

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ



Huỳnh Tiên Dũng

TÍCH HỢP CÁC PHÉP BIẾN HÌNH CẤU  
TRÚC CÂY CẢI TIẾN SUY LUẬN TIẾN  
HÓA THEO TIÊU CHUẨN PARSIMONY  
TRONG MPBOOT

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY  
Ngành: Công nghệ thông tin

HÀ NỘI - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Huỳnh Tiên Dũng

TÍCH HỢP CÁC PHÉP BIẾN HÌNH CẤU  
TRÚC CÂY CẢI TIẾN SUY LUẬN TIẾN  
HÓA THEO TIÊU CHUẨN PARSIMONY  
TRONG MPBOOT

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY  
Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: TS. Hoàng Thị Diệp

HÀ NỘI - 2024

## LỜI CAM ĐOAN

Em xin cam đoan: Khóa luận tốt nghiệp với đề tài “Tích hợp các phép biến hình cấu trúc cây cải tiến suy luận tiến hóa theo tiêu chuẩn parsimony trong MPBoot” trong báo cáo này là do em thực hiện dưới sự hướng dẫn của Tiến Sĩ Hoàng Thị Diệp. Những gì em viết ra hoàn toàn trung thực, chính xác và không có sự sao chép từ các tài liệu, không sử dụng kết quả của người khác mà không trích dẫn cụ thể. Đây là công trình nghiên cứu cá nhân em tự phát triển, không sao chép mã nguồn của người khác. Nếu vi phạm những điều trên, em xin chấp nhận tất cả những truy cứu về trách nhiệm theo quy định của Trường Đại học Công nghệ - ĐHQGHN.

Hà Nội, ngày 10 tháng 12 năm 2024

Sinh viên

Huỳnh Tiên Dũng

## LỜI CẢM ƠN

Lời đầu tiên cho phép em bày tỏ lòng biết ơn sâu sắc đến Khoa Công nghệ Thông tin - Trường Đại học Công nghệ, ĐHQGHN. Đây là nơi em đã có cơ hội tiếp cận với những tri thức mới mẻ, được học hỏi từ các thầy cô xuất sắc và kết nối với những người bạn, anh chị em đầy năng động và tài năng.

Em cũng xin gửi lời cảm ơn chân thành đến cô Hoàng Thị Diệp, người đã luôn là nguồn cảm hứng và sự hướng dẫn quý báu trong suốt thời gian em học tập tại trường. Sự tận tâm và hỗ trợ nhiệt tình của cô đã tiếp thêm động lực để em vượt qua những thử thách trong hành trình nghiên cứu và hoàn thiện khóa luận tốt nghiệp.

Ngoài ra, em xin gửi lời cảm ơn đến gia đình, bạn bè và những người đã luôn giúp đỡ, động viên, đồng hành cùng em suốt chặng đường học tập ở trường và khoảng thời gian thực hiện khóa luận.

Kính chúc tất cả mọi người luôn vui vẻ, hạnh phúc và gặt hái được nhiều thành công trong cuộc sống.

## TÓM TẮT

**Tóm tắt:** Bài toán xây dựng cây bootstrap tiến hóa (Phylogenetic bootstrapping) là một phần quan trọng trong sinh học tiến hóa, nhằm tái tạo cây tiến hóa với số lượng thay đổi tối thiểu dựa trên tiêu chí tính tiết kiệm tối đa (maximum parsimony - MP) đồng thời tính độ tin cậy của các phân hoạch nhị phân trong cây này. MPBoot2 cải tiến so với phiên bản trước, MPBoot, bằng cách tích hợp kỹ thuật biến đổi cây Tree Bisection and Reconnection (TBR), cho phép khám phá không gian cây một cách toàn diện hơn so với Subtree Pruning and Regrafting (SPR). Hai phép biến đổi này bổ trợ lẫn nhau để tăng cường khả năng khám phá không gian cây. Để nâng cao hiệu suất hơn nữa, chúng tôi giới thiệu MPBoot-RL, áp dụng giải thuật tối ưu đàm kiến (ant colony optimization) nhằm kết hợp động giữa SPR và TBR, cải thiện cả độ chính xác và thời gian chạy. MPBoot2 còn bao gồm các tính năng tiên tiến như hệ thống checkpoint, hỗ trợ nhiều loại dữ liệu khác nhau, và quản lý bộ nhớ tối ưu, giúp nó phù hợp với các tập dữ liệu lớn và phức tạp. Kết quả thực nghiệm cho thấy hiệu suất vượt trội về cả độ chính xác lẫn tốc độ thực thi, khẳng định MPBoot2 và MPBoot-RL là những công cụ đa năng và mạnh mẽ cho phân tích phát sinh chủng loại dựa trên tiêu chí MP.

**Từ khóa:** MPBoot, MPBoot2, MPBoot-RL, Phát sinh chủng loại học

# MỤC LỤC

Lời cam đoan .....	i
Lời cảm ơn .....	ii
Tóm tắt .....	iii
Mục lục .....	iv
Danh mục hình ảnh .....	vii
Danh mục bảng .....	x
Danh mục giải thuật .....	xi
<b>Chương 1. Giới thiệu .....</b>	<b>1</b>
1.1. Tổng quan .....	1
1.2. Mục tiêu của đề tài .....	4
1.3. Cấu trúc của khóa luận .....	4
<b>Chương 2. Các khái niệm cơ sở .....</b>	<b>5</b>
2.1. Bài toán xây dựng cây bootstrap tiến hóa .....	5
2.1.1. Mô hình bài toán .....	5
2.1.2. Tiêu chuẩn đánh giá cây tiến hóa .....	7
2.1.3. Các kỹ thuật biến đổi cây thông dụng .....	9
2.1.4. Các công trình liên quan .....	12
2.2. Giải thuật tối ưu đòn kiến .....	14
2.2.1. Tổng quan .....	15
2.2.2. Cấu trúc thuật toán .....	15
2.2.3. Các biến thể của giải thuật tối ưu đòn kiến .....	17
<b>Chương 3. Tích hợp phép biến đổi TBR vào MPBoot .....</b>	<b>19</b>
3.1. Khung thuật toán MPBoot .....	19

3.2. Việc áp dụng phép biến đổi cây trong thuật toán MPBoot gốc .....	20
3.3. Đề xuất tính toán nhanh một phép biến đổi TBR .....	20
3.4. Đề xuất tìm kiếm cây lân cận sử dụng TBR .....	22
3.4.1. Chiến lược tìm kiếm “tốt nhất” .....	22
3.4.2. Chiến lược tìm kiếm “tốt hơn” .....	23
3.5. Đề xuất thuật toán leo đồi TBR .....	24
3.6. Đề xuất thuật toán MPBoot-TBR .....	25
<b>Chương 4. Đề xuất phương pháp MPBoot2 .....</b>	<b>27</b>
4.1. Các thay đổi so với MPBoot .....	27
4.2. Đánh giá thực nghiệm trên các bộ dữ liệu vừa và nhỏ .....	28
4.2.1. Dữ liệu .....	28
4.2.2. Cài đặt thực nghiệm .....	29
4.2.3. Tiêu chí đánh giá .....	29
4.2.4. Kết quả .....	30
4.2.5. Phân tích về hiệu năng của TBR5 so với SPR6 .....	36
4.3. Đánh giá thực nghiệm trên các bộ dữ liệu lớn .....	38
<b>Chương 5. Đề xuất phương pháp MPBoot-RL .....</b>	<b>41</b>
5.1. Áp dụng ACO vào MPBoot2 .....	41
5.1.1. Cấu trúc đồ thị .....	42
5.1.2. Thông tin heuristic .....	43
5.1.3. Vết mùi pheromone và các quy tắc cập nhật .....	43
5.1.4. Quy trình bước đi ngẫu nhiên để xây dựng giải pháp .....	44
5.1.5. Cài đặt .....	45
5.2. Đánh giá thực nghiệm .....	45

5.2.1. Cài đặt thực nghiệm .....	45
5.2.2. Kết quả .....	46
5.2.3. Khảo sát giữa các phép biến đổi cây với độ khó của dữ liệu ...	49
<b>Chương 6. Kết luận .....</b>	<b>52</b>
6.1. Kết quả đạt được .....	52
6.2. Các định hướng phát triển .....	53
<b>Công bố liên quan .....</b>	<b>54</b>
<b>Tài liệu tham khảo .....</b>	<b>55</b>
<b>Phụ lục .....</b>	<b>58</b>
A. Kết quả phân tích bổ sung của các phiên bản MPBoot2 .....	58
B. Kết quả phân tích bổ sung của các phiên bản MPBoot-RL .....	65
C. Kết quả bổ sung của các phiên bản MPBoot-RL với các bộ siêu tham số khác nhau .....	74

## DANH MỤC HÌNH ẢNH

Hình 1.1: Hình minh họa cây tiến hóa trong quyển “The Origin of Species” của Charles Darwin .....	2
Hình 1.2: Cây tiến hóa mô tả các biến thể SARS-CoV-2 .....	3
Hình 2.1: Ví dụ sắp hàng đa chuỗi (MSA) và cây tiến hóa đơn giản .....	5
Hình 2.2: Ví dụ minh họa tổng hợp kết quả bootstrap .....	6
Hình 2.3: Ví dụ đơn giản về tiêu chí MP .....	9
Hình 2.4: Minh họa các phép biến đổi cây .....	10
Hình 2.5: Minh họa phép biến đổi cây NNI .....	11
Hình 2.6: Minh họa phép biến đổi cây SPR .....	11
Hình 2.7: Minh họa phép biến đổi cây TBR .....	12
Hình 2.8: Phương pháp xấp xỉ bootstrap .....	13
Hình 2.9: Minh họa sử dụng kĩ thuật REPS để tính điểm MP của sắp hàng bootstrap .....	14
Hình 3.1: Một vòng lặp trong pha khám phá (pha 2) của khung thuật toán MPBoot .....	19
Hình 3.2: Một phép biến đổi TBR với cạnh cắt $R$ và hai cạnh nối $I_1$ và $I_2$ ...	21
Hình 3.3: Nhận diện những đỉnh cần phải tính lại điểm với cạnh cắt $R$ và hai cạnh nối $I_1$ và $I_2$ .....	21
Hình 4.1: Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu DNA và protein từ TreeBASE. Các biểu đồ cột thể hiện số lượng bộ dữ liệu (trong tổng số 115 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát. ....	31
Hình 4.2: So sánh TBR5-SC100 và TNT trên chi phí đồng nhất ( <b>a</b> , <b>b</b> ) và chi phí không đồng nhất ( <b>c</b> , <b>d</b> ) trên các MSAs DNA và protein từ TreeBASE. ....	34
Hình 4.3: Độ chính xác bootstrap của các phương pháp TBR5 (đường màu cam), TBR5-BETTER (đường màu xanh) và SPR6 (đường màu đen - phiên bản MPBoot cũ) .....	35

Hình 4.4: Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu morphology từ TreeBASE. Các biểu đồ cột thể hiện tỉ lệ số bộ dữ liệu (trong tổng số 30 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát .....	35
Hình 4.5: So sánh kết quả của TBR5 với SPR6 trên 115 bộ dữ liệu TreeBASE	38
Hình 5.1: Minh họa cho khung thuật toán MPBoot-RL .....	42
Hình 5.2: Cấu trúc đồ thị ACO .....	42
Hình 5.3: Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu DNA và protein từ TreeBASE. Các biểu đồ cột thể hiện số lượng bộ dữ liệu (trong tổng số 115 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát .....	47
Hình 5.4: Độ chính xác bootstrap của các phương pháp ACO-MUL (đường màu cam), ACO-ONCE (đường màu xanh) và SPR6 (đường màu đen - của phiên bản MPBoot) .....	49
Hình 5.5: Khảo sát liên quan giữa ba phép biến đổi cây với độ khó Pythia của ACO-MUL và ACO-ONCE (đã làm mượt với window size = 30) .....	50
Hình A.1: So sánh kết quả của TBR5-SC100 với SPR6 trên 115 bộ dữ liệu TreeBASE .....	58
Hình A.2: So sánh kết quả của TBR5-BETTER với SPR6 trên 115 bộ dữ liệu TreeBASE .....	59
Hình A.3: So sánh kết quả của TBR6 với SPR6 trên 115 bộ dữ liệu TreeBASE	60
Hình A.4: So sánh kết quả của SPR6 với TNT trên 115 bộ dữ liệu TreeBASE	61
Hình A.5: So sánh kết quả của TBR5 với TNT trên 115 bộ dữ liệu TreeBASE	62
Hình A.6: So sánh kết quả của TBR5-BETTER với TNT trên 115 bộ dữ liệu TreeBASE .....	63
Hình A.7: So sánh kết quả của TBR6 với TNT trên 115 bộ dữ liệu TreeBASE	64
Hình B.8: So sánh kết quả của ACO-MUL với SPR6 trên 115 bộ dữ liệu TreeBASE .....	65
Hình B.9: So sánh kết quả của ACO-MUL với TBR5 trên 115 bộ dữ liệu TreeBASE .....	66

Hình B.10: So sánh kết quả của ACO-MUL với TBR5-BETTER trên 115 bộ dữ liệu TreeBASE .....	67
Hình B.11: So sánh kết quả của ACO-MUL với TNT trên 115 bộ dữ liệu TreeBASE .....	68
Hình B.12: So sánh kết quả của ACO-ONCE với SPR6 trên 115 bộ dữ liệu TreeBASE .....	69
Hình B.13: So sánh kết quả của ACO-ONCE với TBR5 trên 115 bộ dữ liệu TreeBASE .....	70
Hình B.14: So sánh kết quả của ACO-ONCE với TBR5-BETTER trên 115 bộ dữ liệu TreeBASE .....	71
Hình B.15: So sánh kết quả của ACO-ONCE với TNT trên 115 bộ dữ liệu TreeBASE .....	72
Hình B.16: So sánh kết quả của ACO-MUL với ACO-ONCE trên 115 bộ dữ liệu TreeBASE .....	73

## **DANH MỤC BẢNG**

Bảng 4.1: Thời gian chạy tổng cộng (giờ) và tỷ lệ thời gian (so với SPR6) của các phương pháp trên 115 bộ dữ liệu từ bộ dữ liệu TreeBASE .....	32
Bảng 4.2: Thời gian chạy tổng cộng (giờ) của các phương pháp trên 30 bộ dữ liệu morphology .....	36
Bảng 4.3: Phân tích các bộ dữ liệu mà SPR6 hoặc TBR5 vươn lên so với phương pháp còn lại .....	37
Bảng 4.4: Điểm MP và thời gian chạy (giờ) của các phương pháp trên các bộ dữ liệu lớn với điều kiện chi phí đồng nhất .....	39
Bảng 4.5: Điểm MP và thời gian chạy (giờ) của các phương pháp trên các bộ dữ liệu lớn với điều kiện chi phí không đồng nhất .....	40
Bảng 5.1: Thời gian chạy tổng cộng (giờ) và tỷ lệ thời gian (so với SPR6) của các phương pháp trên 115 bộ dữ liệu từ bộ dữ liệu TreeBASE .....	48
Bảng C.1: Thống kê số lượng bộ dữ liệu đạt được kết quả tốt nhất (trong 115 bộ TreeBASE) của các phương pháp MPBoot-RL với các bộ siêu tham số khác nhau .....	74

## **DANH MỤC GIẢI THUẬT**

Thuật toán 3.1: Thực hiện phép biến đổi TBR với cạnh cắt $R$ trên cây $T$ ....	23
Thuật toán 3.2: Chiến thuật tìm kiếm “tốt hơn” sử dụng TBR .....	24
Thuật toán 3.3: Thuật toán leo đồi sử dụng TBR trên cây $T$ .....	24
Thuật toán 3.4: Thuật toán MPBoot-TBR trong xấp xỉ bootstrap .....	25

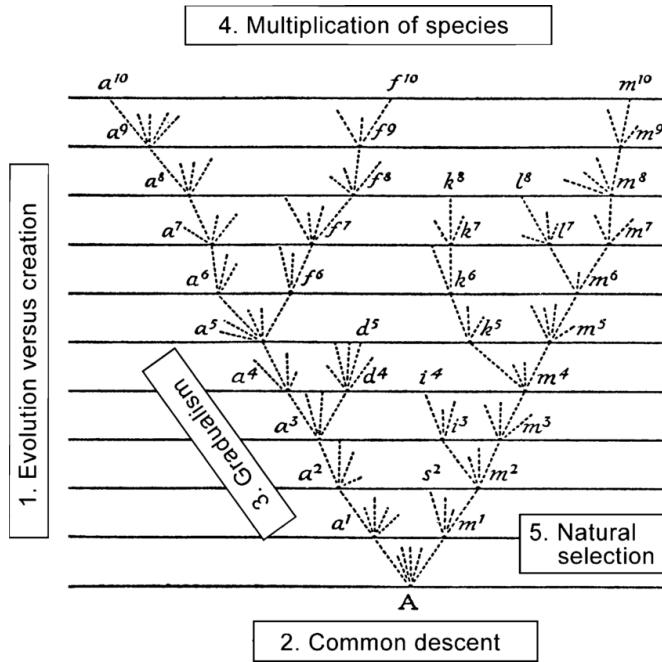
# Chương 1

## Giới thiệu

### 1.1. Tổng quan

Tin sinh học (Bioinformatics) là một lĩnh vực liên ngành kết hợp giữa sinh học, công nghệ thông tin và toán học nhằm thu thập, phân tích và giải thích dữ liệu sinh học quy mô lớn, như DNA, RNA, protein và bộ gen. Mục tiêu của tin sinh học là hiểu và mô hình hóa các quá trình sinh học phức tạp, phát triển các thuật toán và công cụ để phân tích dữ liệu, từ đó khám phá kiến thức mới về gen, protein, hoặc các vùng chức năng quan trọng, đồng thời hỗ trợ nghiên cứu y học và phát triển các liệu pháp điều trị bệnh. Lĩnh vực này được ứng dụng rộng rãi trong y học và dược phẩm, bao gồm phát triển thuốc, y học cá nhân hóa, nghiên cứu ung thư; trong hệ gen học (genomics) với việc giải mã bộ gen, phân tích biến thể di truyền; và trong hệ protein học (proteomics) với việc dự đoán cấu trúc và chức năng protein. Ngoài ra, tin sinh học còn hỗ trợ phân tích mạng lưới sinh học, nghiên cứu tiến hóa, và xử lý dữ liệu sinh học lớn, đóng vai trò quan trọng trong việc phát triển khoa học sự sống và công nghệ sinh học hiện đại.

Cây tiến hóa (phylogenetic tree) là một biểu diễn đồ họa dùng để mô tả mối quan hệ tiến hóa giữa các loài sinh vật dựa trên nguồn gốc chung của chúng. Cây này minh họa cách các loài hoặc nhóm sinh vật phân nhánh từ một tổ tiên chung qua thời gian, phản ánh quá trình tiến hóa thông qua sự thay đổi di truyền và chọn lọc tự nhiên. Ý tưởng về cây tiến hóa lần đầu tiên được Charles Darwin giới thiệu trong tác phẩm nổi tiếng The Origin of Species (1859) [1], nơi ông sử dụng hình minh họa duy nhất trong cuốn sách để thể hiện khái niệm về sự phân nhánh và đa dạng hóa các loài trong tự nhiên (xem [Hình 1.1](#)).

