại trong sắp hàng là:

$r_3 = 0,\ r_4 = \frac{7}{11},\ r_5 = 0, r_6 = \frac{7}{11},\ r_7 = 1, r_8 = \frac{7}{11}$ và $r_9 = 1$

Giá trị tốc độ biến thiên trong khoảng từ 0 tới 1. Tốc độ bằng 0 tương ứng với trường hợp tất cả các trạng thái trong một ví trí đều bằng nhau và đây là một vị trí bất biến. Ngược lại, tốc độ bằng 1 xảy ra khi không có trình tự nào có chung trạng thái tại vị trí đó. Giá trị tốc độ của một vị trí lớn hơn tương ứng với vị trí có tốc độ tiến hóa nhanh hơn. Thuật toán fastTIGER có độ phức tạp $O(n \times n \times l)$ trong khi TIGER có

độ phức tạp $O(n \times l \times l)$. Do vậy, với những sắp hàng dài ($l \gg n$) và giá trị l rất lớn, fastTIGER thực hiện nhanh hơn nhiều so với TIGER.

4.2.2 gPartition

gPartition có đầu vào và đầu ra giống với thuật toán mPartiton, tuy nhiên gPartition chỉ nhận đầu vào là sắp hàng DNA và sắp hàng có thể có kích thước lớn đến rất lớn (hàng triệu vị trí). gPartition chia nhỏ sắp hàng đầu vào thành các sắp hàng con nhỏ,

Algorithm Thuật toán gPartition tìm lược đồ phân vùng cho sắp hàng DNA lớn

input: Sắp hàng D
output: Lược đồ phân vùng của D
procedure gPartition(D)
    fastTIGER(D)
    Phân tập vị trí thành các cụm hạt giống ss_i sử dụng tốc độ r_i
    Tối ưu mô hình Q_i cho sắp hàng con ss_i
    Loại bỏ các mô hình trùng lặp; tập các mô hình còn lại là **Q**
    for $i = 1 \rightarrow l$ **do**
        for each $Q_j \in$ **Q do**
            Tính giá trị khả năng $slh[i, j]$ của vị trí i với mô hình Q_j
        end for
    end for
    for $i = 1 \rightarrow l$ **do**
        if i là vị trí bất biến **then**
            Vị trí i thuộc vào nhóm thứ $j(j = \overline{1,n})$ với xác suất $\frac{e^{slh[i,j]}}{\sum_k e^{slh[i,k]}}$
        else
            Vị trí i thuộc vào nhóm có giá trị khả năng cao nhất
        end if
    end for
end procedure

Thuật toán 4. 3 tìm lược đồ phân vùng cho sắp hàng DNA lớn

mỗi sắp hàng chứa các vị trí có tốc độ tiến hóa tương tự nhau. Những sắp hàng con có cùng mô hình tiến hóa được gộp lại để tạo ra lược đồ phân vùng kết quả.

Thuật toán bao gồm bốn bước: bước 1, ước lượng tốc độ tiến hóa cho từng vị trí trên sắp hàng; bước 2, tạo phân vùng hạt giống, là các phân vùng nhỏ ban đầu dựa trên tốc độ tiến hóa cụ thể của mỗi vị trí; bước 3 lựa chọn mô hình - tìm ra các mô hình thay thế phù hợp nhất cho mỗi phân vùng con trong bước hai và hợp nhất các phân vùng có mô hình gần như tương đương để thu được các phân vùng lớn hơn; cuối cùng, tinh chỉnh lại các phân vùng dựa trên mô hình thay thế phù hợp hất của mỗi vị trí để tăng giá trị khả năng của lược đồ phân vùng. Thuật toán 4. 3 tìm lược đồ phân vùng cho sắp hàng DNA lớn tóm tắt quy trình phân hoạch tự động để tạo lược đồ phân vùng cho sắp hàng A.

Trong bước *ước lượng tốc độ* (dòng 4) có thể sử dụng bất kỳ phương pháp nào để ước lượng tốc độ tiến hóa cho các vị trí trong sắp hàng. Tuy nhiên, vì thuật toán này cần tính tốc độ cho sắp hàng kích thước cỡ hệ gen, phương pháp được khuyên dùng là fastTIGER.

Bước *phân cụm hạt giống* (dòng 5) nhóm các vị trí sử dụng tốc độ tiến hóa tính được trong bước trên. Trước tiên khoảng giá trị tốc độ ban đầu được chia thành các khoảng nhỏ theo giá trị tăng dần. Mỗi vị trí được xếp vào nhóm tương ứng với khoảng tốc độ mà vị trí thuộc về.

Bước *lựa chọn mô hình* (dòng 6 và 7) lựa chọn mô hình thay thế phù hợp nhất cho mỗi phân vùng hạt giống bằng thuật toán ModelFinder [70]. Sau đó, mỗi mô hình sẽ được chuyển đổi thành một vectơ 8 tham số như trong mô hình GTR. Các vectơ này được dùng để tính hệ số tương quan Pearson giữa các cặp vectơ. Nếu hệ số tương quan giữa một cặp vectơ lớn hơn ngưỡng tương đương định trước, thì hai phân vùng tương ứng được nhập thành một. Sau khi hoàn thành bước này, lược đồ phân vùng mới còn lại một số phân vùng tương ứng với một tập mô hình khác nhau đôi một.

Bước điều chỉnh phân vùng: điều chỉnh vị trí ứng với phân vùng phù hợp nhất để thu được phân vùng kết quả. Đầu tiên, thuật toán tính giá trị khả năng tại mỗi vị trí ứng

với từng mô hình trong danh sách mô hình còn lại. Sau đó, các vị trí có biến đổi được gán cho phân vùng tương ứng với mô hình có giá trị khả năng cao nhất trong khi các vị trí bất biến được gán cho các phân vùng theo mô hình phân phối xác suất.

Phương pháp gPartition có thể áp dụng cho dữ liệu DNA vì các tham số của mô hình đều được tính riêng cho từng bộ dữ liệu, từ đó ta có thể nhập các mô hình gần như tương đương để giảm số lượng phân vùng mà vẫn sử dụng được thông tin về mô hình tiến hóa tại các vị trí. Điều này giúp cho kết quả phân hoạch tốt hơn so với chỉ sử dụng tốc độ tiến hóa như trong RatePartition.

4.3 Thực nghiệm

4.3.1 Dữ liệu

Tám sắp hàng DNA được sử dụng để thử nghiệm hai thuật toán fastTIGER và gPartition. Trong đó có một sắp hàng ngắn (độ dài chưa đến 6000), nhiều trình tự (164 trình tự) và bảy sắp hàng có độ dài từ vừa đến lớn, sắp hàng dài nhất có hơn một triệu vị trí, sắp hàng có nhiều trình tự nhất có 187 trình tự. Chi tiết của tám bộ dữ liệu được mô tả trong Bảng 4. 1.