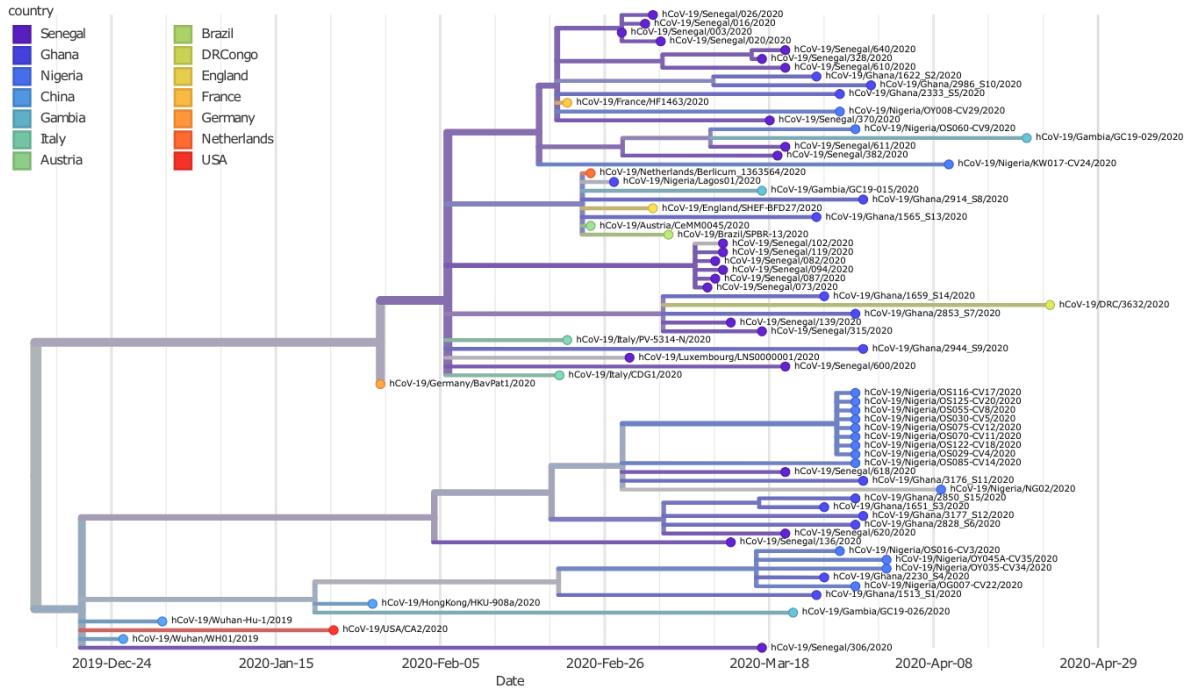


**Hình 1.1** — Hình minh họa cây tiến hóa trong quyển “The Origin of Species” của Charles Darwin

Cây tiến hóa thường được mô tả dưới dạng cây phả hệ - một biểu đồ phân nhánh biểu thị mối quan hệ tiến hóa giữa các loài hoặc nhóm sinh vật, với mỗi đỉnh (node) và mỗi nhánh (branch) mang ý nghĩa riêng biệt. Các đỉnh trong cây đại diện cho các đơn vị sinh học, chẳng hạn như loài, quần thể, hoặc tổ tiên chung, trong đó các đỉnh lá (leaf nodes) thường biểu thị các loài hiện tại hoặc nhóm sinh vật được nghiên cứu, còn các đỉnh bên trong (internal nodes) thể hiện tổ tiên chung của các nhóm con cháu. Các nhánh trong cây kết nối các đỉnh, mô tả quá trình tiến hóa từ tổ tiên đến con cháu.

Phân tích cây tiến hóa đã chứng tỏ vai trò thiết yếu trong phân loại học [2,3], hay gần đây nhất là việc nghiên cứu và kiểm soát đại dịch SARS-CoV-2 (Covid 19), khi nó giúp các nhà khoa học theo dõi sự tiến hóa và lây lan của virus trên toàn cầu [4,5]. Bằng cách xây dựng cây tiến hóa dựa trên dữ liệu bộ gen của hàng triệu mẫu virus được thu thập từ các bệnh nhân, các nhà nghiên cứu có thể xác định được mối quan hệ giữa các biến thể của SARS-CoV-2, từ đó theo dõi nguồn gốc và sự xuất hiện của các biến thể mới như Alpha, Delta, và Omicron. Cây tiến hóa cung cấp thông tin quan trọng để hiểu cách các đột biến di truyền ảnh hưởng đến khả năng lây nhiễm, đặc tính của virus, và từ đó tối ưu hiệu quả của vaccine. Ngoài ra, cây tiến hóa cũng hỗ trợ việc phát triển các chiến lược y tế công cộng, như giám sát dịch tễ học và dự báo sự bùng phát dịch

bệnh, nhằm ứng phó kịp thời và hiệu quả với các biến thể nguy hiểm. Phân tích này không chỉ minh chứng cho sức mạnh của khoa học dữ liệu trong y học hiện đại mà còn mở ra hướng đi mới trong việc kiểm soát các dịch bệnh tương lai.



**Hình 1.2 — Cây tiến hóa mô tả các biến thể SARS-CoV-2 [6]**

Tiêu chí Maximum Parsimony (MP) được sử dụng để xây dựng cây tiến hóa bằng cách tối thiểu hóa chi phí thay thế cần thiết để giải thích các chuỗi trong một sắp hàng đa chuỗi (Multiple Sequence Alignment - MSA). Điểm MP của một cây được tính bằng cách cộng tổng điểm của từng vị trí (cột) trong MSA.

Bài toán xây dựng cây bootstrap tiến hóa (Phylogenetic Bootstrapping) (nguồn gốc từ phương pháp bootstrap trong thống kê [7]) liên quan đến việc tạo ra nhiều bản sao bootstrap bằng cách lấy mẫu một số cột (có thể trùng nhau) của MSA gốc. Mục tiêu là xác định cây tốt nhất cho cả MSA gốc và mỗi bản sao bootstrap, theo tiêu chí MP. Kết quả được tóm tắt dưới dạng tần suất phân chia nhánh, được sử dụng để đánh giá giá trị hỗ trợ cho mỗi phân nhánh của cây. Xây dựng cây bootstrap tiến hóa theo tiêu chuẩn Maximum Parsimony là bài toán tối ưu tổ hợp thuộc lớp NP-complete [8].

MPBoot [9] là một công cụ phần mềm được thiết kế để thực hiện phân tích bootstrap MP một cách hiệu quả. MPBoot sử dụng các phép thao tác biến đổi cây như Nearest Neighbor Interchange (NNI) và Subtree Pruning and Regrafting

(SPR) để tìm kiếm cây tốt nhất. Tuy nhiên, MPBoot hiện tại chủ yếu dựa vào SPR. Điều này có thể hạn chế hiệu suất trong việc khám phá không gian cây.

## 1.2. Mục tiêu của đề tài

Xây dựng cây bootstrap tiến hóa theo tiêu chí maximum parsimony là một bài toán NP-complete [8], khiến thuật toán leo đồi trở thành lựa chọn phổ biến. MPBoot hiện sử dụng leo đồi để cải thiện lời giải thông qua các kỹ thuật biến đổi cây như Nearest Neighbor Interchange (NNI) và Subtree Pruning and Regrafting (SPR). Tuy nhiên, phép biến đổi Tree Bisection and Reconnection (TBR), vốn mạnh mẽ hơn, vẫn chưa được tích hợp vào MPBoot. Mục tiêu của khóa luận này là tích hợp TBR vào MPBoot, đồng thời phát triển nhiều tính năng mới như checkpoint, hỗ trợ dữ liệu khác như morphology, binary, dữ liệu lớn, và cải tiến các tính năng hiện có để phát triển phiên bản MPBoot2. Để khai thác tối đa sức mạnh của các phép biến đổi, khóa luận còn phát triển MPBoot-RL, sử dụng giải thuật đàn kiến (Ant Colony Optimization) để kết hợp linh hoạt các phép biến đổi, nhằm tối ưu hóa cả điểm số và thời gian tính toán. Ngoài ra, khóa luận cũng thực hiện khảo sát liên quan giữa các thông số sử dụng các phép biến đổi cây với “độ khó” của tập dữ liệu.

## 1.3. Cấu trúc của khóa luận

Phần còn lại của khóa luận này được trình bày như sau:

- [Chương 2](#): Giới thiệu các khái niệm cơ sở về bài toán xây dựng cây bootstrap tiến hóa và giải thuật đòn kiến.
- [Chương 3](#): Trình bày phương pháp tích hợp kĩ thuật biến đổi cây TBR vào MPBoot.
- [Chương 4](#): Trình bày phiên bản MPBoot2.
- [Chương 5](#): Trình bày phiên bản MPBoot-RL.
- [Chương 6](#): Kết luận về các thuật toán đề xuất và kết quả thực nghiệm, đồng thời chỉ ra các định hướng cải tiến trong tương lai.
- [Phụ lục](#): Trình bày phụ lục của khóa luận.

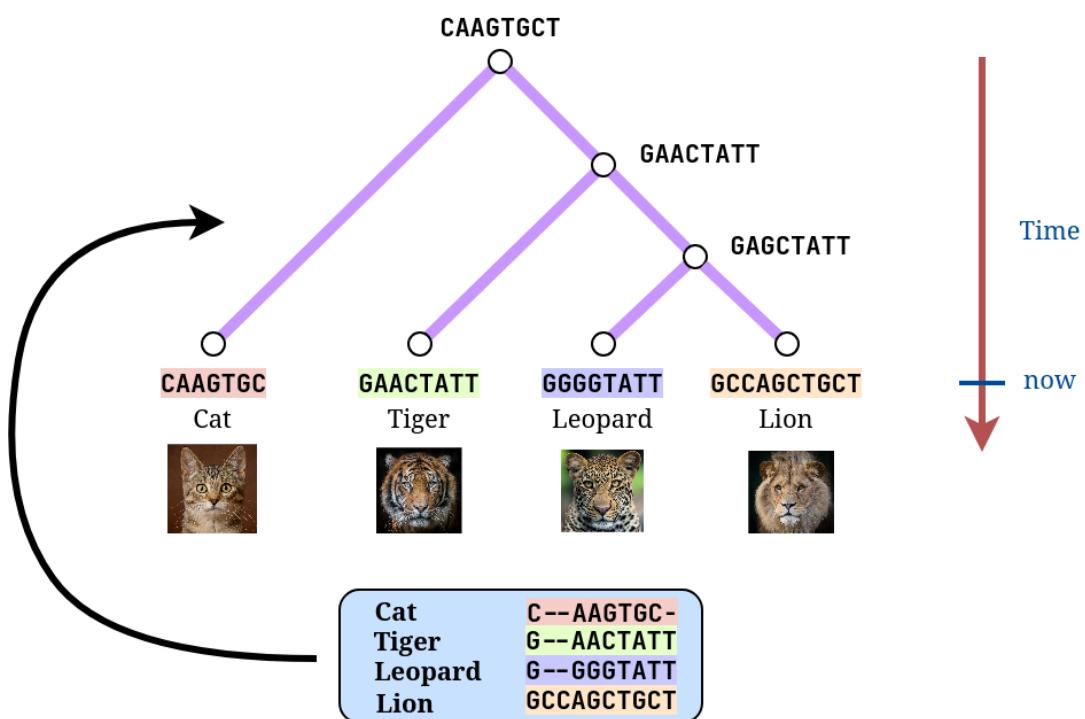
# Chương 2

## Các khái niệm cơ sở

### 2.1. Bài toán xây dựng cây bootstrap tiến hóa

#### 2.1.1. Mô hình bài toán

Bài toán xây dựng cây tiến hóa bootstrap nhận vào một ma trận  $A^{\text{data}}$  kích thước  $n \times m$ , đại diện cho  $n$  chuỗi, mỗi chuỗi có  $m$  ký tự. Mỗi chuỗi trong số  $n$  chuỗi này tương ứng với chuỗi sinh học của một loài đang được nghiên cứu. Các đặc điểm sinh học rất đa dạng và có thể bao gồm DNA, protein, morphology hoặc dữ liệu nhị phân. Ngoài ra, một số  $B$  (thông thường  $B = 1000$ ) được cung cấp, đại diện cho số lượng các mẫu bootstrap. Mỗi cây tiến hóa ứng viên  $T$  cho  $A^{\text{data}}$  là một cây nhị phân có gốc với  $n$  lá, trong đó mỗi lá tương ứng với một chuỗi (xem [Hình 2.1](#)).

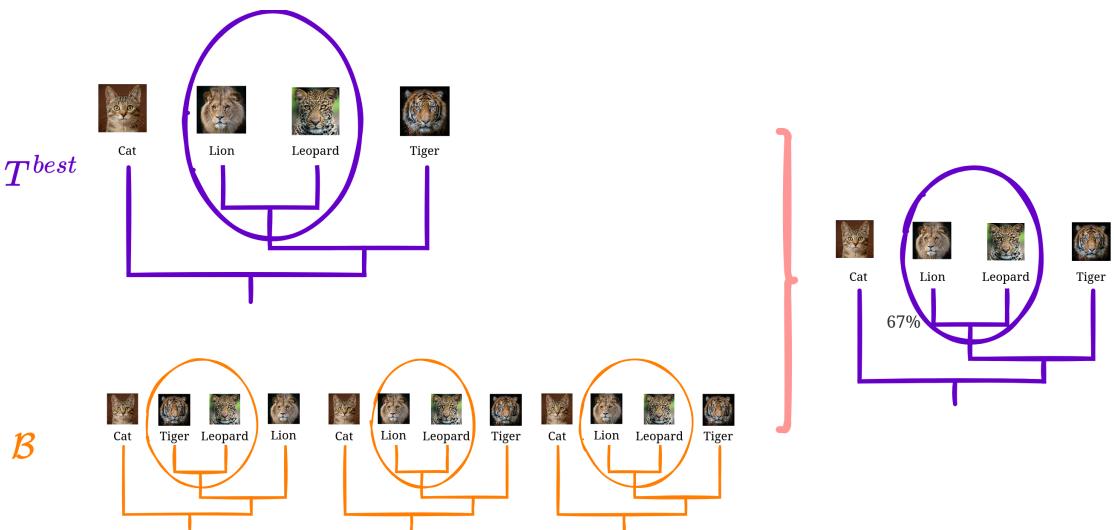


**Hình 2.1** — Ví dụ sắp hàng đa chuỗi (MSA) và cây tiến hóa đơn giản

Kết quả đầu ra của bài toán là cây tốt nhất  $T^{\text{best}}$ , giải thích tốt nhất sắp hàng  $A^{\text{data}}$ , và một tập hợp  $\mathcal{B}$  các cây tốt nhất  $T_b^{\text{best}}$  cho các mẫu bootstrap  $A_b$  (với  $b = 1, \dots, B$ ). Mỗi sắp hàng bootstrap  $A_b$  là một mẫu bootstrap của sắp hàng ban đầu  $A^{\text{data}}$ , được tạo ra với cùng kích thước bằng cách lấy mẫu cột (cho phép lặp lại) đúng  $m$  lần từ  $A^{\text{data}}$ . Chất lượng của một cây được đánh giá dựa trên một tiêu chí xác định trước.

Tập hợp các cây bootstrap  $\mathcal{B}$  thường được tóm tắt dưới dạng một vectơ tần suất của các phân hoạch nhị phân được ánh xạ lên  $T^{\text{best}}$  (xem [Hình 2.2](#)) hoặc dưới dạng một cây đồng thuận. Hai phương pháp tóm tắt phổ biến là bootstrap chuẩn và bootstrap xấp xỉ. Phương pháp bootstrap chuẩn [7,10] xây dựng các cây trong tập  $\mathcal{B}$  một cách độc lập bằng cách thực hiện các lần duyệt không gian tìm kiếm riêng biệt cho từng mẫu bootstrap. Tần suất được gán cho mỗi phân hoạch nhị phân trên  $T^{\text{best}}$  được gọi là giá trị hỗ trợ bootstrap cho nhánh đó.

Do mỗi cây bootstrap được tối ưu bằng cách thực hiện một lần tìm kiếm độc lập trên mỗi sắp hàng bootstrap, phương pháp này tiêu tốn khá nhiều chi phí tính toán. MPBoot tối ưu bằng phương pháp bootstrap xấp xỉ, chỉ thực hiện một lần duyệt không gian tìm kiếm duy nhất, và tập hợp các cây  $\mathcal{B}$  được chọn từ các cây mà thuật toán đã đánh giá trong lần tìm kiếm này.



**Hình 2.2** — Ví dụ minh họa tổng hợp kết quả bootstrap

Các nghiên cứu về việc xây dựng cây tiền hóa thường dựa vào hai giả định quan trọng sau đây [11]:

- Sự tiền hóa giữa các cột trong dữ liệu (trên một cây nhất định) là độc lập.

- Quá trình tiến hóa là độc lập giữa các phân hoạch nhị phân trên cây.

### 2.1.2. Tiêu chuẩn đánh giá cây tiến hóa

Trong bài toán xây dựng cây tiến hóa, ba tiêu chuẩn thường được sử dụng để đánh giá một cây tiến hóa là:

- Tiếp cận Bayesian
- Tiêu chuẩn Maximum Likelihood
- Tiêu chuẩn Maximum Parsimony

#### 2.1.2.1. Tiếp cận Bayesian

Tiếp cận Bayes sử dụng suy diễn Bayes để đánh giá một cấu trúc cây. Với một vectơ độ dài nhánh  $l$  và mô hình thay thế  $\theta$ , công thức (2.1) sau đánh giá mức độ phù hợp của cấu trúc cây  $T$  với sắp xếp  $A^{\text{data}}$ :

$$P(T, l, \theta | A^{\text{data}}) = \frac{P(A^{\text{data}} | T, l, \theta) \cdot P(T, l, \theta)}{P(D)} \quad (2.1)$$

Trong đó:

- $P(A^{\text{data}} | T, l, \theta)$  là xác suất hợp lý của  $A^{\text{data}}$  cho cây  $T$ .
- $P(T, l, \theta)$  là xác suất tiên nghiệm.
- $P(D)$  là xác suất biên.

Xác suất hợp lý  $P(A^{\text{data}} | T, l, \theta)$  được tính bằng thuật toán cắt tia Felsenstein. Do không gian tìm kiếm rất lớn, tiếp cận Bayes thường được sử dụng cùng các thuật toán Markov Chain Monte Carlo (MCMC) để xấp xỉ.

#### 2.1.2.2. Tiêu chuẩn Maximum Likelihood

Ước lượng hợp lý cực đại (Maximum Likelihood Estimation - MLE) là phương pháp ước lượng tham số tối ưu để tối đa hóa xác suất quan sát dữ liệu thực tế dưới một mô hình. Trong xây dựng cây tiến hóa loài, MLE dùng hàm likelihood để tính xác suất dữ liệu trình tự sinh học đọc theo cấu trúc cây và độ dài nhánh.

Hàm likelihood  $L(T, l, \theta)$  cho sắp xếp dữ liệu  $A^{\text{data}}$ , cấu trúc cây  $T$ , độ dài nhánh  $l$ , và mô hình thay thế  $\theta$  được định nghĩa theo công thức (2.2) (dựa trên giả định về sự tiến hóa của các cột  $A_i^{\text{data}}$  là độc lập):

$$L(A^{\text{data}} \mid T, l, \theta) = \prod_{i=1}^n P(A_i^{\text{data}} \mid T, l, \theta) \quad (2.2)$$

Trong đó:

- $A_i^{\text{data}}$  là cột thứ  $i$  của sáp xếp dữ liệu  $A^{\text{data}}$ ,
- $P(A_i^{\text{data}} \mid T, l, \theta)$  là xác suất hợp lý cho cột thứ  $i$  của sáp xếp dữ liệu.

Xác suất  $P(A_i^{\text{data}} \mid T, l, \theta)$  được tính qua thuật toán cắt tia Felsenstein, là tổng hợp các xác suất chuyển trạng thái đọc theo các nhánh của cây, tính theo công thức (2.3):

$$P(A_i^{\text{data}} \mid T, l, \theta) = \sum_x \pi_x L_u(x) = \sum_x \pi_x \prod_v \left( \sum_y L_v(y) \cdot p_{xy}(l_v) \right) \quad (2.3)$$

Trong đó:

- $\pi_x$  là tần số ổn định của ký tự  $x$ ,
- $L_v(y)$  là xác suất tại nút  $v$  giả định ký tự  $y$  tại nút đó,
- $p_{xy}(l_v)$  là xác suất chuyển từ ký tự  $x$  sang  $y$  đọc theo nhánh có độ dài  $l_v$ .

Phương pháp này giúp tìm ra các tham số cây tiến hóa sao cho xác suất đúng với quan sát dữ liệu là cao nhất.

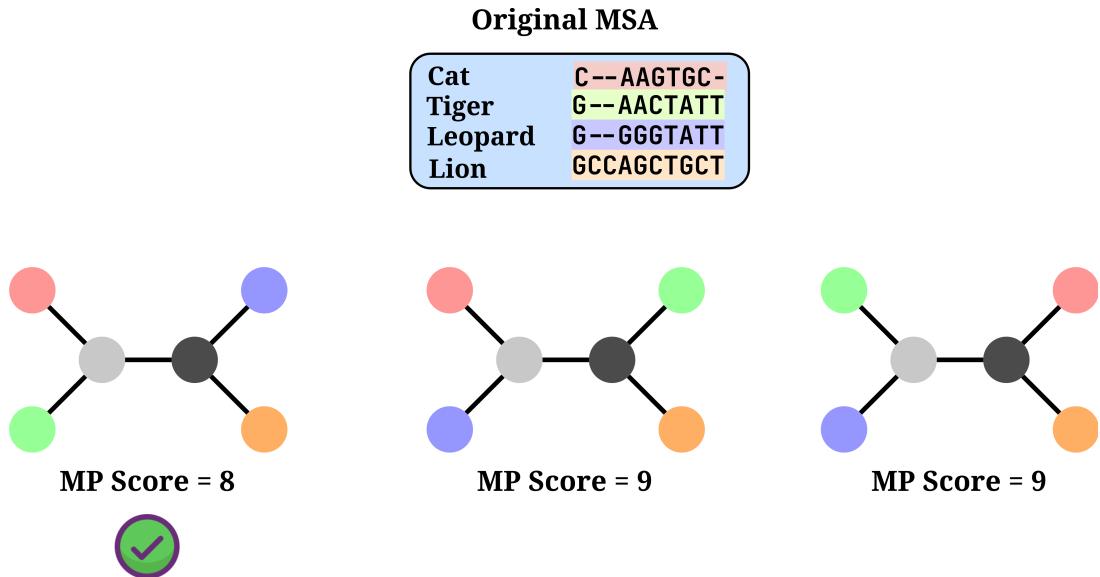
### 2.1.2.3. Tiêu chuẩn Maximum Parsimony

Tiêu chí Tiết kiệm Tối đa (Maximum Parsimony - MP) sử dụng điểm MP để đánh giá mức độ phù hợp của một cấu trúc cây  $T$  trong việc giải thích sáp xếp trình tự  $A^{\text{data}}$ . Điểm MP của  $T$  là chi phí tối thiểu của các thay đổi ký tự cần thiết để giải thích các trình tự quan sát tại các nút lá, dựa trên tổ tiên chung gần nhất của chúng. Điểm MP cho toàn bộ sáp hàng được tính theo công thức (2.4):

$$\text{MP}(T \mid A^{\text{data}}) = \sum_{i=1}^m \text{MP}(T \mid D_i) \quad (2.4)$$

Ở đây,  $\text{MP}(T \mid D_i)$  là điểm MP của cây  $T$  cho cột  $D_i$ . Điểm MP cho một cột được tính bằng thuật toán Fitch [12] (cho các thay đổi có chi phí đồng nhất) hoặc thuật toán Sankoff [13] (cho các thay đổi có chi phí không đồng nhất).

Mục tiêu khi xây dựng cây tiến hóa là tìm cây có điểm MP nhỏ nhất mô tả  $A^{\text{data}}$ , gọi là cây MP.

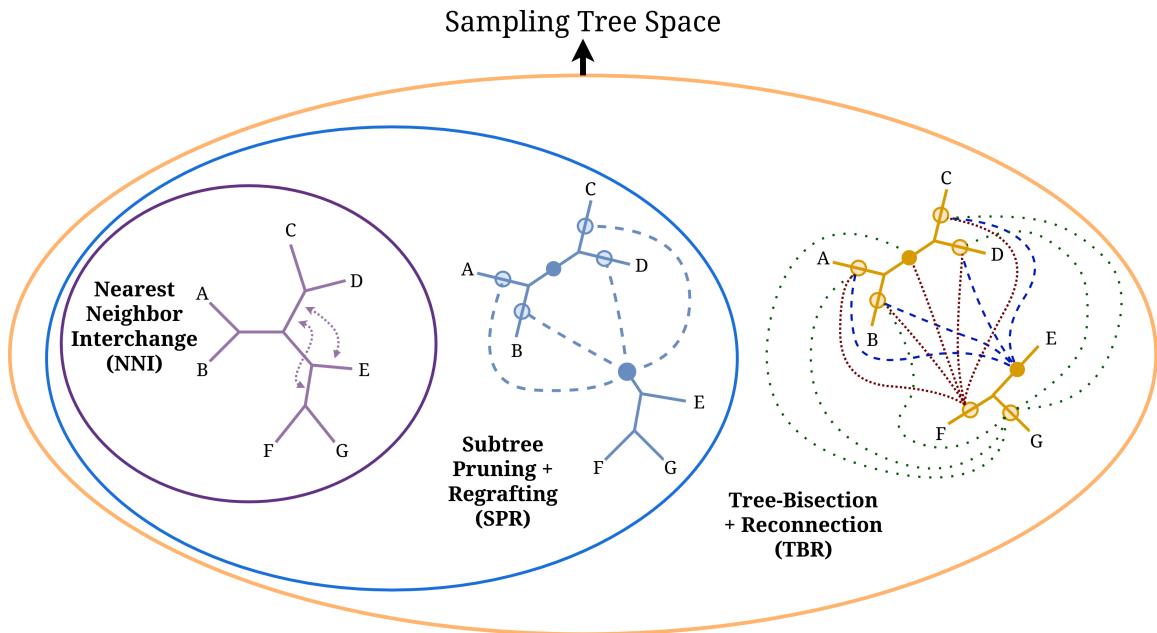


**Hình 2.3** — Ví dụ đơn giản về tiêu chí MP

### 2.1.3. Các kỹ thuật biến đổi cây thông dụng

Khi xây dựng cây tiến hóa, một trong những thách thức lớn là tìm ra cấu trúc cây tối ưu nhất phù hợp với dữ liệu sinh học. Do không gian cây tiến hóa rất rộng lớn (hàm giai thừa của  $n$  với cây có  $n$  lá), các phương pháp đơn giản như so sánh trực tiếp giữa tất cả các cấu trúc cây có thể không thực tế do chi phí tính toán quá lớn. Vì vậy, các chiến lược heuristics để tìm kiếm cây đủ tốt thường được áp dụng [14]. Từ đây, các kỹ thuật biến đổi cây đã được phát triển để cải thiện hiệu quả tìm kiếm và tối ưu hóa cây tiến hóa.

Các kỹ thuật biến đổi cây giúp thay đổi cấu trúc của cây hiện tại thông qua các phép toán chỉnh sửa cây, nhằm khám phá các cấu trúc cây khác có thể phù hợp hơn với dữ liệu. Những phép toán này giúp tối ưu hóa cây tiến hóa bằng cách điều chỉnh các nhánh và cấu trúc của cây mà không cần phải xây dựng lại cây từ đầu. Trong đó, các kỹ thuật như nearest-neighbor interchange (NNI), subtree pruning and regrafting (SPR), và tree bisection and reconnection (TBR) là những phương pháp biến đổi cây thông dụng trong phân tích tiến hóa.

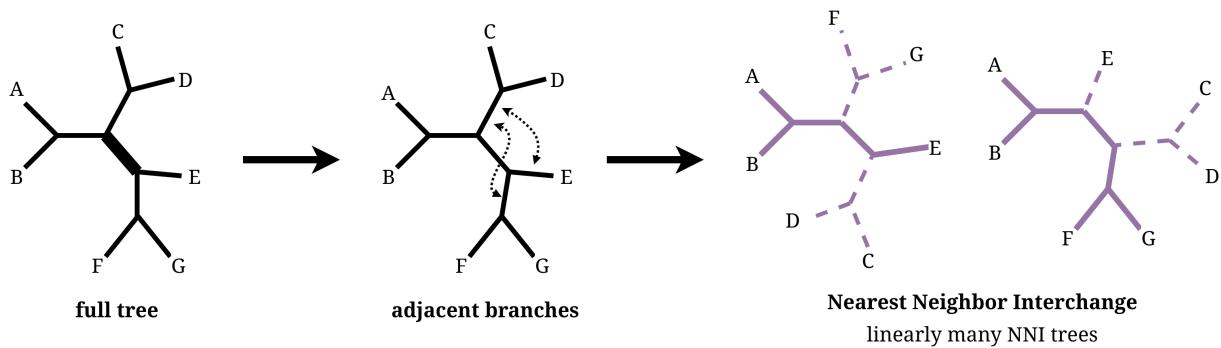


**Hình 2.4** — Minh họa các phép biến đổi cây

#### 2.1.3.1. Nearest Neighbor Interchange (NNI)

Nearest Neighbor Interchange (NNI) (xem [Hình 2.5](#)) là một trong những phép biến đổi cây đơn giản và phổ biến nhất trong phân tích phát sinh loài. Cụ thể, NNI thực hiện trên một cặp nhánh kề nhau, thay thế chúng bằng hai cấu trúc cây thay thế. Phép toán này có phạm vi hạn chế vì chỉ cho phép thay đổi giữa hai nút liền kề, nghĩa là không khám phá tất cả các khả năng sắp xếp cây. Tuy nhiên, sự đơn giản và chi phí tính toán tương đối thấp khiến NNI trở thành phương pháp phổ biến để tối ưu hóa cây.

NNI đặc biệt hữu ích trong việc khám phá nhanh chóng các thay đổi nhỏ trong một phần cây, làm cho nó hiệu quả trong việc cải thiện khả năng (likelihood) hoặc điểm tính toán parsimony của cây mà không cần thay đổi cấu trúc lớn. Phép biến đổi này cũng dễ tính toán và cài đặt, vì chỉ thay đổi cấu trúc của một phần nhỏ trong cây.



**Hình 2.5** — Minh họa phép biến đổi cây NNI

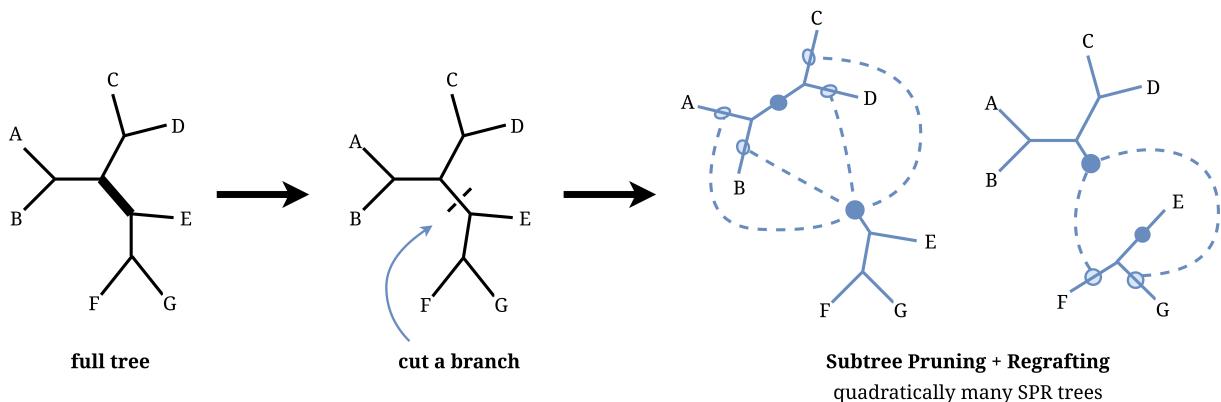
### 2.1.3.2. Subtree pruning and regrafting (SPR)

Subtree Pruning and Regrafting (SPR) (xem [Hình 2.6](#)) là một phép toán cây phức tạp hơn so với NNI. Nó bao gồm hai bước chính:

- Cắt bỏ một nhánh con (prune) khỏi cây (loại bỏ một nhóm các nút và các cạnh nối với chúng).
- Nối lại nhánh con đã cắt vào một nhánh khác trong cây gốc.

SPR cho phép thực hiện các thay đổi lớn hơn trong cấu trúc cây so với NNI vì nó có thể di chuyển nhánh con qua các phần lớn hơn của cây. Phép toán này mở ra một phạm vi rộng hơn các thay đổi cấu trúc cây, có thể dẫn đến các giải pháp tốt hơn về khả năng (likelihood) hoặc điểm parsimony.

Mặc dù tồn kén về mặt tính toán hơn NNI, SPR có thể cung cấp những cái nhìn sâu hơn về các mối quan hệ phát sinh loài giữa các taxon. SPR đặc biệt hữu ích trong các trường hợp cây có thể có các mối quan hệ phát sinh loài phức tạp, yêu cầu các thay đổi phức tạp hơn để tìm kiếm một cách hiệu quả.

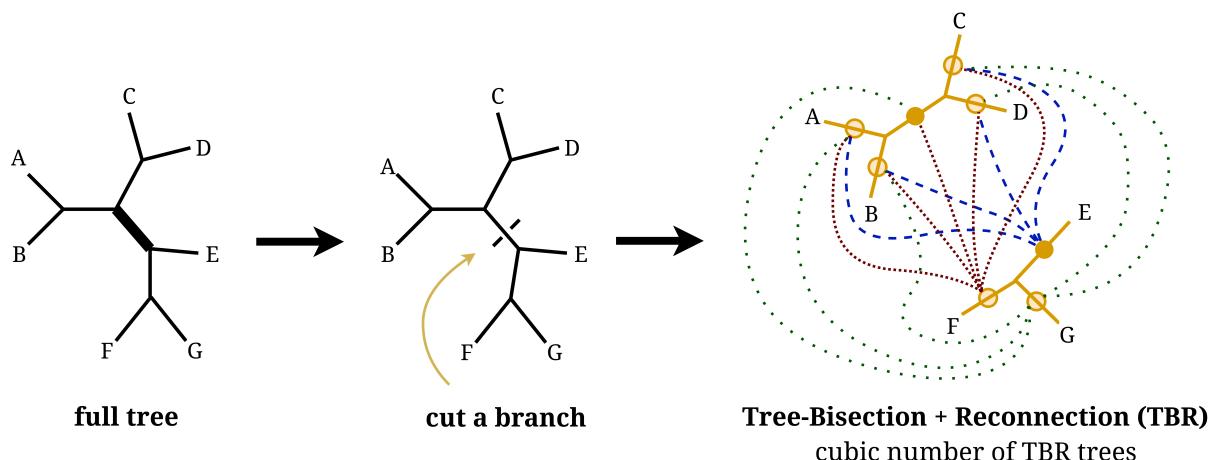


**Hình 2.6** — Minh họa phép biến đổi cây SPR

### 2.1.3.3. Tree bisection and reconnection (TBR)

Tree Bisection and Reconnection (TBR) (xem [Hình 2.7](#)) là một trong những phép toán cây mạnh mẽ nhất trong phân tích phát sinh loài, cho phép thực hiện các thay đổi lớn nhất về cấu trúc cây. TBR hoạt động bằng cách chia đôi cây thành hai phần, thường là cắt một cạnh, sau đó kết nối lại hai phần này theo một cấu trúc mới. Phép toán này có thể được thực hiện theo nhiều cách khác nhau, cung cấp một số lượng lớn các khả năng tái kết nối để khám phá.

TBR rất linh hoạt và có khả năng khám phá không gian cây toàn diện hơn so với NNI hoặc SPR. Điều này làm cho nó trở thành phương pháp mạnh mẽ để tối ưu hóa cây phát sinh loài, vì nó có thể tìm ra các cấu trúc cây chính xác hơn, phù hợp với dữ liệu tốt hơn. Tuy nhiên, vì độ phức tạp của nó, TBR tốn kém về mặt tính toán và có thể yêu cầu nhiều tài nguyên và thời gian hơn để thực hiện. Dù vậy, TBR thường được sử dụng trong các trường hợp mà các phương pháp tìm kiếm cây phức tạp hơn là cần thiết để bao quát hết sự đa dạng của các cấu trúc cây có thể có.



**Hình 2.7** — Minh họa phép biến đổi cây TBR

### 2.1.4. Các công trình liên quan

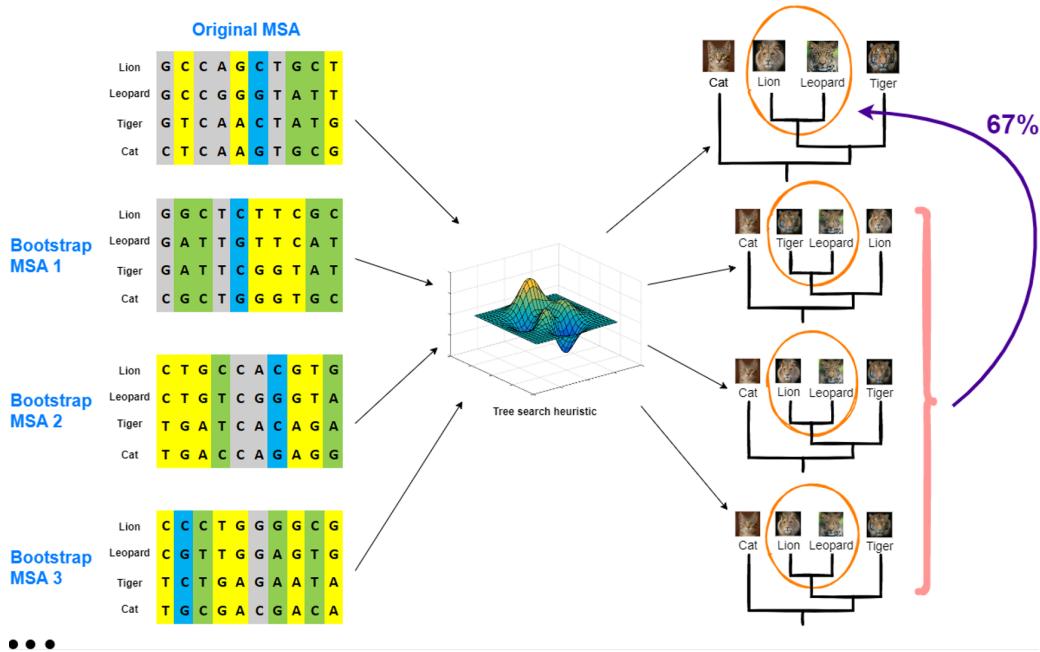
#### 2.1.4.1. Các công trình sử dụng phương pháp bootstrap chuẩn

Phương pháp bootstrap chuẩn xây dựng tập cây  $\mathcal{B}$  thông qua  $B$  lần chạy thuật toán tìm kiếm cây độc lập với mỗi  $A_b$ . Các công trình sử dụng phương pháp này bao gồm TNT [15], PAUP\* [16], MEGA [17]. Vì phương pháp bootstrap chuẩn có xu hướng đánh giá thấp khả năng đúng của một phân hoạch nhị phân

[18,19], nên quy tắc thực hành phổ biến là coi các cạnh có giá trị hỗ trợ bootstrap hơn 70% là đáng tin cậy.