Bảng 4. 1 Tám bộ dữ liệu dùng trong thực nghiệm

	Dữ liệu	**#Số trình tự**	**#Vị trí**	**#Số gen**	**Bài báo**
1	Geometridae	164	5998	8	[88]
2	Osteichthyes	61	19997	61	[92]
3	Actinopterygii	27	149366	491	[93]
4	Aculeata	187	183747	807	[94]
5	Neoaves	33	539526	1541	[95]
6	Phasianidae	18	614159	1501	[96]
7	Xenopsminutus	8	825804	1366	[97]
8	Spermatophyta	32	1296042	3924	[82]

4.3.2 Tham số cài đặt

Trong bước phân cụm hạt giống: số cụm hạt giống được thiết lập bằng $k = 0.01 \times var$ trong đó var là số vị trí có biến đổi. Như vậy trung bình mỗi cụm hạt giống có khoảng 100 vị trí có biến đổi.

Giá trị tốc độ thay đổi từ 0 đến 1, khoảng giá trị này được chia thành các khoảng con, mỗi khoảng tương ứng với một phân vùng vị trí. Cụ thể, vị trí có biến đổi i có tốc độ $r_i > 0$ được phân vào phân vùng thứ s nếu $\frac{\text{S-1}}{\text{k}} \leq \text{r}_\text{i} < \frac{\text{S}}{\text{k}}$.

Để tránh tạo ra các cụm quá nhỏ, mỗi phân vùng có số lượng ít nhất là 50 vị trí như trong bài báo [98]. Các phân vùng có ít hơn 50 vị trí được nhập vào phân vùng nằm bên cạnh để đảm bảo số lượng vị trí tối thiểu trong mỗi phân vùng.

Ngưỡng tương đương của các mô hình được đặt bằng 0.9999. Tức là, hai phân vùng có hệ số tương quan giữa cặp mô hình tiến hóa tương ứng lớn hơn 0.9999 được nhập lại với nhau.

Với thuật toán RatePartition, hệ số phân chia $d = 4$ dựa trên các đánh giá trong [32].

Các tác vụ tính toán lựa chọn mô hình tiến hóa và tính giá trị cực đại khả năng được thực hiện bằng phần mềm IQTREE [22]

Thực nghiệm được triển khai trên máy tính vi xử lý 18-core Intel Xeon, tốc độ 2.30Hz và 128GB RAM.

4.3.3 Thực nghiệm

Một số thực nghiệm đã được thực hiện để đánh giá thuật toán ước lượng tốc độ tiến hóa fastTIGER và thuật toán phân hoạch sắp hàng gPartition:

- Đối với fastTIGER, thời gian để tính ra kết quả của thuật toán được so sánh với thời gian cần dùng để tính ra kết quả của thuật toán TIGER [52]; tốc độ tính được cũng được so sánh với tốc độ tính bởi TIGER và so sánh kết quả khi sử dụng hai loại tốc độ này trong các thuật toán tạo lược đồ phân vùng.

- Đối với gPartition, lược đồ phân vùng tạo bởi thuật toán được so sánh với các lược đồ kết quả của mPartition và RatePartition trên hai thang đo: thời gian thực hiện và độ tốt của cây suy luận được thông qua độ đo AICc.

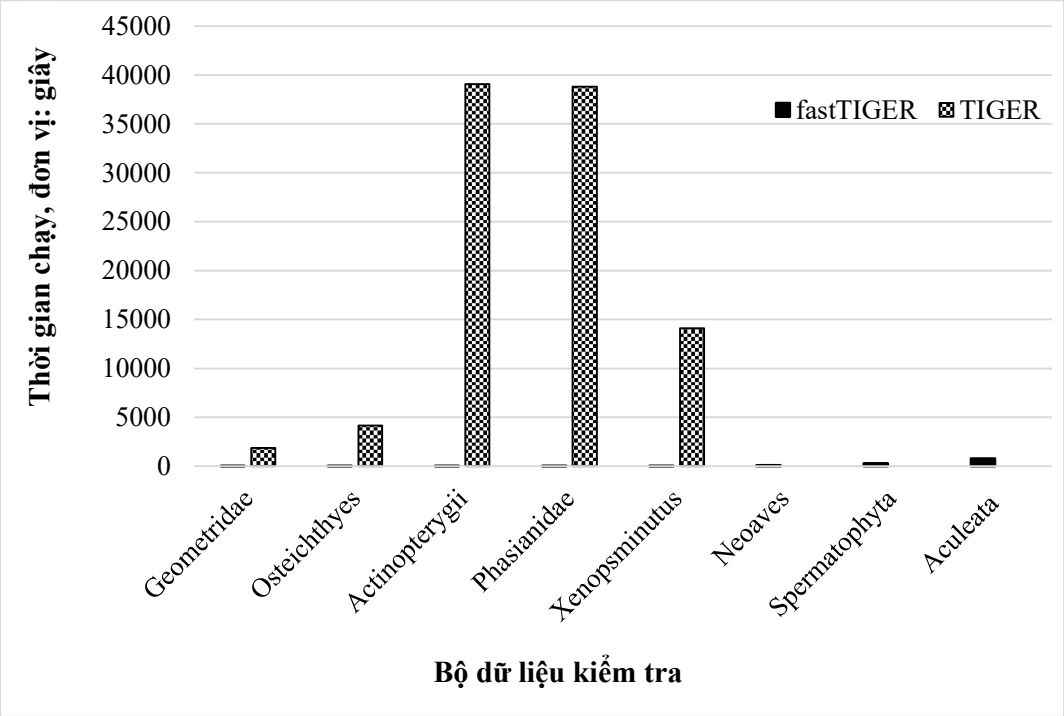
Cụ thể:

Tính hợp lý của cách tiếp cận mới trong việc tính tốc độ được đánh giá qua hệ số tương quan của các cặp tốc độ đối với mỗi sắp hàng. Đồng thời, các tập tốc độ đã tính cho từng vị trí lần được được sử dụng cho hai thuật toán phân hoạch sắp hàng mPartition và RatePartition để tìm lược đồ phân vùng. Các lược đồ phân vùng kết quả được sử dụng để xây dựng cây cực đại khả năng và đánh giá sự tương đồng so với kết quả đã thu được trước đó.

Với ba thuật toán RatePartition [32], mPartition và gPartition, luận án so sánh tổng thời gian cần thiết để tìm lược đồ phân vùng cho mỗi sắp hàng; độ tốt của các lược đồ được đánh giá thông qua điểm AIC của cây cực đại khả năng xây dựng bởi các lược đồ phân vùng tương ứng. Ngoài ra sự phân bố của các vị trí bất biến trong các phân vùng cũng được kiểm tra để đảm bảo thuật toán không tạo phân vùng bất biến.

4.4 Kết quả

Trong số tám sắp hàng, ba sắp hàng lớn nhất là Aculeata, Neoaves và Spermatophyta chưa tính được tốc độ bằng phương pháp TIGER sau 72 giờ, vì vậy kết quả của hai thuật toán chỉ được so sánh trên năm sắp hàng còn lại. Thời gian tính tốc độ tiến hóa của TIGER và fastTIGER thể hiện trong Hình 4. 1. Dễ thấy, tốc độ của fastTIGER nhanh hơn TIGER nhiều lần trong cả năm sắp hàng trường hợp đã hoàn thành. Ví dụ, để tính tốc độ tiến hóa cho sắp hàng Phasianidae, TIGER cần hơn 38 nghìn giây trong khi fastTIGER chỉ cần 40 giây để hoàn thành. Ngay cả đối với Geometridae – là sắp hàng có tỉ lệ giữa độ dài và số trình tự thấp hơn các sắp hàng khác – thời gian chạy TIGER cũng gấp hơn 100 lần so với fastTIGER. Trong cả năm trường hợp có thể sử dụng thuật toán TIGER, thời gian để fastTIGER hoàn thành việc tính toán chưa tới một phút.



Hình 4. 1 Thời gian (giây) tính tốc độ tiến hóa cho mỗi vị trí bằng thuật toán TIGER và fastTIGER.

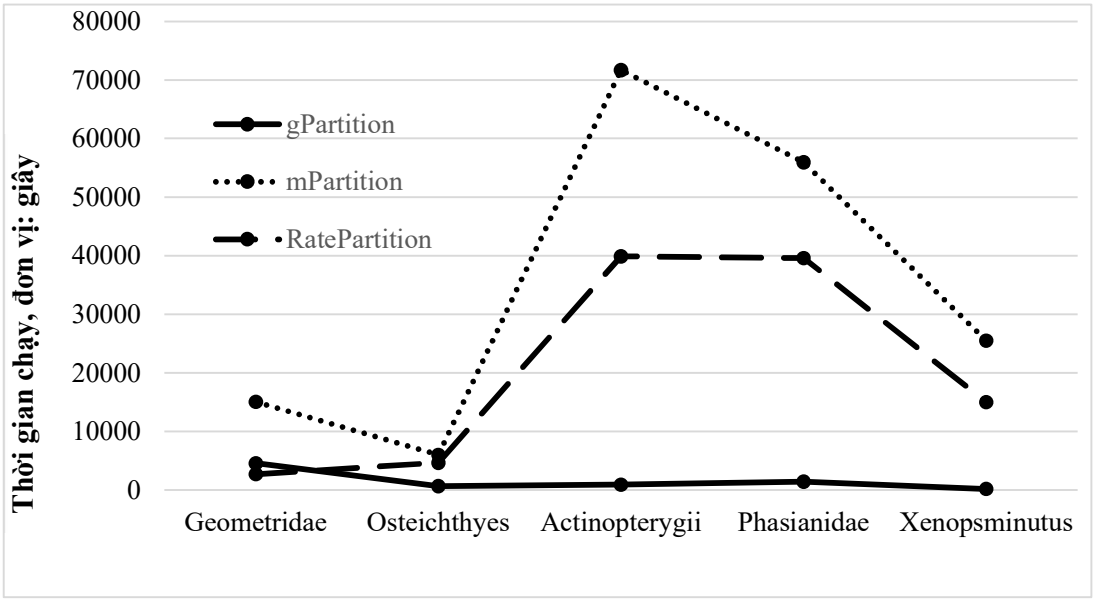
Giá trị tương quan giữa hai tốc độ đã ước lượng cho năm sắp hàng thay đổi trong khoảng từ 65% đến 93% như được trình bày trong Bảng 4. 2. Giá trị tương quan vừa đến cao cho thấy có mối liên hệ giữa ý tưởng của hai cách tính.

Bảng 4. 2 Hệ số tương quan giữa tốc độ tiến hóa ước lượng được bằng fastTIGER và TIGER trong năm bộ dữ liệu thực nghiệm.

Sắp hàng	**Hệ số tương quan**
Geometridae.	82%
Osteichthyes	73%
Actinopterygii	65%
Phasianidae	89%
Xenopsminutus	93%

Trong thực nghiệm đánh giá khả năng sử dụng tốc độ fastTIGER thay cho tốc độ TIGER. Kết quả thu được khi sử dụng fastTIGER tương đồng với kết quả sử dụng TIGER đã thử nghiệm với thuật toán mPartition. Cụ thể, sử dụng lược đồ mPartition giúp xây dựng cây tốt hơn lược đồ RatePartition ở cả hai loại tốc độ. Hầu hết các bộ dữ liệu không có sự khác biệt lớn khi sử dụng tốc độ fastTIGER hoặc TIGER ngoại trừ bộ dữ liệu Xenopsminutus khi áp dụng cho phương pháp mPartition. Trong trường hợp này, điểm AICc của cây được xây dựng bằng lược đồ dùng fastTIGER giảm đáng kể (hơn 26%) so với cây dùng TIGER. Các kết quả trên chứng tỏ tốc độ tiến hóa fastTIGER có thể được sử dụng thay cho TIGER để ước lược tốc độ tiến hóa, thuật toán đặc biệt phù hợp với sắp hàng có độ dài lớn.

Hình 4. 2 Thời gian chạy (giây) của các thuật toán RatePartition, mPartition (dùng tốc độ TIGER) và gPartition (dùng fastTIGER).. thể hiện thời gian phân hoạch sắp hàng của các phương pháp RatePartition, mPartition và gPartition theo cấu hình hiện



Hình 4. 2 Thời gian chạy (giây) của các thuật toán RatePartition, mPartition (dùng tốc độ TIGER) và gPartition (dùng fastTIGER).

tại (tức là RatePartition và mPartition sử dụng tốc độ TIGER, gPartition sử dụng tốc độ fastTIGER).