#### 2.1.4.2. Phương pháp MPBoot

Phương pháp MPBoot sử dụng tiêu chuẩn maximum parsimony (với ưu điểm là tính đơn giản, dễ cài đặt và hiệu quả trong thiết kế cấu trúc dữ liệu) cùng với phương pháp xấp xỉ bootstrap để giải quyết bài toán xây dựng cây bootstrap tiến hóa. Phương pháp xấp xỉ bootstrap trong MPBoot xác định tập hợp  $\mathcal{B}$  bằng cách thực hiện một lần tìm kiếm cây trong không gian cây biểu diễn sáp hàng gốc  $A^{\text{data}}$  (xem [Hình 2.8](#)).



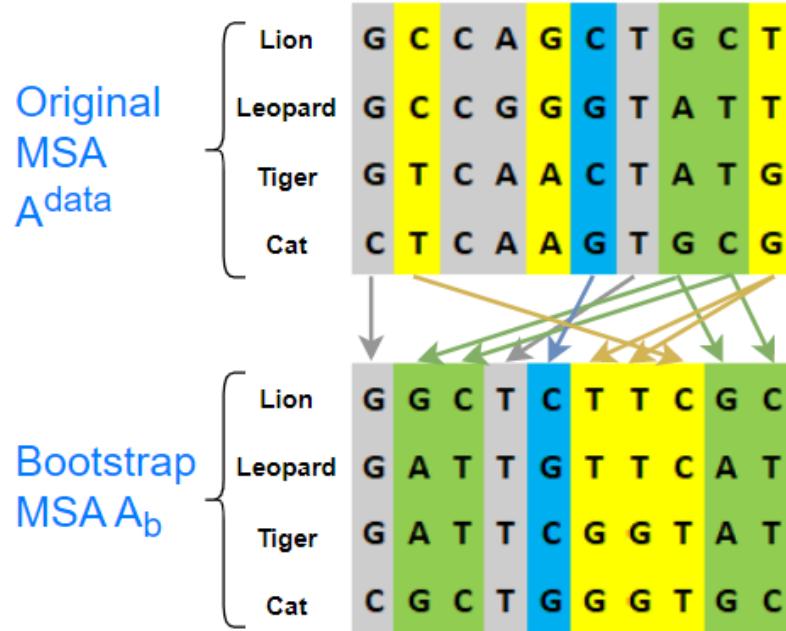
**Hình 2.8 —** Phương pháp xấp xỉ bootstrap

Đối với một cây  $T$  gấp phải trong quá trình tìm kiếm leo đồi SPR, MPBoot tính toán điểm số của cây này trên từng sáp hàng bootstrap  $b$  ( $A_b^{\text{data}}$ ), sau đó cập nhật cây bootstrap cho  $A_b^{\text{data}}$  nếu  $T$  có điểm số MP tốt hơn. Vì quy trình này tiêu tốn nhiều thời gian nên cần phải tính toán hiệu quả điểm MP của  $T$  trên  $A_b^{\text{data}}$ . Để tối ưu hóa công đoạn này, MPBoot sử dụng Resampling parsimony score (REPS) [9] cho từng sáp hàng bootstrap ([Hình 2.9](#)).

Đối với một cây  $T$  và điểm MP tại các cột  $D_i$  ( $\text{MP}(T | D_i)$ ) được tính từ  $A^{\text{data}}$ , điểm MP cho  $A_b^{\text{data}}$  được tính nhanh chóng dưới dạng tổng điểm MP tại các cột tương ứng được sử dụng trong sáp hàng bootstrap  $b$ :

$$\text{MP}(T \mid A_b^{\text{data}}) = \sum_{i=1}^k \text{MP}(T \mid D_i) \cdot d_i^b$$

trong đó  $d_i^b$  là tần suất xuất hiện của cột  $D_i$  trong  $A_b^{\text{data}}$ . Nhờ vậy, không cần phải tính lại điểm tiết kiệm cho từng cột, từng lần lặp bootstrap và từng cây.



**Hình 2.9** — Minh họa sử dụng kỹ thuật REPS để tính điểm MP của sáp hàng bootstrap

Với phương pháp bootstrap xấp xỉ trên, một cạnh được coi là đáng tin cậy nếu giá trị hỗ trợ bootstrap theo MPBoot của cạnh đó cao hơn 95% [9]. MPBoot hiện tại cài đặt hai kỹ thuật biến đổi cây bao gồm NNI và SPR, trong đó SPR được sử dụng chủ đạo trong suốt quá trình tìm kiếm của MPBoot.

## 2.2. Giải thuật tối ưu đàm kién

Giải thuật tối ưu đàm kién (Ant Colony Optimization - ACO) là một kỹ thuật tối ưu hóa dựa trên hành vi tìm đường của đàm kién trong tự nhiên. Đây là một trong những giải thuật thuộc nhóm trí tuệ bầy đàm (Swarm Intelligence), được đề xuất bởi Marco Dorigo vào năm 1992 [20].

### **2.2.1. Tổng quan**

Trong tự nhiên, các cá thể kiến di chuyển ngẫu nhiên và khi tìm thấy thức ăn, chúng quay trở về tổ, đồng thời để lại dấu vết pheromone. Nếu các con kiến khác tìm thấy con đường này, chúng sẽ có xu hướng ngừng di chuyển ngẫu nhiên và thay vào đó đi theo dấu vết pheromone, đồng thời quay lại và củng cố dấu vết đó nếu cuối cùng chúng tìm được thức ăn.

Tuy nhiên, theo thời gian, dấu vết pheromone bắt đầu bay hơi, làm giảm “sức hút” của đường đi đó. Với một con đường dài, thời gian kiến cần để di chuyển tìm kiếm thức ăn và quay lại sẽ càng lâu, dẫn đến các vết mùi pheromone càng bị bay hơi lâu hơn. Ngược lại, một con đường ngắn sẽ được di chuyển thường xuyên hơn, dẫn đến mật độ pheromone trên con đường ngắn cao hơn so với các con đường dài. Sự bay hơi pheromone cũng có lợi thế là tránh hội tụ vào một giải pháp tối ưu cục bộ. Nếu không có sự bay hơi, các con đường được chọn bởi những con kiến đầu tiên sẽ trở nên quá hấp dẫn đối với các con kiến tiếp theo, làm hạn chế việc khám phá không gian giải pháp. Mặc dù tác động của sự bay hơi pheromone trong hệ thống tự nhiên chưa rõ ràng, nhưng trong các hệ thống nhân tạo, sự bay hơi này đóng vai trò rất quan trọng.

Kết quả chung là khi một con kiến tìm thấy một con đường tốt (ví dụ như ngắn) từ tổ đến nguồn thức ăn, các con kiến khác có nhiều khả năng đi theo con đường đó, và phản hồi tích cực sẽ dẫn đến nhiều con kiến cùng theo một con đường duy nhất. Ý tưởng của thuật toán bầy kiến là mô phỏng hành vi này bằng cách sử dụng “kiến mô phỏng” di chuyển trên đồ thị đại diện cho bài toán cần giải quyết.

### **2.2.2. Cấu trúc thuật toán**

Trong các giải thuật tối ưu đàm kiến (ant colony optimization), một con kiến nhân tạo là một “agent” tính toán đơn giản, tìm kiếm các giải pháp tốt cho một bài toán tối ưu hóa nhất định. Để áp dụng thuật toán đàm kiến, bài toán tối ưu hóa cần được chuyển đổi thành bài toán tìm đường đi ngắn nhất trên một đồ thị có trọng số.

Giải thuật ACO hoạt động theo các bước chính sau:

- Khởi tạo các tham số và lượng pheromone ban đầu.

- Xây dựng giải pháp: Mỗi con kiến xây dựng một giải pháp hoàn chỉnh dựa trên xác suất lựa chọn được tính từ ma trận pheromone và thông tin heuristic.
- Cập nhật pheromone: Sau khi tất cả kiến hoàn thành tour của mình, lượng pheromone được cập nhật. Đường đi tốt sẽ được tăng cường pheromone, trong khi các đường kém sẽ bị bay hơi dần.
- Lặp lại quá trình cho đến khi đạt điều kiện dừng.

### 2.2.2.1. Xây dựng giải pháp

Mỗi con kiến cần xây dựng một giải pháp để di chuyển qua đồ thị. Để chọn cạnh tiếp theo trong hành trình của mình, một con kiến sẽ xem xét độ dài của mỗi cạnh có sẵn từ vị trí hiện tại, cũng như mức độ pheromone tương ứng. Ở mỗi bước của thuật toán, mỗi con kiến di chuyển từ trạng thái  $x$  đến trạng thái  $y$ , tương ứng với một giải pháp trung gian đầy đủ hơn. Do đó, mỗi con kiến  $k$  tính toán một tập  $A_{k(x)}$  các mở rộng khả thi đối với trạng thái hiện tại của nó trong mỗi vòng lặp và di chuyển đến một trong các mở rộng này với xác suất. Đối với con kiến  $k$ , xác suất  $p_{xy}^k$  di chuyển từ trạng thái  $x$  đến trạng thái  $y$  phụ thuộc vào sự kết hợp của hai giá trị, sự hấp dẫn  $\eta_{xy}$  của bước di chuyển, được tính toán bằng một số chiến lược heuristic chỉ ra mức độ ưu tiên đối với bước di chuyển đó và mức độ pheromone  $\tau_{xy}$  của bước di chuyển, chỉ ra mức độ hiệu quả của bước di chuyển đó trong quá khứ. Mức độ pheromone đại diện cho sự chỉ dẫn dựa trên kinh nghiệm về mức độ mong muốn của bước di chuyển đó.

Thông thường, con kiến  $k$  di chuyển từ trạng thái  $x$  đến trạng thái  $y$  với xác suất:

$$p_{xy}^k = \frac{(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}{\sum_{z \in \text{allowed}_y} (\tau_{xz}^\alpha)(\eta_{xz}^\beta)}$$

trong đó  $\tau_{xy}$  là lượng pheromone được để lại khi chuyển từ trạng thái  $x$  đến trạng thái  $y$ ,  $\alpha \geq 0$  là tham số điều khiển ảnh hưởng của  $\tau_{xy}$ ,  $\eta_{xy}$  là mức độ “hấp dẫn” khi chuyển trạng thái từ  $x$  đến  $y$  (thường được gọi là thông tin heuristic) và  $\beta \geq 1$  là tham số điều khiển ảnh hưởng của  $\eta_{xy}$ .  $\tau_{xz}$  và  $\eta_{xz}$  đại diện cho mức độ pheromone và sự hấp dẫn đối với các cách chuyển trạng thái khác có thể có.

### 2.2.2.2. Cập nhật pheromone

Các dấu vết thường được cập nhật khi tất cả các con kiến đã hoàn thành giải pháp của chúng, tăng hoặc giảm mức độ pheromone của các giải pháp tương ứng với giải pháp đó “tốt” hay “xấu”. Một ví dụ về quy tắc cập nhật pheromone toàn cục là:

$$\tau_{xy} \leftarrow (1 - \rho)\tau_{xy} + \sum_k^m \Delta\tau_{xy}^k$$

trong đó  $\tau_{xy}$  là lượng pheromone được để lại khi chuyển trạng thái từ  $x$  đến  $y$ ,  $\rho$  là hệ số bay hơi pheromone,  $m$  là số lượng con kiến và  $\Delta\tau_{xy}^k$  là lượng pheromone được thả ra bởi con kiến  $k$ , thường được cho theo công thức:

$$\Delta\tau_{xy}^k := \begin{cases} Q/L_k & \text{nếu kiến } k \text{ sử dụng cạnh } xy \text{ trong lời giải} \\ 0 & \text{nếu không} \end{cases}$$

trong đó  $L_k$  là chi phí của hành trình của con kiến  $k$  (thường là chiều dài) và  $Q$  là một hằng số.

### 2.2.3. Các biến thể của giải thuật tối ưu đàm kiến

Thuật toán được đề cập ở trên là thuật toán ACO đầu tiên, có tên gọi là thuật toán Ant System (AS) [20].

#### 2.2.3.1. Thuật toán Ant Colony System (ACS)

Trong thuật toán ACS, hệ thống kiến gốc đã được chỉnh sửa ở ba khía cạnh:

- Việc chọn cạnh nghiêng về khai thác (tức là ưu tiên chọn các cạnh ngắn nhất có lượng pheromone lớn);
- Trong quá trình xây dựng giải pháp, các con kiến thay đổi mức độ pheromone của các cạnh mà chúng đang chọn bằng cách áp dụng một quy tắc cập nhật pheromone cục bộ;
- Vào cuối mỗi vòng lặp, chỉ có con kiến tốt nhất mới được phép cập nhật các dấu vết bằng cách áp dụng một quy tắc cập nhật pheromone toàn cục đã được sửa đổi.

#### 2.2.3.2. Thuật toán Elitist ant system (EAS)

Trong thuật toán này, giải pháp tốt nhất toàn cục sẽ để lại pheromone trên dấu vết của nó sau mỗi vòng lặp (ngay cả khi dấu vết này không được quay lại),

cùng với tất cả các con kiến khác. Chiến lược ưu tú có mục tiêu chỉ đạo quá trình tìm kiếm của tất cả các con kiến để xây dựng một giải pháp chứa các liên kết của tuyến đường tốt nhất hiện tại.

#### **2.2.3.3. Thuật toán Max–Min Ant System (MMAS)**

Thuật toán này kiểm soát lượng pheromone tối đa và tối thiểu trên mỗi dấu vết. Chỉ có chuyến đi tốt nhất toàn cục hoặc chuyến đi tốt nhất trong vòng lặp mới được phép thêm pheromone vào dấu vết của nó. Để tránh sự trì trệ trong thuật toán tìm kiếm, phạm vi lượng pheromone có thể có trên mỗi dấu vết bị giới hạn trong một khoảng  $[\tau_{\min}, \tau_{\max}]$ . Tất cả các cạnh đều được khởi tạo với  $\tau_{\max}$  để thúc đẩy việc khám phá các giải pháp cao hơn. Các dấu vết sẽ được khởi tạo lại với  $\tau_{\max}$  khi gần đến mức trì trệ.

#### **2.2.3.4. Thuật toán Rank-based ant system (ASrank)**

Tất cả các giải pháp đều được xếp hạng theo chiều dài của chúng. Chỉ có một số lượng con kiến tốt nhất trong vòng lặp này mới được phép cập nhật dấu vết của chúng. Lượng pheromone được lưu lại sẽ được cân nhắc cho từng giải pháp, sao cho các giải pháp có đường đi ngắn hơn sẽ lưu lại nhiều pheromone hơn các giải pháp có đường đi dài hơn.

#### **2.2.3.5. Thuật toán đàn kiến song song (PACO)**

Do các cá thể kiến trong thuật toán rất độc lập với nhau nên các cá thể kiến có thể được chia thành các nhóm và chạy đồng thời trên các bộ xử lý khác nhau. Điều này giúp tăng hiệu suất của thuật toán, tuy nhiên cần có các phương pháp trao đổi thông tin vết mì pheromone giữa các nhóm một cách hiệu quả. Có một số phương pháp trao đổi vết mì thông dụng như all-to-all, directed/undirected ring, hypercube, random,...

#### **2.2.3.6. Thuật toán Max–Min trơn (SMMAS)**

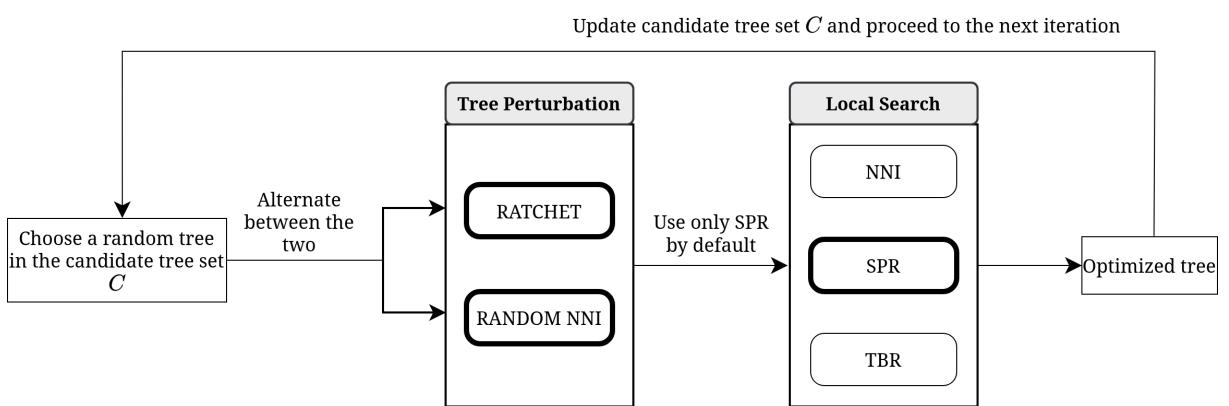
Một cải tiến so với giải thuật đòn kiến gốc là giải thuật đòn kiến Max–Min (MMAS) (đặt thêm giới hạn trên và dưới cho các giá trị mùi nhằm thúc đẩy việc khám phá và tránh sự hội tụ sớm). Xây dựng trên MMAS, giải thuật đòn kiến Max–Min trơn (SMMA) [21] tích hợp một cơ chế làm mượt điều chỉnh vào quy tắc cập nhật mùi nhân tạo. Yếu tố làm mượt này giúp điều chỉnh tốc độ thay đổi của mùi nhân tạo, chuyển từ việc tập trung vào việc khám phá vào giai đoạn đầu của tìm kiếm đến việc khai thác mạnh mẽ hơn khi thuật toán hội tụ.

# Chương 3

## Tích hợp phép biến đổi TBR vào MPBoot

### 3.1. Khung thuật toán MPBoot

Để giải bài xây dựng cây tiến hóa bootstrap, thuật toán MPBoot duy trì một tập cây  $\mathcal{C}$  gồm  $C$  cây tốt nhất tìm được cho tập sáp hàng (MSAs) ban đầu. Tập hợp này được sinh ở pha khởi tạo (pha 1) nhờ chạy 100 lần thủ tục randomized stepwise addition rồi tối ưu bằng leo đồi SPR và chọn ra  $C$  cây tốt nhất. Tập hợp  $\mathcal{C}$  tiếp tục được cải thiện qua pha khám phá (pha 2) (xem [Hình 3.1](#)) nhờ chiến lược lặp phá cây chọn ngẫu nhiên trong  $\mathcal{C}$  rồi leo đồi SPR trên kết quả. Việc phá cây ở pha khám phá được thực hiện nhờ luân phiên (i) random NNIs và (ii) ratchet dùng leo đồi SPR. Ngoài ra, tập cây bootstrap  $\mathcal{B}$  được cập nhật cùng với việc tìm kiếm cây. Ở pha tinh chỉnh bootstrap (pha 3), mỗi cây bootstrap sẽ được tối ưu nhờ leo đồi SPR trên từng MSA bootstrap.