Dữ liệu	**Kích thước**		**AICc tại mỗi vị trí**		
	n	l	gPartition	mPartition	RatePartition
Geometridae	164	5998	**60.7**	62.5	62.9
Osteichthyes	61	19997	**29.3**	31.4	32.0
Actinopterygii	27	149366	8.3	**7.8**	9.6
Aculeata	187	183747	**93.3**	NA	NA
Neoaves	33	539526	**4.8**	NA	NA
Phasianidae	18	614159	3.7	**3.4**	3.9
Xenopsminutus	8	825804	2.78	**2.76**	2.81
Spermatophyta	38	1296042	**29.4**	NA	NA

Ta thấy thuật toán gPartition nhanh hơn nhiều so với mPartition trong tất cả trường hợp; thuật toán cũng nhanh hơn RatePartition ở bốn trường hợp và chậm hơn chỉ ở sắp hàng Geometridae, là sắp hàng có tỉ lệ độ dài : số trình tự nhỏ nhất. Với sắp hvàng này thời gian để ước lượng tốc độ bằng TIGER tuy lâu hơn fastTIGER nhưng vẫn khá nhỏ so với thời gian hơn để tìm mô hình phù hợp và tính giá trị khả năng cho từng vị trí trong sắp hàng. Với các trình tự dài và số lượng trình tự ít hơn, gPartition nổi trội trong ba cách phân hoạch về thời gian thực hiện. Ví dụ: với sắp hàng Xenopsminutus là sắp hàng dài nhất trong số năm sắp hàng có thể tính toán bằng phương pháp TIGER (có độ dài 825804 và 8 trình tự) được phân vùng bởi gPartition trong 186 giây trong khi cần tới 15000 giây để phân vùng bằng RatePartition và 25500 giây để phân vùng bằng mPartition. gPartition hoàn thành việc phân hoạch sắp hàng trong vòng 22 giờ đối với bộ lớn nhất (bộ Aculeata).

Điểm AICc của các cây cực đại khả năng được xây dựng bằng lược đồ phân vùng tạo bởi RatePartition, mParition và gPartition được tổng kết trong

Bảng 4. 3. Trong bảng, giá trị AICc của cây được chia cho số lượng vị trí trong mỗi sắp hàng để ra điểm AICc trung bình trên mỗi vị trí trong sắp hàng. Thông tin điểm

Dữ liệu	**Kích thước**		**AICc tại mỗi vị trí**		
	n	l	gPartition	mPartition	RatePartition
Geometridae	164	5998	**60.7**	62.5	62.9
Osteichthyes	61	19997	**29.3**	31.4	32.0
Actinopterygii	27	149366	8.3	**7.8**	9.6
Aculeata	187	183747	**93.3**	NA	NA
Neoaves	33	539526	**4.8**	NA	NA
Phasianidae	18	614159	3.7	**3.4**	3.9
Xenopsminutus	8	825804	2.78	**2.76**	2.81
Spermatophyta	38	1296042	**29.4**	NA	NA

AICc trung bình ứng với thuật toán RatePartition và mPartition không có ở ba sắp hàng lớn nhất vì quá trình tính tốc độ chưa hoàn thành. Lưu ý rằng điểm AICc nhỏ hơn tương ứng với cây tốt hơn. Kết quả trong bảng cho thấy gPartition giúp tạo lược đồ phân vùng tốt hơn RatePartition trong cả năm trường hợp, và điểm AICc của gPartition và mPartiton không có sự khác biệt quá lớn. Mặc dù vậy, thuật toán mPartition cho kết quả tốt hơn trên các sắp hàng dài hơn. Kết quả này là hợp lý vì mPartition có quá trình xác định mô hình tiến hóa tại các vị trí chi tiết hơn. Việc này sẽ giúp ích cho quá trình đánh giá độ tương đồng của các vị trí tốt hơn.

Bảng 4. 3 Giá trị AICc trung bình trên mỗi vị trí của cây ML sử dụng ba cách phân hoạch vị trí. Giá trị tốt nhất được tô đậm để đánh dấu.

4.5 Kết luận

Các bộ dữ liệu lớn cung cấp nhiều thông tin hơn cho các nhà nghiên cứu; đồng thời cũng là thách thức trên phương diện tính toán vì để tối ưu cùng lúc mô hình và cấu trúc cây là công việc có độ phức tạp NP-khó. Khi độ dài của của một sắp hàng lên đến hàng trăm nghìn hay hàng triệu vị trí, các phương pháp phân hoạch sắp hàng hiện

tại không thể xử lý và tạo ra các lược đồ phân vùng trong thời gian phù hợp (72 giờ - theo thực nghiệm đã làm). Luận án đã đề xuất thuật toán ước lượng nhanh tốc độ tiến hóa fastTIGER và thuật toán gPartition, hai thuật toán này có thể áp dụng trên những sắp hàng lớn và rất lớn.

Thuật toán fastTIGER có thể sử dụng cho các sắp hàng DNA hoặc sắp hàng axit amin. Thử nghiệm trên một số bộ dữ liệu cho thấy tốc độ ước lượng được bởi thuật toán fastTIGER có thể sử dụng thay cho tốc độ TIGER trong các thuật toán phân hoạch sắp hàng, điểm mấu chốt là thời gian tính toán của thuật toán nhanh hơn nhiều lần so với TIGER. Tương tự , thuật toán gPartition có thể phân hoạch sắp hàng DNA có độ dài lên tới hàng triệu vị trí trong thời gian chấp nhận được. Hơn nữa, thuật toán không chỉ nhanh hơn nhiều so với các thuật toán phân vùng hiện có mà còn tạo lược đồ phân vùng tốt hơn thuật toán phân hoạch chỉ sử dụng tốc độ tiến hóa. Do đó, phương pháp này có thể sử dụng để tạo lược đồ phân vùng biểu diễn tính không đồng nhất cho sắp hàng lớn, điều khó thực hiện bởi các thuật toán hiện nay.

Kết luận

Xây dựng cây phân loài sát với thực tế rất quan trọng trong các bài toán phân tích dữ liệu sinh học phân tử và là một trong những vấn đề chính của tin sinh học. Sử dụng mô hình tiến hóa không phù hợp với bản chất của dữ liệu có thể ảnh hưởng đáng kể đến tính đúng đắn của cây phân loài cực đại khả năng xây dưng được. Do đó, các mô hình tiến hóa mới và cách thức biểu diễn mới được nghiên cứu, đề xuất để mô tả tốt nhất quá trình tiến hóa trong một bộ dữ liệu.

Luận án tập trung vào việc nghiên cứu và đưa ra các đề xuất làm tăng tính đúng đắn khi biểu diễn quá trình tiến hóa trong một sắp hàng trên hai khía cạnh: một là đề xuất mô hình đơn biểu diễn tốt hơn quá trình tiến hóa của một nhóm dữ liệu (chi Flavivirus), hai là đề xuất thuật toán phân hoạch để biểu diễn quá trình tiến hóa không đồng nhất của dữ liệu nói chung.

Luận án đã đề xuất mô hình thay thế FLAVI dành riêng cho các vi rút trong chi một chi chứa nhiều virút gây dịch bệnh nghiêm trọng trong khu vực. Thực nghiệm cho thấy mô hình FLAVI giúp xây dựng cây cực đại khả năng tốt hơn đáng kể so với các mô hình hiện có.

Luận án cũng đề xuất hai phương pháp mPartition và gPartition để phân hoạch sắp hàng kích thước lớn và rất lớn. Hai phương pháp đều sử dụng kết hợp tốc độ tiến hóa và mô hình tiến hóa tại từng vị trí để phân hoạch tập vị trí, giúp tạo lược đồ phân vùng tốt hơn so với thuật toán chỉ sử dụng tốc độ tiến hóa. Tuy nhiên các thuật toán được đề xuất vẫn còn một số nhược điểm chưa giải quyết được: thuật toán mPartition có thể chạy được với mọi loại dữ liệu nhưng chưa áp dụng được cho dữ liệu hệ gen, ngược lại, gPartition áp dụng được cho dữ liệu hệ gen nhưng lại chỉ dùng được cho dữ liệu DNA.

Bên cạnh đó, luận án đề xuất phương pháp ước lượng nhanh tốc độ tiến hóa fastTIGER. Phương pháp ước lượng tốc độ mới có độ phức tạp nhỏ hơn và có thể tính ra kết quả trong thời gian rất ngắn. Ngoài việc ứng dụng trong các thuật toán

phân hoạch sắp hàng, phương pháp ước lượng nhanh tốc độ tiến hóa fastTIGER cũng có những ứng dụng riêng; ví dụ: kiểm tra và xóa những vị trí có mẫu đặc biệt để giảm tính dị biệt của các vị trí trong sắp hàng.

Danh mục các công trình khoa học của tác giả liên quan đến luận án

[CT1] **Le Kim Thu**, Cuong Dang Cao, and Vinh Le Sy. 2018. "Building a specific amino acid substitution model for dengue viruses." 2018 10th International Conference on Knowledge and Systems Engineering (KSE) 242-246.

[CT2] **Le Kim Thu**, and Vinh Le Sy. 2020. "A protein alignment partitioning method for protein phylogenetic inference." 2020 RIVF International Conference on Computing and Communication Technologies (RIVF) 1-5.

[CT3] **Le Kim Thu**, Vinh Le Sy, Dong Do Duc, Thang Bui Ngoc, and Phuong Thao Nguyen Thi. 2020. "iK-means: an improvement of the iterative k-means partitioning algorithm." 2020 12th International Conference on Knowledge and Systems Engineering (KSE) 300-305.

[CT4] **Le Kim Thu**, and Vinh Le Sy. 2020. "FLAVI: An Amino Acid Substitution Model for Flaviviruses." Journal of molecular evolution 88 (5): 445-452.

[CT5] **Le Kim Thu**, and Vinh Le Sy. 2020. "mPartition: A Model-based method for partitioning alignments." Journal of Molecular Evolution 88 (8): 641-652.

[CT6] **Le Kim Thu**, and Vinh Le Sy. 2021. "fastTIGER: A rapid method for estimating evolutionary rates of sites from large datasets." 2021 13th International Conference on Knowledge and Systems Engineering (KSE).

[CT7] **Le Kim Thu**, and Vinh Le Sy. 2022. "A protein secondary structure-based algorithm for partitioning large protein alignments." 2022 14th International Conference on Knowledge and Systems Engineering (KSE) 1-5.

[CT8] **Le Kim Thu**, Diep Hoang Thi, Dong Do Duc, Thang Bui Ngoc, Phuong Thao Nguyen Thi, and Vinh Le Sy. 2022. "gPartition: An Efficient Alignment Partitioning Program for Genome Datasets." VNU Journal of Science: Computer Science and Communication Engineering.

Tài liệu tham khảo

[1] L. Harvey *et al.*, *Molecular Cell Biology*, 8th ed. W. H. Freeman, 2016.

[2] P. Lemey, M. Salemi, and A.-M. Vandamme, *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*, 2nd ed. Cambridge University Press, 2009.

[3] J. Felsenstein, "Evolutionary trees from DNA sequences: a maximum likelihood approach," *J. Mol. Evol.*, vol. 17, pp. 368–376, 1981.

[4] M. Hasegawa, H. Kishino, and T. Yano, "Dating of the human-ape splitting by a molecular clock of mitochondrial DNA," *J. Mol. Evol.*, vol. 22, no. 2, pp. 160–174, 1985.

[5] K. Tamura and M. Nei, "Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees.," *Mol. Biol. Evol.*, vol. 10, no. 3, pp. 512–526, May 1993.

[6] S. Tavare, "Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences," *Lect. Math. Life Sci.*, vol. 17, pp. 57–86, 1986.

[7] M. . Dayhoff, R. . Schwartz, and B. . Orcutt, "A Model of Evolutionary Change in Proteins," *Atlas protein Seq. Struct.*, vol. 5, pp. 345–351, 1987.

[8] S. Whelan and N. Goldman, "A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach," *Mol. Biol. Evol.*, vol. 18, no. 5, pp. 691–699, 2001.

[9] S. Q. Le and O. Gascuel, "An improved general amino acid replacement matrix," *Mol. Biol. Evol.*, vol. 25, no. 7, pp. 1307–1320, 2008.

[10] J. DT, T. WR, T. JM, D. T. Jones, W. R. Taylor, and J. M. Thornton, "The rapid generation of mutation data matrices from protein sequences," *Comput Appl Biosci*, vol. 8, pp. 275–282, 1992.

[11] D. C. D. Nickle *et al.*, "HIV-specific probabilistic models of protein evolution," *PLoS One*, vol. 2, no. 6, p. e503, 2007.

[12] D. Cuong, L. Quang, G. Olivier, and V. Le, "FLU, an amino acid substitution model for influenza proteins," *BMC Evol. Biol.*, vol. 10, p. 99, 2010.

[13] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, no. 5, pp. 1792–1797, 2004.

[14] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Math. Biosci.*, vol. 53, pp. 131–147, 1981.

[15] D. T. Hoang, L. S. Vinh, T. Flouri, A. Stamatakis, A. von Haeseler, and B. Q. Minh, "MPBoot: fast phylogenetic maximum parsimony tree inference and

bootstrap approximation.," *BMC Evol. Biol.*, vol. 18, no. 1, p. 11, Feb. 2018.