**Hình 3.1** — Một vòng lặp trong pha khám phá (pha 2) của khung thuật toán MPBoot.

### 3.2. Việc áp dụng phép biến đổi cây trong thuật toán MPBoot gốc

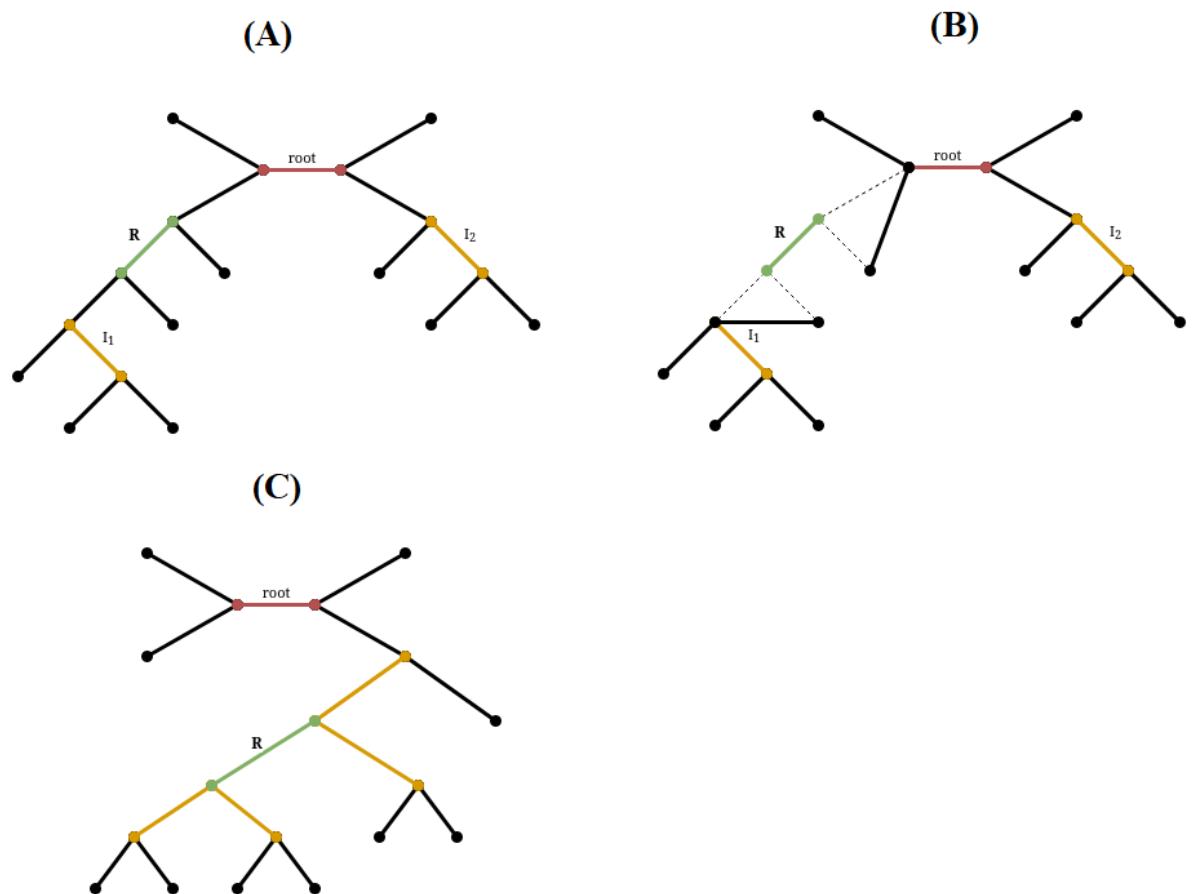
Như vậy trong MPBoot phép biến đổi cây xuất hiện ở cả 3 pha. Việc tối ưu tất cả những thủ tục leo đồi nhờ SPR bằng leo đồi dựa trên một phép biến đổi cây mạnh hơn (ví dụ như TBR) có thể giúp vừa tìm được cây  $T^{\text{best}}$  có điểm số MP tốt hơn vừa tìm được tập cây bootstrap tốt hơn.

### 3.3. Đề xuất tính toán nhanh một phép biến đổi TBR

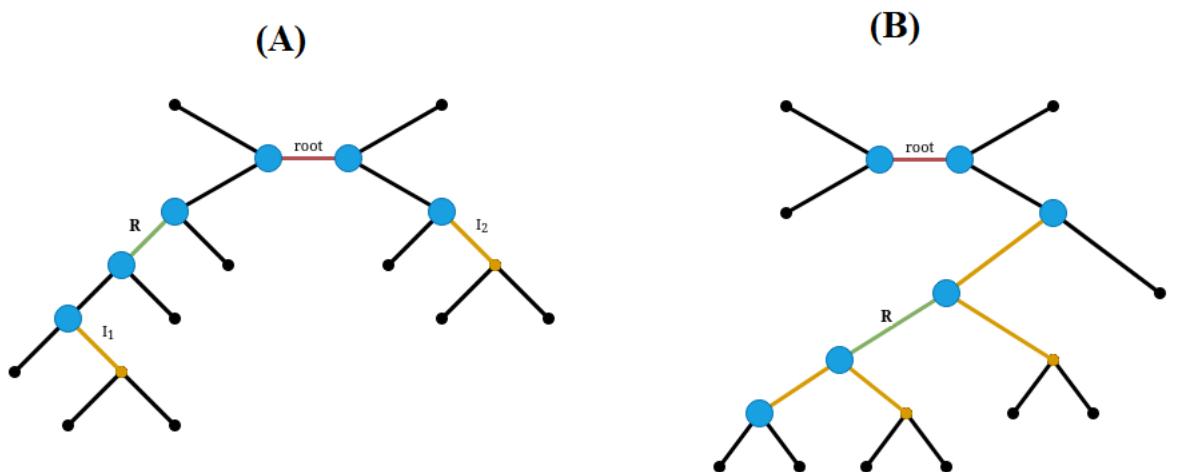
Xét một phép biến đổi TBR trên cây  $T^{\text{lst}}$  (xem [Hình 3.2A](#)). Gọi cạnh  $R$  là cạnh cắt của phép TBR. Trong trường hợp tổng quát, cắt cây  $T^{\text{lst}}$  tại  $R$  tạo ra ba phần tạm thời tách biệt nhau gồm hai cây con  $T_1, T_2$  và cạnh  $R$  (xem [Hình 3.2B](#)). Cạnh  $R$  sau đó sẽ được dùng làm cạnh trung gian để nối  $T_1$  và  $T_2$  (xem [Hình 3.2C](#)). Giả sử cặp cạnh nối là  $(I_1, I_2)$  với  $I_1$  thuộc cây con  $T_1$ ,  $I_2$  thuộc cây con  $T_2$ . Gọi  $T^*$  là cây kết quả nối  $I_1$  và  $I_2$ .

Theo tiếp cận trực tiếp thì việc đánh giá lại điểm của cây  $T^*$  được thực hiện thuận bằng việc duyệt post-order lại toàn bộ cây và tính theo thuật toán Fitch hoặc Sankoff. Tuy nhiên, điểm parsimony của nhiều cây con không thay đổi trước và sau khi nối tạo thành cây  $T^*$ .

Sau đây, chúng tôi đề xuất phương pháp chỉ tính lại điểm của những đỉnh có thay đổi điểm số như sau. Giả sử cây  $T^{\text{lst}}$  đã được tính toán điểm parsimony cho từng đỉnh sử dụng gốc đặt trên cạnh gốc (root) nối 2 đỉnh  $\text{root}_1$  và  $\text{root}_2$ . Mỗi đỉnh sẽ lưu điểm parsimony của cây con tương ứng (xem [Hình 3.3A](#)). Xét cây  $T^*$  với gốc đặt trên cạnh  $R$ . Khi đó những đỉnh cần phải tính lại điểm parsimony là những đỉnh của cây  $T^{\text{lst}}$  thuộc đường đi nối hai cạnh  $I_1$  và root và những đỉnh thuộc đường đi nối hai cạnh  $I_2$  và root (xem [Hình 3.3B](#)). Để xác định được những đỉnh cần tính lại điểm như trên, với mỗi đỉnh, ta lưu thêm một biến con trả tới đỉnh cha ứng với cây  $T^{\text{lst}}$ . Khi đó, việc tìm kiếm và đánh dấu những đỉnh cần tính lại được thực hiện bằng vòng lặp từ  $I_1$  và  $I_2$  nhảy lên đỉnh cha cho tới khi lên tới gốc của  $T^{\text{lst}}$ . Cuối cùng, trên cây  $T^*$  xét gốc ở cạnh  $R$ , ta thực hiện tính toán điểm và chỉnh sửa biến con trả tới đỉnh cha tương ứng ở những đỉnh được đánh dấu tính lại. Sau mỗi phép biến hình cây TBR, cây  $T^*$  sẽ chính là cây  $T^{\text{lst}}$  cho lượt thử tiếp theo.



**Hình 3.2** — Một phép biến đổi TBR với cạnh cắt  $R$  và hai cạnh nối  $I_1$  và  $I_2$



**Hình 3.3** — Nhận diện những đỉnh cần phải tính lại điểm với cạnh cắt  $R$  và hai cạnh nối  $I_1$  và  $I_2$

### 3.4. Đề xuất tìm kiếm cây lân cận sử dụng TBR

#### 3.4.1. Chiến lược tìm kiếm “tốt nhất”

Thuật toán tìm kiếm lân cận sử dụng các phép biến hình cây TBR trên một cây  $T$  với cạnh cắt  $R$  được thực hiện như sau:

- Chọn cạnh  $R$  là cạnh cắt của các phép TBR cần khảo sát. Trong trường hợp tổng quát, cắt cây  $T$  tại  $R$  tạo ra hai cây con  $T_1, T_2$  và cạnh  $R$  tạm thời tách biệt với nhau. Cạnh  $R$  sau đó sẽ được dùng làm cạnh trung gian để nối  $T_1$  và  $T_2$ .
- Xét lần lượt các cặp cạnh nối  $(I_1, I_2)$  với  $I_1$  thuộc cây con  $T_1$ ,  $I_2$  thuộc cây con  $T_2$  và khoảng cách giữa  $I_1, I_2$  ở trên cây ban đầu nằm trong khoảng  $[mintrav, maxtrav]$  cho trước.
  - Thực hiện nối hai cạnh  $I_1$  và  $I_2$  thông qua cạnh  $R$ . (xem [Hình 3.2C](#))
  - Cây kết quả nhận được là cây  $T^*$ . Tính toán, đánh giá cây  $T^*$  thông qua điểm parsimony. Cập nhật cây lân cận tốt nhất tìm được  $T^{best}$ .
  - Thực hiện cắt cạnh  $R$  một lần nữa, nhằm khảo sát những cặp  $(I_1, I_2)$  tiếp theo.
- Sau khi tìm kiếm kết thúc, ta sẽ tìm được cây  $T^{best}$  tốt nhất khi thực hiện các phép TBR trên cây  $T$  với cạnh cắt  $R$ .
- Nối lại cạnh  $R$  vào vị trí ban đầu, rollback về cây  $T$  ban đầu.

Thuật toán được mô tả bằng mã giả trong [Thuật toán 3.1](#) sau đây. Khi đó, với cây  $T$ , cạnh  $R$ , giá trị  $mintrav$  và  $maxtrav$  sau khi thực hiện thủ tục  $COMPUTETBR(R, mintrav, maxtrav)$  sẽ tìm được cây  $T^{best}$  (tương đương với tìm được cặp cạnh tốt nhất  $(I_1, I_2)$ ).

Những trường hợp đặc biệt như  $R$  không phải là cạnh trong (nối với một đỉnh lá) được xử lý riêng do công đoạn cắt cạnh và nối cạnh có phần khác biệt. Ngoài ra, để tính nhanh điểm  $MP(T^*)$ , ta cũng sẽ đổi cạnh gốc của cây  $T^{lst}$  thành chính cạnh cắt  $R$  sau lượt  $TESTTBRMOVE$  đầu tiên và giữ nguyên cho tới khi xét cạnh cắt tiếp theo. Điều này đảm bảo số lượng đỉnh cần phải tính lại sẽ không quá  $O(maxtrav)$  đỉnh (không xét lượt thử đầu tiên do gốc có thể khác  $R$ ).

**Thuật toán 3.1** — Thực hiện phép biến đổi TBR với cạnh cắt  $R$  trên cây  $T$

<b>Input</b>	Tree $T$ Remove-branch $R$ Radius criteria $\mintrav$ and $\maxtrav$ for insert-branches $I_1, I_2$
<hr/> <b>Output</b> Best found tree $T^{\text{bestNei}}$ (best $(I_1, I_2)$ ) with remove-branch $R$	

```

1 COMPUTETBR( $R, \mintrav, \maxtrav$ )
2
3 Function TESTTBRMOVE( $I_1, I_2$ )
4   //  $T$  is already cut into  $R, T_1, T_2$  when called TESTTBRMOVE()
5   Connect branch  $I_1$  and  $I_2$  using  $R$ , result in  $T^*$ 
6   Evaluate parsimony score  $\text{MP}(T^*)$  of  $T^*$ 
7   if  $\text{MP}(T^*) < \text{MP}(T^{\text{bestNei}})$  then
8      $T^{\text{bestNei}} := T^*$ 
9      $I_1^{\text{bestNei}} := I_1$ 
10     $I_2^{\text{bestNei}} := I_2$ 
11  end if
12  Remove branch  $R$ , rollback the changes
13
14 Function COMPUTETBR( $R, \mintrav, \maxtrav$ )
15   Remove branch  $R$  from tree  $T$ 
16   // Find all valid  $(I_1, I_2)$  can be done recursively via DFS
17   for each  $(I_1, I_2)$  satisfied
18     TESTTBRMOVE( $I_1, I_2$ )
19   end for
20   Reconnect branch  $R$ , rollback to  $T$ 
21    $T = \text{APPLYTBR}(R, I_1^{\text{bestNei}}, I_2^{\text{bestNei}})$ 

```

### 3.4.2. Chiến lược tìm kiếm “tốt hơn”

Chúng tôi cũng đề xuất một thuật toán tìm kiếm lân cận TBR tương tự [Thuật toán 3.1](#) nhưng với một số thay đổi nhỏ. Ở thuật toán trên, với mỗi cạnh cắt  $R$ , ta cập nhật cây hiện tại tối đa 1 lần (nếu như cây  $T^{\text{best}}$  cho kết quả tốt hơn). Ở thuật toán thay đổi này (xem [Thuật toán 3.2](#)), với mỗi cặp cạnh cắt  $R$  và cạnh nối  $I_1$  ta cập nhật cây hiện tại tối đa 1 lần. Chi tiết hơn, ta sẽ xét mọi

cạnh nối  $I_2$  thỏa mãn, tìm cây  $T^{\text{best}}$  và cập nhật cho cây hiện tại nếu cho kết quả tốt hơn.

**Thuật toán 3.2** — Chiến thuật tìm kiếm “tốt hơn” sử dụng TBR

```

1 Function COMPUTETBR( $R$ ,  $\text{mintrav}$ ,  $\text{maxtrav}$ )
2   for each  $I_1$  satisfied
3     Remove branch  $R$  from tree  $T$ 
4     for each  $I_2$  satisfied
5       TESTTBRMOVE( $I_1, I_2$ )
6     end for
7     Reconnect branch  $R$ , rollback to  $T$ 
8      $T = \text{APPLYTBR}(R, I_1, I_2^{\text{bestNei}})$ 
9   end for
```

### 3.5. Đề xuất thuật toán leo đồi TBR

**Thuật toán 3.3** — Thuật toán leo đồi sử dụng TBR trên cây  $T$

<b>Input</b>	Tree $T$ Radius criteria $\text{mintrav}$ and $\text{maxtrav}$ for insert-branches $I_1, I_2$
--------------	--

---

<b>Output</b>	Tree $T$ updated to best found neighbor tree $T^{\text{bestNei}}$ consider every remove-branch $R$
---------------	--

---

```

1 do
2   for each branch  $R$  in  $T$ 
3      $T^{\text{bestNei}} := \text{NULL}$ 
4     COMPUTETBR( $R$ ,  $\text{mintrav}$ ,  $\text{maxtrav}$ )
5     if  $\text{MP}(T^{\text{bestNei}}) < \text{MP}(T)$  then
6        $T := T^{\text{bestNei}}$ 
7     end if
8   end for
9 while  $\text{MP}(T)$  still improves
10 end do
```

Thuật toán leo đồi TBR thực hiện leo đồi cập nhật cây  $T$  bằng cây  $T^{\text{best}}$  tìm được (nếu  $T^{\text{best}}$  cho kết quả tốt hơn) với mỗi cạnh cắt  $R$  khảo sát được mô

tả ở [Thuật toán 3.3](#). Vòng lặp leo đồi sẽ tiếp tục trong khi cây  $T$  vẫn được cập nhật bởi một cây tối ưu hơn.

Khi leo đồi sử dụng hai cách tìm kiếm lân cận TBR khác nhau (sau đây gọi tắt là hai cách leo đồi “tốt nhất” và “tốt hơn”) thì mẫu không gian cây khảo sát được cung cấp khác nhau.

### 3.6. Đề xuất thuật toán MPBoot-TBR

Chúng tôi đề xuất MPBoot-TBR (xem [Thuật toán 3.4](#)) bằng cách thay thế toàn bộ leo đồi SPR bằng leo đồi TBR. Nếu một lượt lặp tìm kiếm ở pha 2 không tìm được một cây có điểm số MP thấp hơn so với điểm số của  $T^{\text{best}}$ , thì lượt lặp sẽ được coi là unsuccessful (thất bại). Thuật toán duy trì biến  $n_{\text{unsuccess}}$  lưu số lượt lặp tìm kiếm liên tiếp unsuccessful (thất bại). MPBoot gốc dừng nếu  $n_{\text{unsuccess}}$  đạt  $n'$  (giá trị làm tròn lên tới số hàng trăm gần nhất của  $n$ ). Do tập lân cận của TBR lớn hơn, chúng tôi hiệu chỉnh giới hạn của  $n_{\text{unsuccess}}$  thành  $n' = 100$  giống với IQ-TREE [22].

**Thuật toán 3.4** — Thuật toán MPBoot-TBR trong xấp xỉ bootstrap

<b>Input</b>	an MSA $A^{\text{data}}$ with $n$ sequences the number of bootstrap MSAs $B$ an upperbound for TBR radius $\text{maxtrav}$
<b>Output</b>	A tree $T^{\text{best}}$ with best found $\text{MP}(T^{\text{best}} \mid A^{\text{data}})$ and a set $\mathcal{B}$ of bootstrap trees $\{T_1, T_2, \dots, T_B\}$

- 1 **Phase 1: Initialization**
- 2   | Generate bootstrap MSAs and initialize bootstrap tree set  $\mathcal{B}$ .
- 3   | Initialize the threshold  $\text{MP}_{\text{max}} := +\infty$ .
- 4   | Initialize the candidate set  $C$  for  $A^{\text{data}}$  with 100 random stepwise addition procedures followed by TBR hill-climbing.
- 5
- 6 **Phase 2: Exploration**
- 7   | do

```

8   Improve  $C$  by performing perturbation on a randomly selected
9   tree from the candidate set  $C$  and a subsequent TBR hill-
10  climbing step.
11  if a new tree  $T$  with  $\text{MP}(T \mid A^{\text{data}}) < \text{MP}_{\max}$  is found then
12    Execute REPS to update the bootstrap tree set  $\mathcal{B}$ .
13  end if
14  Update  $T^{\text{best}}$ ,  $\text{MP}_{\max}$ ,  $n_{\text{unsuccess}}$ .
15  while  $n_{\text{unsuccess}} < n'$ 
16  Phase 3: Bootstrap Refinement
17  for each MP-tree  $T_b$  in  $\mathcal{B}$  do
18    Perform TBR hill-climbing search and replace  $T_b$  if a better
19    parsimony score is found.
20  end for
21  Output  $T^{\text{best}}$ , the best MP tree that was found for  $A^{\text{data}}$ .
22  Output set  $\mathcal{B}$  and/or map the support values onto  $T^{\text{best}}$ .