[16] A. Criscuolo, "A fast alignment-free bioinformatics procedure to infer accurate distance-based phylogenetic trees from genome assemblies," *Res. Ideas Outcomes*, vol. 5, p. e36178, 2019.

[17] P. A. Goloboff, S. A. Catalano, and A. Torres, "Parsimony analysis of phylogenomic datasets (II): evaluation of PAUP*, MEGA and MPBoot.," *Cladistics*, vol. 38, no. 1, pp. 126–146, Feb. 2022.

[18] G. Olivier, "BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data," *Mol. Biol. Evol.*, vol. 14, pp. 685–695, 1997.

[19] V. Lefort, R. Desper, and O. Gascuel, "FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program," *Mol. Biol. Evol.*, vol. 32, no. 10, pp. 2798–2800, 2015.

[20] C. Vaz, M. Nascimento, J. A. Carriço, T. Rocher, and A. P. Francisco, "Distance-based phylogenetic inference from typing data: a unifying view," *Brief. Bioinform.*, vol. 22, no. 3, p. bbaa147, May 2021.

[21] A. Kozlov, D. Darriba, T. Flouri, B. Morel, and A. Stamatakis, "RAxML-NG: A fast, scalable, and user-friendly tool for maximum likelihood phylogenetic inference," *Bioinformatics*, vol. 35, May 2019.

[22] B. Q. Minh *et al.*, "IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era," *Mol. Biol. Evol.*, vol. 37, no. 5, pp. 1530–1534, 2020.

[23] C. Piñeiro, J. M. Abuín, and J. C. Pichel, "Very Fast Tree: speeding up the estimation of phylogenies for large alignments through parallelization and vectorization strategies," *Bioinformatics*, vol. 36, no. 17, pp. 4658–4659, Nov. 2020.

[24] J. L. Thorne, "Models of protein sequence evolution and their applications," *Curr. Opin. Genet. Dev.*, vol. 10, pp. 602–605, 2000.

[25] L. T. Nguyen, H. A. Schmidt, A. Von Haeseler, and B. Q. Minh, "IQ-TREE: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies," *Mol. Biol. Evol.*, vol. 32, no. 1, pp. 268–274, 2015.

[26] S. Q. S. Le, C. C. C. Dang, and O. Gascuel, "Modeling protein evolution with several amino acid replacement matrices depending on site rates," *Mol Biol Evol*, vol. 29, pp. 2921–36, 2012.

[27] Q. LS, G. O, and L. N, "Empirical profile mixture models for phylogenetic reconstruction," *Bioinformatics*, vol. 24, pp. 2317–23, 2008.

[28] S. Whelan and N. Goldman, "A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood

approach.," *Mol. Biol. Evol.*, vol. 18, no. 5, pp. 691–699, 2001.

[29] L. S. Vinh and A. Von Haeseler, "IQPNNI: Moving fast through tree space and stopping in time," *Mol. Biol. Evol.*, vol. 21, no. 8, pp. 1565–1571, 2004.

[30] P. B. P. Frandsen, B. Calcott, C. Mayer, and R. Lanfear, "Automatic selection of partitioning schemes for phylogenetic analyses using iterative k-means clustering of site rates," *BMC Evol. Biol.*, vol. 15, no. 1, p. 13, 2015.

[31] D. Kainer and R. Lanfear, "The effects of partitioning on phylogenetic inference," *Mol. Biol. Evol.*, vol. 32, pp. 1611–1627, 2015.

[32] J. Rota, T. Malm, N. Chazot, C. Peña, and N. Wahlberg, "A simple method for data partitioning based on relative evolutionary rates," *PeerJ*, vol. 6, p. e5498, 2018.

[33] D. T. Hoang, O. Chernomor, A. von Haeseler, B. Q. Minh, and S. V. Le, "UFBoot2: Improving the ultrafast bootstrap approximation," *Mol. Biol. Evol.*, vol. 35, no. 2, pp. 518–522, 2017.

[34] Z. Yang, "Among-site rate variation and its impact on phylogenetic analyses," *Trends Ecol. Evol.*, vol. 11, pp. 367–72, 1996.

[35] Z. Yang, "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods," *J. Mol. Evol.*, vol. 39, no. 3, pp. 306–314, 1994.

[36] M. Pagel and A. Meade, "A phylogenetic mixture model for detecting pattern-heterogeneity in gene sequence or character-state data," *Syst. Biol.*, vol. 53, pp. 571–581, 2004.

[37] R. Lanfear, C. B, S. Y. Ho, and S. Guindon, "PartitionFinder: combined selection of partitioning schemes and substitution models for phylogenetic analyses," *Mol. Biol. Evol.*, vol. 29, pp. 1695–1701, 2012.

[38] S. M. Crotty *et al.*, "GHOST: recovering historical signal from heterotachously evolved sequence alignments," *Syst. Biol.*, vol. 69, no. 2, pp. 249–264, 2020.

[39] B. E. R. Rubin, B. M. Jones, B. G. Hunt, and S. D. Kocher, "Rate variation in the evolution of non-coding DNA associated with social evolution in bees," *Phil. Trans. R. Soc. B*, vol. 374, no. 1777, 2019.

[40] M. S. Springer and J. Gatesy, "On the importance of homology in the age of phylogenomics," *Syst. Biodivers.*, vol. 16, no. 3, pp. 210–228, Apr. 2018.

[41] A. Pandey and E. L. Braun, "Phylogenetic Analyses of Sites in Different Protein Structural Environments Result in Distinct Placements of the Metazoan Root," *Biology*, vol. 9, no. 4. 2020.

[42] S. Le, C. Dang, and O. Gascuel, "Modeling protein evolution with several

amino acid replacement matrices depending on site rates," *Mol Biol Evol*, vol. 29, pp. 2921–36, 2012.

[43] R. Lanfear, P. B. Frandsen, A. M. Wright, T. Senfeld, and B. Calcott, "PartitionFinder 2: new methods for selecting partitioned models of evolution formolecular and morphological phylogenetic analyses," *Mol. Biol. Evol.*, vol. 34, no. 3, pp. 772–773, 2016.

[44] M. Brandley, A. Schmitz, and T. W. Reeder, "Partitioned Bayesian analyses, partition choice, and the phylogenetic relationships of scincid lizards," *Syst Biol*, vol. 54, pp. 373–390, 2005.

[45] N. Lartillot and H. Philippe, "A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process," *Mol Biol Evol*, vol. 21, pp. 1095–1109, 2004.

[46] J. Nylander, F. Ronquist, J. P. Huelsenbeck, and J. Nieves-Aldrey, "Bayesian phylogenetic analysis of combined data," *Syst Biol*, vol. 53, pp. 47–67, 2004.

[47] R. Lanfear, B. Calcott, D. Kainer, C. Mayer, and A. Stamatakis, "Selecting optimal partitioning schemes for phylogenomic datasets," *BMC Evol. Biol.*, vol. 14, no. 1, p. 82, 2014.

[48] S. Baca, E. Toussaint, K. Miller, and A. Short, "Molecular phylogeny of the aquatic beetle family Noteridae (Coleoptera: Adephaga) with an emphasis on data partitioning strategies," *Mol. Phylogenet. Evol.*, vol. 107, pp. 282–292, 2017.

[49] S. Baca, E. Toussaint, and K. Miller, "Molecular phylogeny of the aquatic beetle family Noteridae (Coleoptera: Adephaga) with an emphasis on data partitioning strategies," *Mol. Phylogenet. Evol.*, vol. 107, pp. 282–292, 2017.

[50] L. Bofkin and N. Goldman, "Variation in Evolutionary Processes at Different Codon Positions," *Mol. Biol. Evol.*, vol. 24, no. 2, pp. 513–521, Feb. 2007.

[51] I. Mayrose, A. Mitchell, and T. Pupko, "Site-specific evolutionary rate inference: taking phylogenetic uncertainty into account.," *J. Mol. Evol.*, vol. 60, no. 3, pp. 345–353, Mar. 2005.

[52] C. A. Cummins and J. O. McInerney, "A method for inferring the rate of evolution of homologous characters that can potentially improve phylogenetic inference, resolve deep divergence and correct systematic biases," *Syst. Biol.*, vol. 60, pp. 833–844, 2011.

[53] H. Brinkmann and H. Philippe, "Archaea sister group of bacteria? Indications from tree reconstruction artifacts in ancient phylogenies," *Mol. Biol. Evol.*, vol. 16, no. 6, pp. 817–25, 1999.

[54] J. B. Ahrens, J. Rahaman, and J. Siltberg-Liberles, "Large-Scale Analyses of

Site-Specific Evolutionary Rates across Eukaryote Proteomes Reveal Confounding Interactions between Intrinsic Disorder, Secondary Structure, and Functional Domains," *Genes*, vol. 9, no. 11. 2018.

[55] T. F. Hansen, G. H. Bolstad, and M. Tsuboi, "Analyzing Disparity and Rates of Morphological Evolution with Model-Based Phylogenetic Comparative Methods," *Syst. Biol.*, vol. 71, no. 5, pp. 1054–1072, Sep. 2022.

[56] V. S. Le, C. C. Dang, and Q. S. Le, "Improved mitochondrial amino acid substitution models for metazoan evolutionary studies," *BMC Evol. Biol.*, vol. 17, p. 136, 2017.

[57] H. Kishino and M. Hasegawa, "Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in hominoidea," *J. Mol. Evol.*, vol. 29, no. 2, pp. 170–179, 1989.

[58] H. Shimodaira and M. Hasegawa, "Multiple Comparisons of Log-Likelihoods with Applications to Phylogenetic Inference," *Mol. Biol. Evol.*, vol. 16, no. 8, p. 1114, 1999.

[59] H. Shimodaira, "An approximately unbiased test of phylogenetic tree selection," *Syst. Biol.*, vol. 51, no. 3, pp. 492-508., 2002.

[60] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Contr.*, vol. 19, pp. 716–723, 1974.

[61] C. M. Hurvich and C. Tsai, "Regression and time series model selection in small samples," *Biometrika*, vol. 76, pp. 297–307, 1989.

[62] G. Schwarz, "Estimating the dimension of a model," *Ann Stat*, vol. 6, pp. 461–464, 1978.

[63] S. Abadi, D. Azouri, T. Pupko, and I. Mayrose, "Model selection may not be a mandatory step for phylogeny reconstruction.," *Nat. Commun.*, vol. 10, no. 1, p. 934, Feb. 2019.

[64] R. Del Amparo and M. Arenas, "Consequences of Substitution Model Selection on Protein Ancestral Sequence Reconstruction," *Mol. Biol. Evol.*, vol. 39, no. 7, p. msac144, Jul. 2022.

[65] R. Del Amparo and M. Arenas, "HIV Protease and Integrase Empirical Substitution Models of Evolution: Protein-Specific Models Outperform Generalist Models.," *Genes (Basel).*, vol. 13, no. 1, Dec. 2021.

[66] G. E. Scolaro and E. L. Braun, "The Structure of Evolutionary Model Space for Proteins across the Tree of Life," *Biology*, vol. 12, no. 2. 2023.

[67] E. L. Hatcher *et al.*, "Virus Variation Resource-improved response to emergent viral outbreaks," *Nucleic Acids Res.*, vol. 45, no. D1, pp. D482–D490, 2017.

[68] Q. Jing and M. Wang, “Dengue epidemiology,” *Glob. Heal. J.*, vol. 3, no. 2, pp. 37–45, 2019.

[69] C. C. Dang, V. S. Le, O. Gascuel, B. Hazes, and Q. S. Le, “FastMG: a simple, fast, and accurate maximum likelihood procedure to estimate amino acid replacement rate matrices from large data sets,” *BMC Bioinformatics*, vol. 15, p. 341, 2014.

[70] S. Kalyaanamoorthy, B. quang Minh, K. W. Thomas, von H. Arndt, and S. J. Lars, “ModelFinder: Fast model selection for accurate phylogenetic estimates,” *Nat. Methods*, vol. 14, pp. 587–589, 2017.

[71] S. Q. Le, O. Gascuel, L. SQ, and G. O, “An improved general amino acid replacement matrix,” *Mol Biol Evol*, vol. 25, no. 7, pp. 1307–20, 2008.

[72] M. Bollati *et al.*, “Structure and functionality in flavivirus NS-proteins: Perspectives for drug design,” *Antiviral Res.*, vol. 87, no. 2, pp. 125–128, 2010.

[73] M. W. Dimmic, J. S. Rest, D. P. Mindell, and R. A. Goldstein, “rtREV: An amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny,” *J. Mol. Evol.*, vol. 55, no. 1, pp. 65–73, 2002.

[74] H. Shimodaira and M. Hasegawa, “CONSEL: for assessing the confidence of phylogenetic tree selection,” *Bioinformatics*, vol. 17, pp. 1246–7, 2001.

[75] J. J. Dziak, D. L. Coffman, S. T. Lanza, R. Li, and L. S. Jermiin, “Sensitivity and specificity of information criteria,” *Brief. Bioinform.*, vol. 21, no. 2, pp. 553–565, 2020.