```

# Chương 4

## Đề xuất phương pháp MPBoot2

### 4.1. Các thay đổi so với MPBoot

Chúng tôi cài đặt MPBoot2 bằng ngôn ngữ C/C++ dưới dạng một phần mềm dòng lệnh mã nguồn mở, dựa trên mã nguồn công bố của MPBoot và cấu trúc cây của thư viện PLL [23]. MPBoot2 được công khai mã nguồn tại địa chỉ Github <https://github.com/HynDuf/mpboot/tree/mpboot2>.

MPBoot2 đã được sửa nhiều lỗi và tối ưu hóa đáng kể so với phiên bản 1, dẫn đến cải thiện khả năng tìm kiếm cây tối ưu hóa tối đa parsimony ngay khi sử dụng. Ngoài ra, MPBoot2 đã được nâng cấp đáng kể để hỗ trợ đầy đủ thao tác tree bisection and reconnection (TBR). Dựa trên phương pháp MPBoot-TBR mô tả ở [Chương 3](#), chúng tôi đã tích hợp TBR vào MPBoot hỗ trợ tính toán cho cả chi phí đồng nhất và không đồng nhất, sử dụng thuật toán Fitch [12] và Sankoff [13].

Cú pháp dòng lệnh để chạy MPBoot2 với các thao tác TBR trên MSA gốc được chỉ định bởi `<alignment_file>` như sau:

```
$ mpboot -s <alignment_file> -tbr_pars
```

Chiến lược tìm kiếm mặc định cho các thao tác TBR là chiến lược “tốt nhất”, trong đó đánh giá tất cả các cặp cạnh thêm vào  $I_1, I_2$  có thể xảy ra cho mỗi cạnh cắt  $R$  và chọn cặp nhánh mang lại cải tiến tốt nhất. Một chiến lược tìm kiếm khác là chiến lược “tốt hơn”, đánh giá các cạnh thêm  $I_2$  tốt nhất trên cây con còn lại cho mỗi cạnh cắt  $R$  và cạnh thêm đầu tiên  $I_1$ . Để sử dụng chiến lược này, thêm tùy chọn “`-tbr_better`” vào dòng lệnh. Để thực hiện tìm kiếm với điều kiện dừng 100 vòng lặp, sử dụng tùy chọn “`-stop_cond 100`”.

Một tính năng nổi bật khác được bổ sung vào MPBoot2 là tính năng checkpoint, cho phép chương trình tiếp tục từ checkpoint cuối cùng đã lưu trong trường hợp bị gián đoạn hoặc gặp sự cố. Tính năng này giúp tránh phải chạy lại toàn bộ phân tích, từ đó tiết kiệm thời gian và tài nguyên tính toán. Hệ thống checkpoint tương thích với cả thuật toán tìm kiếm cây MP tiêu chuẩn và phương pháp xấp xỉ bootstrap. Để kích hoạt tính năng này, thêm tùy chọn “`-ckp`”. Các tùy chọn mở rộng bao gồm “`-ckp_all`”, lưu tất cả các checkpoint thay vì chỉ lưu checkpoint gần nhất; “`-ckp_rerun`”, bỏ qua các checkpoint hiện có và chạy lại phân tích từ đầu; và “`-ckptime <checkpoint_time_interval_in_seconds>`”, đặt khoảng thời gian tối thiểu giữa các lần xuất dữ liệu checkpoint.

Ngoài ra, các cải tiến khác đã được thực hiện để nâng cao hiệu suất của MPBoot2 khi làm việc với các tập dữ liệu lớn (tự động phát hiện dữ liệu lớn), tối ưu hóa quản lý bộ nhớ và tốc độ xử lý để xử lý các phân tích phả hệ lớn hơn và phức tạp hơn.

Hơn nữa, MPBoot2 hiện hỗ trợ tốt hơn cho nhiều loại dữ liệu khác nhau. Nhiều lỗi đã được sửa để cải thiện khả năng xử lý dữ liệu morphology (thêm tùy chọn “`-st MORPH`”) và dữ liệu nhị phân (thêm tùy chọn “`-st BIN`”), đảm bảo các phân tích chính xác và đáng tin cậy hơn trên các định dạng tập dữ liệu khác nhau.

## 4.2. Đánh giá thực nghiệm trên các bộ dữ liệu vừa và nhỏ

### 4.2.1. Dữ liệu

Để đánh giá MPBoot2 trên các bộ dữ liệu thực tế có kích thước nhỏ và trung bình, chúng tôi sử dụng dữ liệu từ TreeBASE [24], như đã được phân tích trước đây bởi [25]. Bộ dữ liệu này bao gồm 115 sắp hàng đa chuỗi (MSA), trong đó 70 MSAs là loại DNA, có kích thước từ 200 đến hơn 700 dãy. 45 MSAs còn lại bao gồm các chuỗi protein (axit amin), với số lượng chuỗi trong sắp hàng dao động từ 50 đến gần 200.

Đối với bộ dữ liệu đánh giá độ chính xác bootstrap, chúng tôi tạo ra các bộ dữ liệu mô phỏng bằng công cụ Seq-gen. Bộ dữ liệu này bao gồm ba bộ dữ liệu DNA và hai bộ dữ liệu protein, mỗi loại chứa 200 MSAs. Các bộ dữ liệu được tạo ra bằng mô hình Yule-Harding để mô phỏng các mối quan hệ tiến hóa.

Ngoài ra, chúng tôi thử nghiệm MPBoot2 trên 30 bộ dữ liệu morphology, cũng được lấy từ cơ sở dữ liệu TreeBASE [24].

Cuối cùng, chúng tôi đánh giá MPBoot2 bằng cách sử dụng một bộ dữ liệu nhị phân, được lấy từ nghiên cứu của [26].

#### 4.2.2. Cài đặt thực nghiệm

Chúng tôi so sánh phiên bản SPR gốc (MPBoot phiên bản 1), kí hiệu là SPR6 (leo đồi SPR với bán kính 6), với một số phương pháp từ MPBoot2: TBR5 và TBR6 (leo đồi TBR với bán kính lần lượt là 5 và 6, sử dụng chiến lược mặc định tìm kiếm “tốt nhất”), TBR5-SC100 (TBR5 với điều kiện dừng ngắn hơn là 100 lần lặp không thành công), và TBR5-BETTER (TBR5 sử dụng chiến lược tìm kiếm “tốt hơn”). Tất cả các phương pháp này giữ lại một cây tốt nhất duy nhất cho mỗi bản sao bootstrap. Đối với ma trận chi phí, chúng tôi sử dụng cả ma trận chi phí đồng nhất và không đồng nhất, như được mô tả trong [9]. Việc phá cây bằng parsimony ratchet trong những iteration chẵn được dùng như nhau trong tất cả các phương pháp.

Đối với TNT (phiên bản 1.6, tháng 11 năm 2023), chúng tôi sử dụng chiến lược tìm kiếm chuyên sâu của phần mềm. Cụ thể, chúng tôi áp dụng lệnh “`xmult = notarget hits 3 level 0 chklevel +1 1`” cho MSA gốc, kết hợp nhiều chiến lược tìm kiếm khác nhau, bao gồm ratchet, tree fusing, sectorial search và tree drifting [15]. Sau đó, chúng tôi sử dụng lệnh “`mult = rep 1 hold 1`” trên các bootstrap MSAs, thực hiện việc thêm ngẫu nhiên theo từng bước, sau đó sử dụng leo đồi TBR đầy đủ.

Thực nghiệm được thực hiện cho cả dữ liệu mô phỏng và dữ liệu sinh học trên hệ thống tính toán hiệu năng cao của Trường Đại học Công Nghệ, ĐHQGHN.

#### 4.2.3. Tiêu chí đánh giá

##### 4.2.3.1. Điểm MP

Chúng tôi trước tiên so sánh các thuật toán theo điểm MP của cây  $T^{\text{best}}$ . Cụ thể, với phương pháp  $X$  và dataset  $Y$ , chúng tôi tính số lượng bộ dữ liệu trong  $Y$  mà phương pháp  $X$  đạt được điểm số tốt nhất trong số các phương pháp khảo sát.

#### 4.2.3.2. Độ chính xác bootstrap

Chúng tôi cũng so sánh chất lượng tập  $\mathcal{B}$  các cây bootstrap của các thuật toán nhờ tính toán độ chính xác bootstrap trên kết quả phân tích dữ liệu mô phỏng. Độ chính xác của một phương pháp bootstrap,  $Z$ , được định nghĩa bởi  $f_Z(v)$ , là tỷ lệ của số cạnh có mặt trong cây đúng trong số tất cả các cạnh có giá trị hỗ trợ bootstrap  $v\%$  (đếm trên các cây  $T^{\text{best}}$ ) [18].

#### 4.2.3.3. Thời gian thực thi

Với thước đo thứ 3 là thời gian thực thi, chúng tôi quan sát tổng thời gian (giờ) của việc phân tích bootstrap trên các MSAs.

### 4.2.4. Kết quả

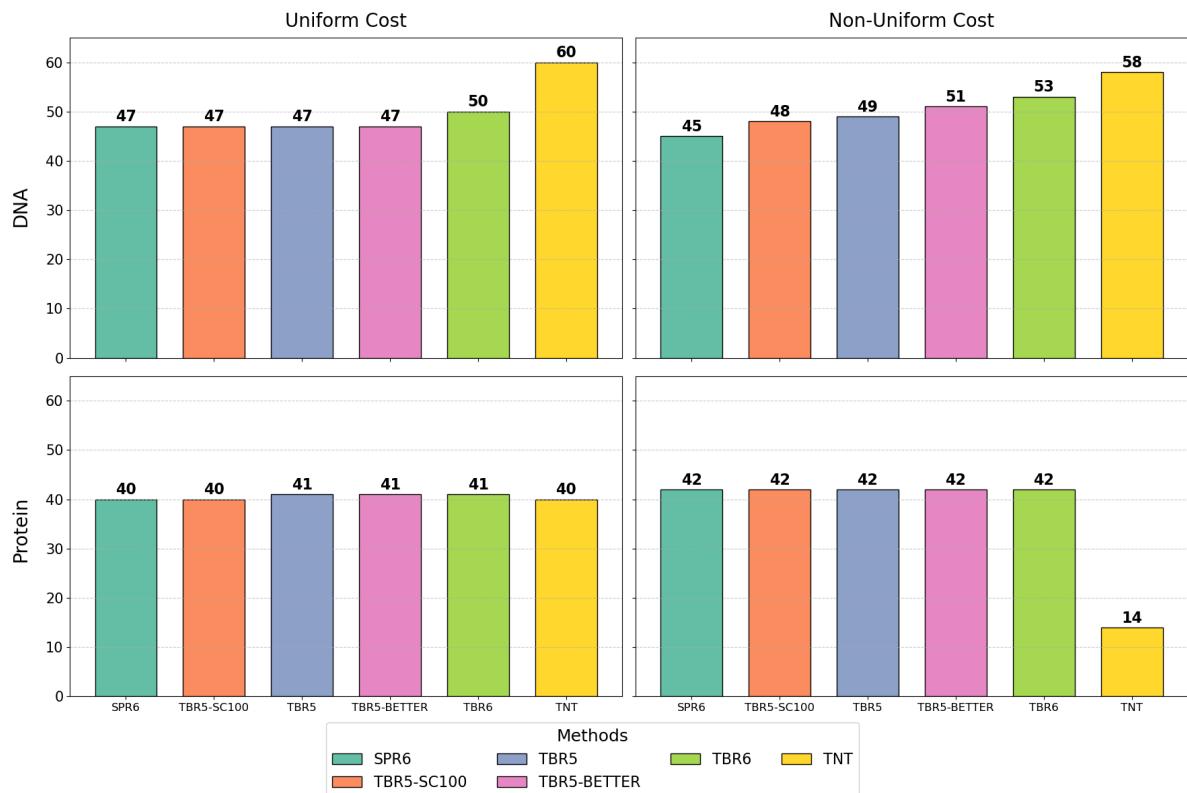
#### 4.2.4.1. Điểm MP

Như mô tả trong [Hình 4.1](#), hiệu suất của các phương pháp khác nhau thay đổi đáng kể giữa dữ liệu DNA và protein dưới các điều kiện chi phí đồng nhất và không đồng nhất. Đối với dữ liệu DNA, dưới điều kiện chi phí đồng nhất, phương pháp TNT đạt kết quả tốt nhất, với 60 bộ dữ liệu đạt được điểm tối đa theo phương pháp đơn giản hóa tối ưu. Hiệu suất này được theo sau bởi TBR6 (50), TBR5-BETTER (47), và cả TBR5 và TBR5-SC100 (47), thể hiện hiệu quả tương tự như SPR6 (47). Dưới điều kiện chi phí không đồng nhất, TBR6 (53) và TBR5-BETTER (51) là các phương pháp có hiệu suất cao nhất, trong khi TNT có phần thấp hơn với 58 bộ dữ liệu. Các phương pháp khác như TBR5 (49) và TBR5-SC100 (48) cũng thể hiện hiệu suất mạnh mẽ, trong khi đó SPR6 chỉ đạt được 45 bộ dữ liệu.

Đối với dữ liệu protein, điều kiện chi phí đồng nhất cho thấy kết quả đồng đều hơn giữa các phương pháp. TBR5, TBR5-BETTER, TBR6 đạt được 41 bộ dữ liệu và SPR6, TBR5-SC100, TNT đạt được 40 bộ dữ liệu được điểm tốt nhất. Tuy nhiên, dưới điều kiện chi phí không đồng nhất, TBR6 (42) và TBR5-SC100 (42) là các phương pháp tốt hơn. Ngược lại, TNT cho thấy sự suy giảm đáng kể về hiệu suất, chỉ đạt được 14 bộ dữ liệu, thấp nhất trong tất cả các phương pháp. Các phương pháp khác, bao gồm TBR5 (42), SPR6 (42), và TBR5-BETTER (42), duy trì hiệu suất ổn định.

Những kết quả này cho thấy rõ sự thay đổi trong hiệu suất của các phương pháp phụ thuộc vào kiểu dữ liệu và điều kiện chi phí. Phương pháp TNT cho

thấy hiệu suất vượt trội với dữ liệu DNA dưới điều kiện chi phí đồng nhất, nhưng hiệu suất giảm mạnh đối với dữ liệu protein dưới điều kiện chi phí không đồng nhất. Mặt khác, các phương pháp TBR\* mang lại độ chính xác cao ổn định trên cả hai loại dữ liệu và điều kiện chi phí, làm nổi bật tính tin cậy và mạnh mẽ của chúng, đặc biệt trong các tình huống yêu cầu hiệu suất ổn định trên các điều kiện khác nhau.



**Hình 4.1** — Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu DNA và protein từ TreeBASE. Các biểu đồ cột thể hiện số lượng bộ dữ liệu (trong tổng số 115 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát.

#### 4.2.4.2. Thời gian thực thi

Tổng thời gian chạy (tính bằng giờ) của mỗi phương pháp trên 115 bộ dữ liệu từ TreeBASE được trình bày trong [Bảng 4.1](#), so sánh hiệu suất trong các điều kiện chi phí đồng nhất và không đồng nhất. Ngoài ra, bảng còn cung cấp tỷ lệ thời gian trung bình và trung vị so với SPR6 cho cả hai điều kiện chi phí.

Trong điều kiện chi phí đồng nhất, phương pháp nhanh nhất là TBR5-SC100 (30,2 giờ), tiếp theo là SPR6 (37,2 giờ). Các phương pháp chậm nhất là

TBR6 (78,2 giờ) và TNT (75,5 giờ). Tỷ lệ thời gian trung bình của các phương pháp so với SPR6 trong điều kiện chi phí đồng nhất là 1.04 cho TBR5-SC100, 1.42 cho TBR5, 1.62 cho TBR5-BETTER, và 1.98 cho TBR6, với giá trị trung vị lần lượt là 0.96, 1.43, 1.60, và 1.96. TNT có tỷ lệ thời gian trung bình là 1.23 và trung vị là 0.47.

Trong điều kiện chi phí không đồng nhất, SPR6 vẫn là phương pháp hiệu quả nhất với thời gian chạy là 146.8 giờ, vượt trội hơn TBR5-SC100 (192.8 giờ) và TBR5 (240.1 giờ). Các phương pháp chậm nhất là TBR6 (338.8 giờ) và TNT (682.3 giờ). Tỷ lệ thời gian trung bình trong điều kiện chi phí không đồng nhất là 1.26 cho TBR5-SC100, 1.52 cho TBR5, 1.72 cho TBR5-BETTER, và 2.21 cho TBR6, với giá trị trung vị lần lượt là 1.17, 1.52, 1.71, và 2.18. TNT cho thấy tỷ lệ thời gian cao đáng kể với giá trị trung bình là 6.47 và trung vị là 3.55.

Đáng chú ý, TBR5-SC100 rất hiệu quả trong điều kiện chi phí đồng nhất nhưng kém cạnh tranh hơn trong điều kiện không đồng nhất, như được thể hiện qua cả thời gian chạy và tỷ lệ thời gian so với SPR6.

Tóm lại, TBR5-SC100 và SPR6 là hai phương pháp cân bằng, vừa nhanh vừa cho kết quả đủ tốt. Trong điều kiện chi phí đồng nhất, TBR5-SC100 chạy nhanh hơn SPR6, trong khi ở điều kiện chi phí không đồng nhất, TBR5-SC100 chạy chậm hơn một chút nhưng đạt được điểm MP tốt hơn so với SPR6. Chi tiết phân tích về điểm số và thời gian giữa các phương pháp được trình bày bổ sung ở [Phụ lục A](#).