[76] K. Strimmer and A. von Haeseler, “Likelihood-mapping: a simple method to visualize phylogenetic content of a sequence alignment,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 94, no. 13, pp. 6815–6819, 1997.

[77] V. Tagliacollo and R. Lanfear, “Estimating improved partitioning schemes for UltraConserved Elements (UCEs),” *Mol. Biol. Evol.*, vol. 35, no. 7, pp. 1798–1811, 2018.

[78] B. Chor and T. Tuller, “Maximum likelihood of evolutionary trees: hardness and approximation,” *Bioinformatics*, vol. 21, pp. 97–106, 2005.

[79] J. Ballesteros and P. Sharma, “A critical appraisal of the placement of Xiphosura (Chelicerata) with account of known sources of phylogenetic error,” *Syst. Biol.*, vol. 68, pp. 896–917, 2019.

[80] M. Chen, D. Liang, and P. Zhang, “Selecting question-specific genes to reduce incongruence in phylogenomics: a case study of jawed vertebrate backbone phylogeny,” *Syst. Biol.*, vol. 64, pp. 1104–1120, 2015.

[81] I. Irisarri *et al.*, “Phylotranscriptomic consolidation of the jawed vertebrate timetree,” *Nat. Ecol. Evol.*, vol. 1, pp. 1370–1378, 2017.

[82] J.-H. Ran, T.-T. Shen, M.-M. Wang, and X.-Q. Wang, “Phylogenomics resolves the deep phylogeny of seed plants and indicates partial convergent or homoplastic evolution between Gnetales and angiosperms,” *Proc. R. Soc. B*, vol. 285, no. 1881, 2018.

[83] S. Wu, S. Edwards, and L. Liang, “Genome-scale DNA sequence data and the evolutionary history of placental mammals,” *Data Br.*, vol. 18, pp. 1972–1975, 2018.

[84] R. Katja, J. Mappes, K. Lauri, and N. Wahlberg, “Putting Parasemia in its phylogenetic place: a molecular analysis of the subtribe Arctiina (Lepidoptera): molecular phylogeny of Arctiina,” *Syst. Entomol.*, vol. 41, pp. 844–853, 2016.

[85] P. Matos-Maraví, R. P. Núñez, C. Miller, J. Sourakov, and A. N. Wahlberg, “Causes of endemic radiation in the Caribbean: Evidence from the historical biogeography and diversification of the butterfly genus Calisto (Nymphalidae: Satyrinae: Satyrini),” *BMC Evol. Biol.*, vol. 14, p. 199, 2014.

[86] W. N. Rota J, “Exploration of data partitioning in an eight-gene data set: phylogeny of metalmark moths (Lepidoptera, Choreutidae),” *Zool. Scr.*, vol. 41, pp. 536–546, 2012.

[87] Kodandaramaiah Ullasa Pena, C. Braby, M. Grund, R. Müller, C. N. Sören, and W. Niklas, “Phylogenetics of Coenonymphina (Nymphalidae: Satyrinae) and the problem of rooting rapid radiations,” *Mol. Phylogenet. Evol.*, vol. 54, pp. 386–394, 2009.

[88] M. P. Sihvonen, M. Kaila, L. Brehm, G. Hausmann, and A. S. Hermann, “Comprehensive Molecular Sampling Yields a Robust Phylogeny for Geometrid Moths (Lepidoptera: Geometridae),” *PLoS One*, vol. 6, p. e20356, 2011.

[89] C. Penz, P. Devries, and N. Wahlberg, “Diversification of Morpho butterflies (Lepidoptera, Nymphalidae): a re-evaluation of morphological characters and new insight from DNA sequence data,” *Syst. Entomol.*, vol. 37, pp. 670–685, 2012.

[90] Z. Zahiri, L. Niklas, B. J. D. Schmidt, H. Kitching, I. Mutanen, and M. Wahlberg, “Relationships among the basal lineages of Noctuidae (Lepidoptera, Noctuoidea) based on eight gene regions,” *Zool. Scr.*, vol. 42, pp. 488–507, 2013.

[91] N. Wahlberg, J. B. M. Rota, P. Naomi, and W. Christopher, “Revised systematics and higher classification of pierid butterflies (Lepidoptera: Pieridae) based on molecular data,” *Zool. Scr.*, vol. 43, pp. 641–650, 2014.

[92] R. E. Broughton, R. Betancur-R, C. Li, G. Arratia, and G. Ortí, “Multi-locus phylogenetic analysis reveals the pattern and tempo of bony fish evolution.,”

PLoS Curr., vol. 5, Apr. 2013.

[93] B. C. Faircloth, L. Sorenson, F. Santini, and M. E. Alfaro, "A Phylogenomic Perspective on the Radiation of Ray-Finned Fishes Based upon Targeted Sequencing of Ultraconserved Elements (UCEs)," *PLoS One*, vol. 8, no. 6, pp. 1–7, 2013.

[94] M. G. Branstetter *et al.*, "Phylogenomic Insights into the Evolution of Stinging Wasps and the Origins of Ants and Bees.," *Curr. Biol.*, vol. 27, no. 7, pp. 1019–1025, Apr. 2017.

[95] J. E. McCormack, M. G. Harvey, B. C. Faircloth, N. G. Crawford, T. C. Glenn, and R. T. Brumfield, "A Phylogeny of Birds Based on Over 1,500 Loci Collected by Target Enrichment and High-Throughput Sequencing," *PLoS One*, vol. 8, no. 1, pp. 1–11, 2013.

[96] K. A. Meiklejohn, B. C. Faircloth, T. C. Glenn, R. T. Kimball, and E. L. Braun, "Analysis of a Rapid Evolutionary Radiation Using Ultraconserved Elements: Evidence for a Bias in Some Multispecies Coalescent Methods," *Syst. Biol.*, vol. 65, no. 4, pp. 612–627, 2016.

[97] B. T. Smith, M. G. Harvey, B. C. Faircloth, T. C. Glenn, and R. T. Brumfield, "Target Capture and Massively Parallel Sequencing of Ultraconserved Elements for Comparative Studies at Shallow Evolutionary Time Scales," *Syst. Biol.*, vol. 63, no. 1, pp. 83–95, 2013.

[98] T. Le Kim and V. Le Sy, "mPartition: A Model-based method for partitioning alignments," *J. Mol. Evol.*, vol. 88, no. 8, pp. 641–652, 2020.

[99] G. Ávila-Pérez, A. Nogales, V. Martín, F. Almazán, and L. Martínez-Sobrido, "Reverse Genetic Approaches for the Generation of Recombinant Zika Virus," *Viruses*, vol. 10, no. 11, 2018.