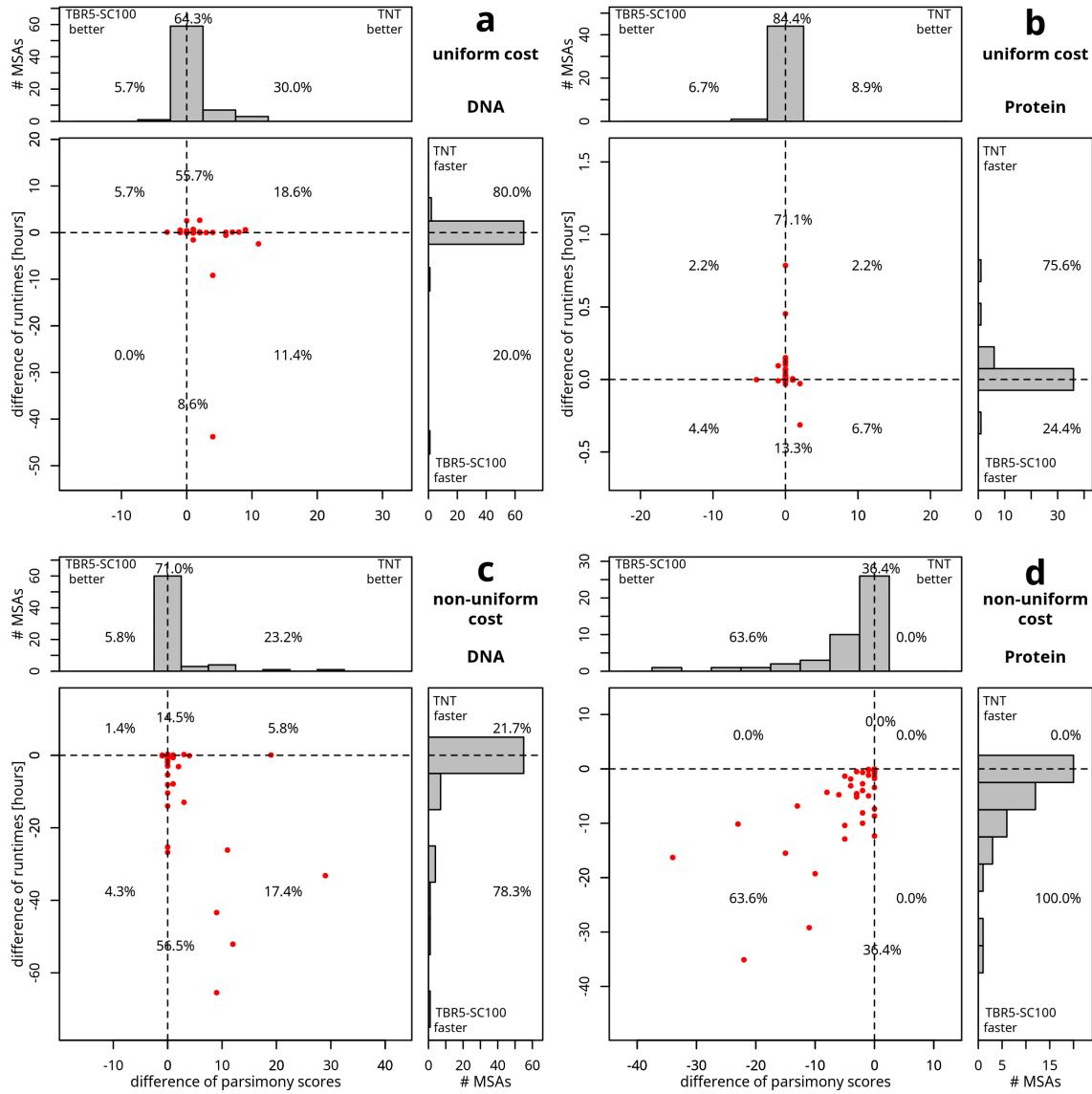
**Bảng 4.1** — Thời gian chạy tổng cộng (giờ) và tỷ lệ thời gian (so với SPR6) của các phương pháp trên 115 bộ dữ liệu từ bộ dữ liệu TreeBASE

Method	Uniform cost			Non-uniform cost		
	Time (hours)	Mean	Median	Time (hours)	Mean	Median
SPR6	37.2	1.00	1.00	<b>146.8</b>	1.00	1.00
TBR5-SC100	<b>30.2</b>	1.04	0.96	192.8	1.26	1.17
TBR5	54.5	1.42	1.43	240.1	1.52	1.52
TBR5-BETTER	67.3	1.62	1.60	269.3	1.72	1.71
TBR6	78.2	1.98	1.96	338.8	2.21	2.18
TNT	75.5	1.23	0.47	682.3	6.47	3.55

Các thông tin trong [Hình 4.1](#) và [Bảng 4.1](#) chỉ cung cấp thống kê tổng quan về các phương pháp, nên chúng tôi so sánh TBR5-SC100 (phương pháp cân bằng nhất) và TNT toàn diện hơn trong [Hình 4.2](#). Mỗi điểm trong hình biểu thị một bộ dữ liệu (trong tổng số 115 bộ), với trục ngang biểu thị sự chênh lệch điểm số parsimony tối đa và trục dọc biểu thị sự chênh lệch thời gian chạy. Các biểu đồ cột được đặt ở phía trên và bên cạnh cung cấp tầm suât biến. Các điểm nằm bên trái đường đứt nét dọc biểu thị các căn chỉnh mà TBR5-SC100 đạt được điểm số parsimony tốt hơn. Các điểm nằm bên dưới đường đứt nét ngang biểu thị các trường hợp mà phân tích bootstrap của TBR5-SC100 nhanh hơn. Các tỷ lệ phần trăm trong các vùng của biểu đồ cột biểu thị tỷ lệ các điểm dữ liệu trong các khu vực đó. Các tỷ lệ phần trăm dọc theo đường đứt nét biểu thị tỷ lệ các sấp hàng mà cả hai phương pháp đều đạt được cùng một điểm số MP.

Đối với các bộ dữ liệu DNA (xem [Hình 4.2a](#) và [c](#)), TNT cho thấy điểm số MP tốt hơn trong điều kiện chi phí đồng nhất (30% so với 5.7%) và trong điều kiện chi phí không đồng nhất (22.1% so với 5.9%). Tuy nhiên, đối với các bộ dữ liệu protein ([Hình 4.2b](#) và [d](#)), TBR5-SC100 vượt trội hơn TNT trong điều kiện chi phí không đồng nhất (63.6% so với 0%) và kém hơn một chút so với TNT trong điều kiện chi phí đồng nhất (6.7% so với 8.9%).

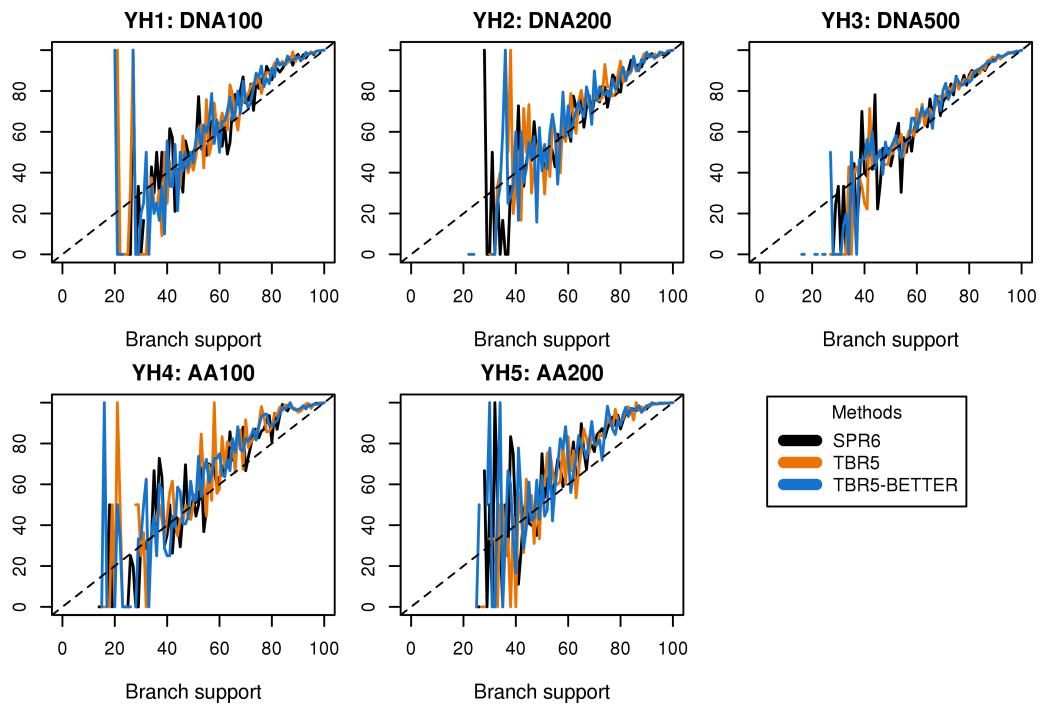
Về thời gian chạy, trong điều kiện chi phí không đồng nhất (xem [Hình 4.2c](#) và [d](#)), TBR5-SC100 chạy nhanh hơn đáng kể so với TNT (77.9% so với 22.1% cho các bộ dữ liệu DNA và 100% so với 0% cho các bộ protein). Tuy nhiên, trong điều kiện chi phí đồng nhất, TNT nhanh hơn so với TBR5-SC100 (xem [Hình 4.2a](#) và [b](#)).



**Hình 4.2** — So sánh TBR5-SC100 và TNT trên chi phí đồng nhất (**a, b**) và chi phí không đồng nhất (**c, d**) trên các MSAs DNA và protein từ TreeBASE.

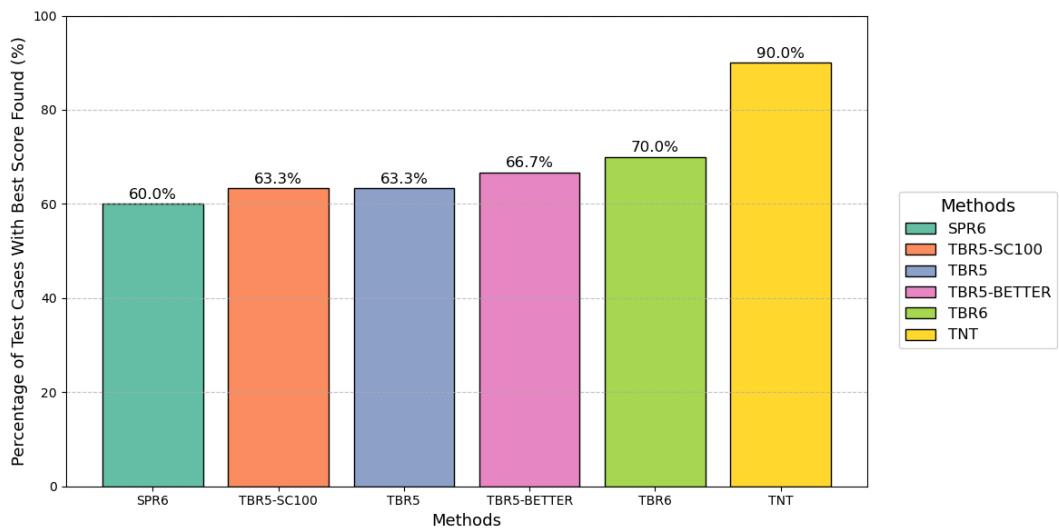
#### 4.2.4.3. Độ chính xác bootstrap

Hàm  $f_{\text{SPR}_6}(v)$  (đường màu đen), hàm  $f_{\text{TBR}5}(v)$  (đường màu cam) và hàm  $f_{\text{TBR}5\text{-BETTER}}(v)$  (đường màu xanh) cho 5 bộ YuleHarding được minh họa ở [Hình 4.3](#). Trong cả 5 đồ thị, 2 đường cong của 2 hàm này nằm sát nhau và cùng nằm phía trên đường chéo cho thấy phiên bản mới cho độ chính xác bootstrap tương đương MPBoot.



**Hình 4.3** — Độ chính xác bootstrap của các phương pháp TBR5 (đường màu cam), TBR5-BETTER (đường màu xanh) và SPR6 (đường màu đen - phiên bản MPBoot cũ)

#### 4.2.4.4. Dữ liệu morphology và nhị phân



**Hình 4.4** — Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu morphology từ TreeBASE. Các biểu đồ cột thể hiện tỉ lệ số bộ dữ liệu (trong tổng số 30 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát

**Bảng 4.2** — Thời gian chạy tổng cộng (giờ) của các phương pháp trên 30 bộ dữ liệu morphology

Method	Runtime (hours)
SPR6	1.9
TBR5-SC100	<b>1.6</b>
TBR5	2.4
TBR5-BETTER	3.1
TBR6	3.5
TNT	6.9

Trước đây, phiên bản MPBoot không hỗ trợ dữ liệu morphology. Trong phiên bản MPBoot2, tính năng này đã được hỗ trợ chính thức. Hiệu suất được trình bày trong [Hình 4.4](#) và [Bảng 4.2](#). Mặc dù tất cả các phương pháp trong MPBoot2 không hiệu quả bằng TNT về điểm MP, nhưng lại nhanh hơn đáng kể.

Đối với dữ liệu nhị phân, MPBoot2 hiện cũng đã hỗ trợ. Khi chạy trên một bộ dữ liệu duy nhất, tất cả các phương pháp đều đạt được điểm tốt nhất là 1,847,943. NNI mất 144 giây, SPR6 mất 2,074 giây, TBR5 mất 3,264 giây, TBR5-BETTER mất 3,159 giây, và TNT mất 1,217 giây.

#### 4.2.5. Phân tích về hiệu năng của TBR5 so với SPR6

Trong [Hình 4.5](#) là phần phân tích kỹ càng hơn về hai phương pháp TBR5 và SPR6. Có thể thấy rằng, nhìn chung, TBR5 cho điểm MP tốt hơn SPR6, nhưng SPR6 lại chạy nhanh hơn đáng kể. Tuy nhiên, khi phân tích chi tiết hơn (xem [Bảng 4.3](#)) trên 115 bộ dữ liệu TreeBASE, xét trong hai trường hợp:

Đối với chi phí đồng nhất và chi phí không đồng nhất:

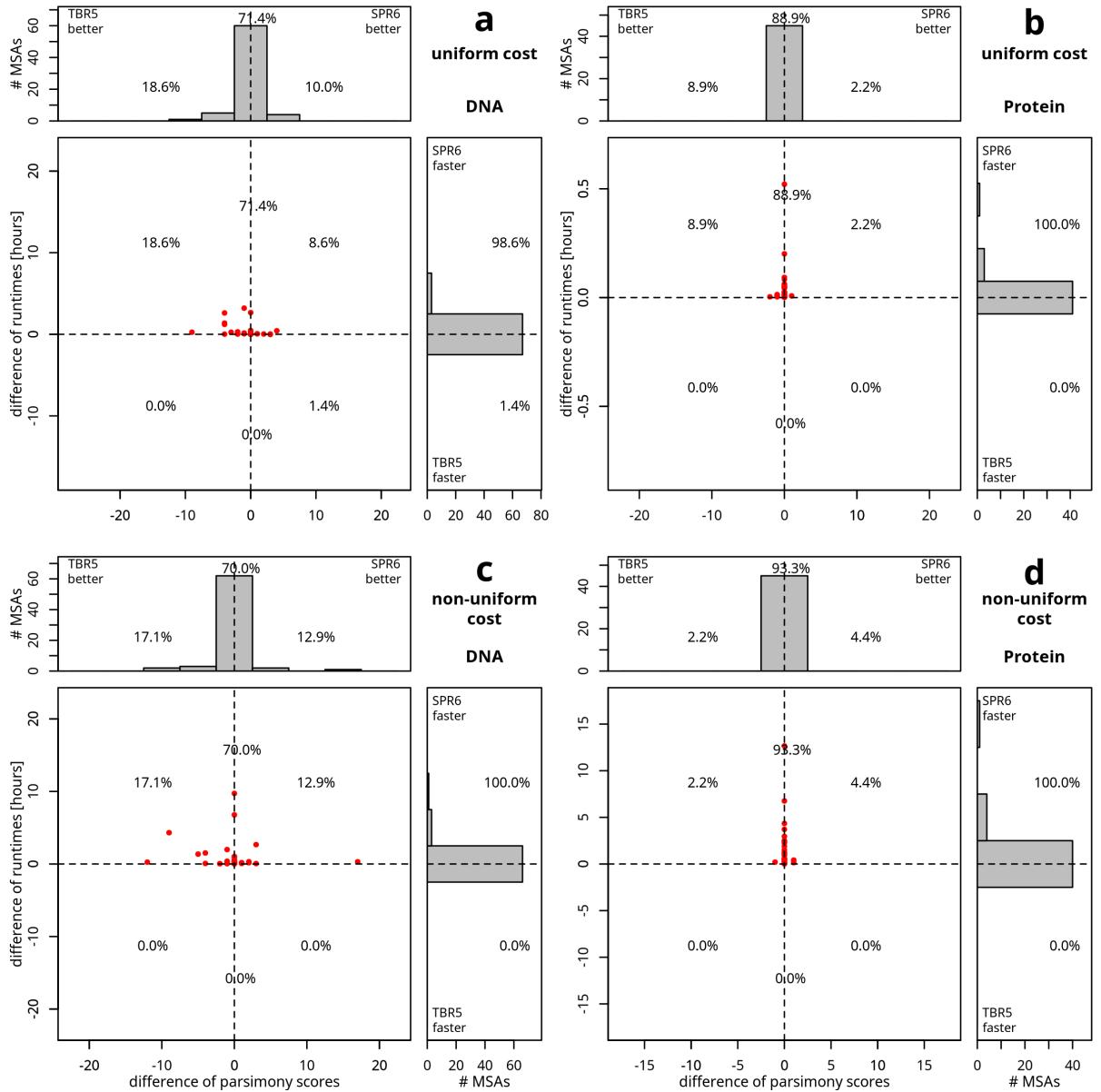
- (1) Những bộ dữ liệu mà TNT tốt hơn SPR6, nhưng TBR5 lại tốt hơn hoặc bằng TNT.
- (2) Những bộ dữ liệu mà TNT tốt hơn TBR5, nhưng SPR6 lại tốt hơn hoặc bằng TNT.

**Bảng 4.3** — Phân tích các bộ dữ liệu mà SPR6 hoặc TBR5 vươn lên so với phương pháp còn lại

		Testcases	MP Score		
			SPR6	TBR5	TNT
Uniform Cost	(1)	dna_M7929_428_15016	91399	<b>91395</b>	91396
		dna_M7964_640_25260	256917	<b>256916</b>	<b>256916</b>
		dna_M4794_204_12113	34926	<b>34923</b>	<b>34923</b>
		prot_M11344_84_691	9230	<b>9228</b>	<b>9228</b>
	(2)	dna_M1838_228_1131	<b>20531</b>	20534	20532
		dna_M11745_316_1494	<b>3277</b>	3278	<b>3277</b>
		dna_M9143_228_1223	<b>1146</b>	1148	1147
		prot_M1118_137_348	<b>2153</b>	2154	<b>2153</b>
Non-uniform Cost	(1)	dna_M7929_428_15016	120458	<b>120457</b>	<b>120457</b>
		dna_M7024_767_5814	122834	<b>122825</b>	<b>122825</b>
		dna_M7292_213_7572	61697	<b>61696</b>	<b>61696</b>
		dna_M9033_300_1394	5955	<b>5954</b>	<b>5954</b>
		dna_M3031_276_1518	4232	<b>4228</b>	<b>4228</b>
	(2)	dna_M1224_210_8235	<b>80910</b>	80912	<b>80910</b>
		dna_M5931_298_4948	<b>16631</b>	16632	<b>16631</b>
		dna_M11745_316_1494	<b>4169</b>	4170	4169
		prot_M11338_100_567	<b>8219</b>	8220	<b>8219</b>

Nhận thấy ở cả hai trường hợp, số lượng bộ dữ liệu mà một phương pháp vượt trội so với phương pháp còn lại là tương đương nhau. Điều này gợi ý rằng SPR6 và TBR5 phù hợp tùy thuộc vào từng bộ dữ liệu cụ thể và mỗi phương

pháp có điểm mạnh riêng. Nếu có một cách kết hợp cả hai phép biến đổi này, rất có thể sẽ tận dụng được các điểm mạnh của cả hai phương pháp.



**Hình 4.5** — So sánh kết quả của TBR5 với SPR6 trên 115 bộ dữ liệu TreeBASE

### 4.3. Đánh giá thực nghiệm trên các bộ dữ liệu lớn

Chúng tôi sử dụng các bộ dữ liệu lớn, bao gồm Plant ( $38 \times 432014$ ), Bird ( $52 \times 4519041$ ), Insect ( $144 \times 383161$ ), và Mammal ( $90 \times 1848196$ ) được lấy từ [27]. Chúng tôi so sánh hiệu suất của một số phương pháp trên các bộ căn chỉnh chuỗi lớn (MSAs) này bằng cách sử dụng cả mô hình chi phí đồng nhất và không

đồng nhất. Kết quả được tóm tắt trong [Bảng 4.4](#) và [Bảng 4.5](#), trình bày điểm MP đạt được và thời gian chạy (tính bằng giờ) của từng phương pháp.

Như trình bày trong [Bảng 4.4](#), tất cả các phương pháp — NNI, SPR6, TBR5, và TNT — đều đạt được các điểm MP bằng nhau trên cả bốn bộ MSA (Plant, Bird, Insect, và Mammal). Các điểm số này cho thấy các phương pháp đều chính xác ngang nhau đối với tiêu chí Maximum Parsimony. Tuy nhiên, hiệu suất thời gian chạy lại khác biệt đáng kể. NNI luôn cho thấy thời gian chạy nhanh nhất trên tất cả các bộ dữ liệu, với thời gian từ 0.3 giờ đối với bộ dữ liệu Plant đến 5.7 giờ đối với bộ dữ liệu Bird. Ngược lại, TNT có thời gian thực thi lâu nhất, với bộ dữ liệu Bird mất 7.2 giờ, tiếp theo là Mammal, Insect, và Plant với lần lượt 6.0, 13.0, và 0.5 giờ. SPR6 và TBR5 có thời gian chạy trung bình, trong đó TBR5 thường mất nhiều thời gian hơn SPR6.

Đối với điều kiện chi phí không đồng nhất, như trình bày trong [Bảng 4.5](#), chúng tôi nhận thấy rằng các phương pháp khác mất quá nhiều thời gian để có thể so sánh, vì vậy chúng tôi chỉ trình bày kết quả cho NNI và TNT. NNI đạt được điểm MP tốt hơn TNT đối với cả hai bộ dữ liệu Plant và Insect, với điểm số lần lượt là 1,684,167 và 6,797,376, so với 1,684,262 và 6,797,627 của TNT. Mặc dù sự khác biệt về điểm số là rất nhỏ, NNI cũng vượt trội hơn TNT về thời gian chạy, hoàn thành phân tích bootstrap trong 81.2 giờ đối với Plant và 76.3 giờ đối với Insect. Trong khi đó, TNT chạy mất 112.9 giờ đối với Plant và lâu hơn nhiều đối với Insect (587.0 giờ), cho thấy một khoảng cách lớn về hiệu năng giữa hai phương pháp trong điều kiện chi phí không đồng nhất.

**Bảng 4.4 — Điểm MP và thời gian chạy (giờ) của các phương pháp trên các bộ dữ liệu lớn với điều kiện chi phí đồng nhất**

MSA	Maximum Parsimony Score				Runtime (hours)			
	NNI	SPR6	TBR5	TNT	NNI	SPR6	TBR5	TNT
Plant	1,520,763				<b>0.3</b>	1.7	2.4	0.5
Bird	7,378,469				<b>5.7</b>	49.3	67.1	7.2
Insect	5,651,669				<b>2.0</b>	24.0	34.1	13.0
Mammal	7,260,920				<b>3.8</b>	34.5	57.8	6.0

**Bảng 4.5** — Điểm MP và thời gian chạy (giờ) của các phương pháp trên các bộ dữ liệu lớn với điều kiện chi phí không đồng nhất

MSA	Maximum Parsimony Score		Runtime (hours)	
	NNI	TNT	NNI	TNT
Plant	<b>1,684,167</b>	1,684,262	<b>81.2</b>	112.9
Insect	<b>6,797,376</b>	6,797,627	<b>76.3</b>	587.0

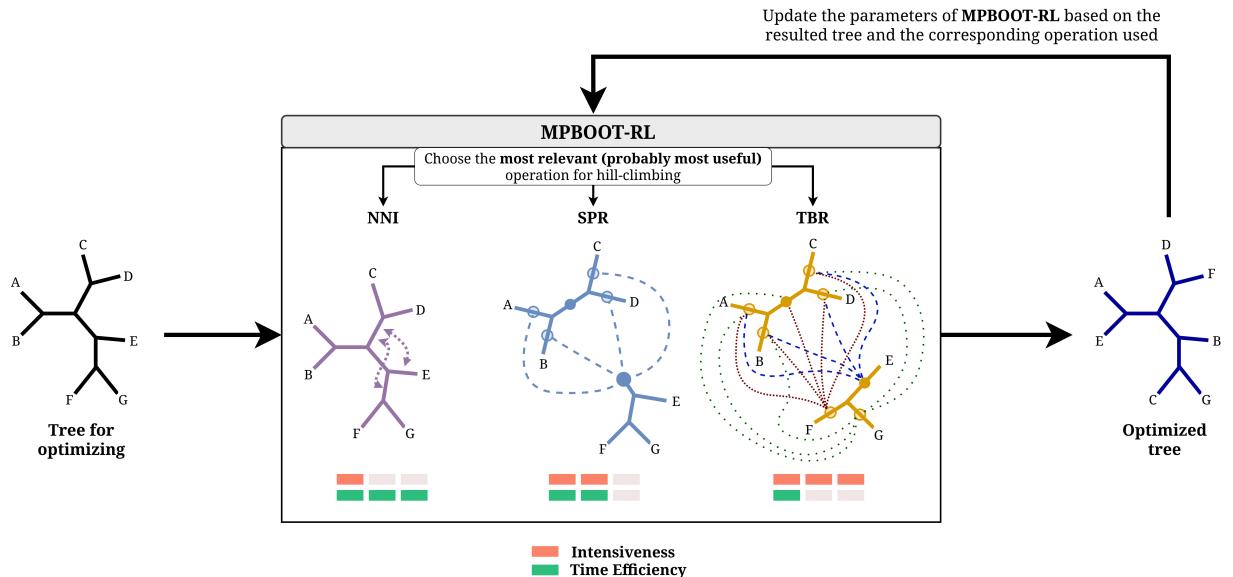
# Chương 5

## Đề xuất phương pháp MPBoot-RL

### 5.1. Áp dụng ACO vào MPBoot2

Như đã phân tích ở [Chương 4.2.5](#), dù leo đồi sử dụng TBR5 nhìn chung có cải thiện hơn so với sử dụng SPR6, nhưng vẫn có nhiều bộ dữ liệu mà SPR6 tỏ ra hiệu quả hơn so với TBR5. Không những thế, với những bộ dữ liệu dễ, sử dụng thuật toán NNI khi đó sẽ tăng tốc độ tìm kiếm đáng kể. Vì những lý do trên, việc kết hợp NNI, SPR và TBR có tiềm năng cải thiện hiệu suất thuật toán, cải thiện điểm số, và từ đó tăng cường độ chính xác của bootstrap. Kết hợp các phép biến đổi cây có thể được thực hiện ở bước leo đồi ở pha khám phá của MPBoot (xem [Hình 3.1](#)).

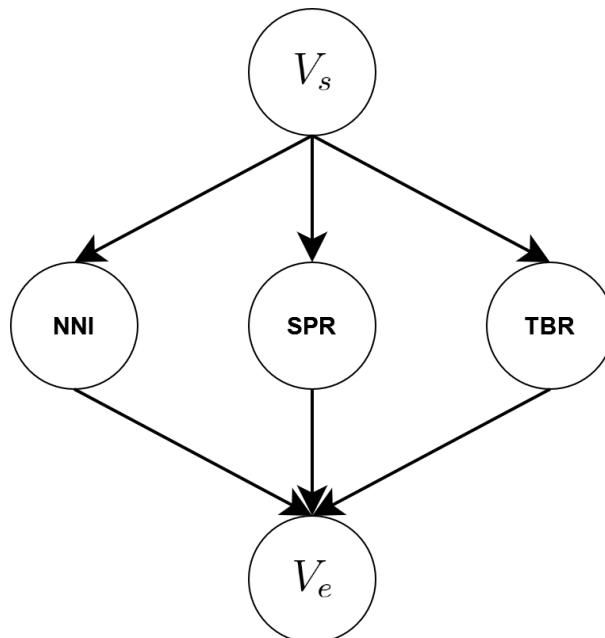
Chúng tôi đề xuất thuật toán MPBoot-RL để tăng cường pha khám phá (pha 2) của thuật toán MPBoot2 bằng cách sử dụng một phương pháp học tăng cường, cụ thể là giải thuật tối ưu đòn kién (ACO). Thuật toán MPBoot-RL thay thế quá trình tối ưu hóa cây, trước đây chỉ sử dụng leo đồi SPR, bằng việc chọn một trong ba phép toán biến đổi cây: NNI, SPR và TBR (xem [Hình 5.1](#)). Ngoài ra, chúng tôi thực hiện khảo sát liên quan giữa các thông số sử dụng các phép biến đổi cây với “độ khó” của tập dữ liệu (sử dụng Pythia [28]), gợi mở về cách dự đoán độ khó dữ liệu theo MPBoot2.



**Hình 5.1** — Minh họa cho khung thuật toán MPBoot-RL

### 5.1.1. Cấu trúc đồ thị

Cấu trúc đồ thị bao gồm một đỉnh khởi đầu  $V_s$ , một đỉnh kết thúc  $V_e$  và một lớp giữa. Lớp này bao gồm ba đỉnh biểu diễn việc sử dụng các thuật toán leo đồi NNI, SPR và TBR. Đỉnh khởi đầu  $V_s$  sẽ có các cạnh kết nối với ba đỉnh của lớp giữa. Cuối cùng, các đỉnh trong lớp sẽ kết nối với đỉnh kết thúc  $V_e$ . Tổng cộng, mạng bao gồm 5 đỉnh và 6 cạnh (xem [Hình 5.2](#)).



**Hình 5.2** — Cấu trúc đồ thị ACO

### 5.1.2. Thông tin heuristic

Thông tin heuristics được cung cấp dưới dạng các tham số đầu vào, bao gồm các chi tiết heuristics cho các thuật toán leo đồi NNI, SPR và TBR. Do đó, thông tin heuristics cho một bước đi của con kiến sẽ được trích xuất tương ứng với đỉnh đích, cụ thể là NNI, SPR hoặc TBR.

### 5.1.3. Vết mùi pheromone và các quy tắc cập nhật

Thông tin về vết mùi pheromone được đặt trên mỗi cạnh của cấu trúc đồ thị và được cập nhật theo các quy tắc cụ thể được sử dụng.

#### 5.1.3.1. Quy tắc SMMAS

Quy tắc cập nhật của vết pheromone với hệ thống kiến SMMA (Smooth Max-Min Ant System) [21] có thể được tổng kết như sau. Ban đầu, pheromone trên tất cả các cạnh được giảm ('bốc hơi') theo tỷ lệ  $\rho$  ( $0 < \rho < 1$ ). Sau đó, nếu một cạnh thuộc giải pháp tốt nhất được tìm thấy trong thế hệ hiện tại, mức độ pheromone được cập nhật bổ sung tiến gần đến một giá trị tối đa  $\tau_{\max}$ ; nếu không, nó được điều chỉnh về một giá trị tối thiểu  $\tau_{\min}$ .

Chúng tôi đề xuất áp dụng quy tắc SMMAS vào thuật toán MPBoot-ML như sau. Trong mỗi thế hệ kiến của thuật toán MPBoot-RL, quá trình cập nhật bao gồm một quy trình lựa chọn cẩn thận các giải pháp tốt nhất nhằm tối đa hóa điểm MP. Thuật toán xác định các con kiến có giải pháp tìm ra một cây có điểm MP tốt hơn hoặc bằng với điểm MP tốt nhất hiện tại đã tìm được của thuật toán. Những giải pháp này được coi là đáng kỳ vọng và được chọn cho quá trình cập nhật tiếp theo. Tuy nhiên, nếu không có giải pháp nào đáp ứng tiêu chí này, phép biến đổi cây nhanh nhất (phép NNI) sẽ được chọn.

Sau khi một thế hệ kiến hoàn thành, thuật toán tiến hành cập nhật vết mùi pheromone trên các cạnh. Tất cả các cạnh tồn tại trong ít nhất một trong các giải pháp tốt được lựa chọn được cập nhật lên mức pheromone tối đa  $\tau_{\max}$ , phản ánh sự đóng góp của chúng vào các giải pháp có triển vọng.

$$\tau_e \leftarrow (1 - \rho) \cdot \tau_e + \rho \cdot \tau_{\max}$$

Ngược lại, các cạnh không thuộc bất kỳ giải pháp được lựa chọn nào sẽ được cập nhật xuống mức pheromone tối thiểu  $\tau_{\min}$ .

$$\tau_e \leftarrow (1 - \rho) \cdot \tau_e + \rho \cdot \tau_{\min}$$

Quá trình cập nhật này cung cấp các đường đi liên quan đến các giải pháp tốt và “né tránh” những đường đi không đóng góp đáng kể vào việc cải thiện điểm MP.

### 5.1.3.2. Quy tắc thử nghiệm SMMAS-multiple

Quy tắc thử nghiệm SMMAS-multiple đề xuất thêm một sự thay đổi cho mô hình cập nhật vết mùi pheromone đã được mô tả trước đó trong thuật toán MPBoot-ACO. Khác với quy tắc SMMAS ban đầu, nơi mỗi cạnh được cập nhật một lần duy nhất trong một thế hệ kiến, quy tắc SMMAS-multiple được thiết kế để ưu tiên cạnh mà tham gia vào **nhiều** (thay vì chỉ một) giải pháp tốt.

Theo quy tắc SMMAS-multiple, tiêu chí lựa chọn để xác định các giải pháp triển vọng vẫn được giữ nguyên. Tuy nhiên, khác với quy tắc SMMAS gốc, cách tiếp cận này cho phép cạnh tham gia trong nhiều giải pháp được chọn được cập nhật nhiều lần lên mức pheromone tối đa. Cụ thể, nếu một cạnh đóng góp  $k$  lần trong các giải pháp được chọn, nó sẽ được cập nhật  $k$  lần để đạt đến mức pheromone tối đa  $\tau_{\max}$ .

$$\tau_e \leftarrow (1 - \rho) \cdot \tau_e + \rho \cdot \tau_{\max} \quad (\text{lặp lại } k \text{ lần})$$

Các cạnh không đóng góp vào bất kỳ giải pháp được chọn nào vẫn tuân theo quy tắc cập nhật SMMAS gốc, được cập nhật một lần đến mức pheromone tối thiểu  $\tau_{\min}$ .

Bằng cách cho phép các cạnh tích lũy cập nhật dựa trên sự đóng góp của chúng trong các giải pháp được chọn, cơ chế cập nhật SMMAS-multiple cung cấp sự ảnh hưởng của các cạnh liên quan đến các giải pháp đã thể hiện thành công liên tục qua nhiều vòng lặp.

### 5.1.4. Quy trình bước đi ngẫu nhiên để xây dựng giải pháp

Con kiến di chuyển tuần tự từ  $V_s$  đến lớp đầu tiên và sau đó trở lại  $V_e$ . Giả sử con kiến đang ở đỉnh  $A$ , nó sẽ di chuyển ngẫu nhiên dọc theo một cạnh đến một đỉnh trong lớp tiếp theo. Xác suất cho con kiến tại đỉnh  $A$  di chuyển đến đỉnh  $B$  tỉ lệ phụ thuộc vào mức độ của vết mùi pheromone và thông tin heuristic của đường đi  $A \rightarrow B$ .

$$\text{prob}_{A \rightarrow B} = \frac{\tau_{A \rightarrow B} \cdot \eta_B}{\sum_{C \in \text{adj}(A)} \tau_{A \rightarrow C} \cdot \eta_C}$$

trong đó:

- $\text{prob}_{A \rightarrow B}$  là xác suất con kiến ở đỉnh  $A$  di chuyển đến đỉnh  $B$ .
- $\tau_{A \rightarrow B}$  là mức độ pheromone của cạnh từ  $A$  đến  $B$ .
- $\eta_B$  là thông tin heuristic của đỉnh  $B$ .
- $\text{adj}(A)$  là tập các đỉnh mà  $A$  nối tới.

### 5.1.5. Cài đặt

MPBoot-RL được triển khai bằng C/C++ như một chương trình dòng lệnh mã nguồn mở, dựa trên mã nguồn mở của MPBoot2. Dòng lệnh để chạy tìm kiếm cây đơn giản của MPBoot-RL với chiến lược SMMAS-multiple (hay ACO-MUL) trên MSA gốc được chỉ định bởi “`<alignment_file>`”, có cú pháp như sau:

```
$ mpboot -s <alignment_file> -aco
```

Cú pháp để tìm kiếm với chiến lược SMMAS gốc (hay ACO-ONCE) là:

```
$ mpboot -s <alignment_file> -aco -aco_once
```

Cú pháp đầy đủ để tìm kiếm với các tham số tùy chỉnh:

```
$ mpboot -s <alignment_file> -aco -aco_nni_prior 0.3 -aco_spr_prior 0.4  
-aco_tbr_prior 0.4 -aco_evaporation_rate 0.25 -aco_update_iter 15
```

Để thực hiện MP bootstrapping với  $B$  sấp hàng bootstrap, thêm “`-bb <B>`” vào dòng lệnh.

## 5.2. Đánh giá thực nghiệm

### 5.2.1. Cài đặt thực nghiệm

Chúng tôi so sánh hai phiên bản của MPBoot-RL (ACO-MUL (sử dụng quy tắc cập nhật mì SMMAS-multiple) và ACO-ONCE (sử dụng quy tắc cập nhật mì SMMAS gốc)) với các phương pháp được đề cập ở [Chương 4.2.2](#). Các phiên bản MPBoot-RL sử dụng SPR với bán kính 6 (SPR6) và TBR với bán kính 5 (TBR5). Chúng tôi thử nghiệm hai phiên bản trên với nhiều tập siêu tham số khác nhau nhưng chỉ trình bày ở đây bộ siêu tham số cho kết quả tốt nhất đối với từng phiên bản. Tổng hợp kết quả của các tập siêu tham số khác nhau được trình bày ở [Phụ lục C](#).

Đối với ACO-MUL:

- $\tau_{\max} = 1.0$ ,  $\tau_{\min} = 0.1$

- Độ bay hơi:  $\rho = 0.25$
- Thông tin heuristic:  $\eta_{NNI} = 0.3$ ,  $\eta_{SPR} = 0.4$ ,  $\eta_{TBR} = 0.4$
- Số lượng kiến ở mỗi thẻ hệ:  $L = L_0 + \lceil \frac{n}{100} \rceil$  với  $L_0 = 15$  trong đó  $n$  là số lượng taxa của sáp hàng gốc. Để thấy, số kiến được điều chỉnh động dựa trên kích thước của MSA đầu vào.

Đối với ACO-ONCE:

- $\tau_{\max} = 1.0$ ,  $\tau_{\min} = 0.1$
- Độ bay hơi:  $\rho = 0.1$
- Thông tin heuristic:  $\eta_{NNI} = 0.3$ ,  $\eta_{SPR} = 0.4$ ,  $\eta_{TBR} = 0.4$
- Số lượng kiến ở mỗi thẻ hệ:  $L = L_0 + \lceil \frac{n}{100} \rceil$  với  $L_0 = 5$  trong đó  $n$  là số lượng taxa của sáp hàng gốc. Để thấy, số kiến được điều chỉnh động dựa trên kích thước của MSA đầu vào.

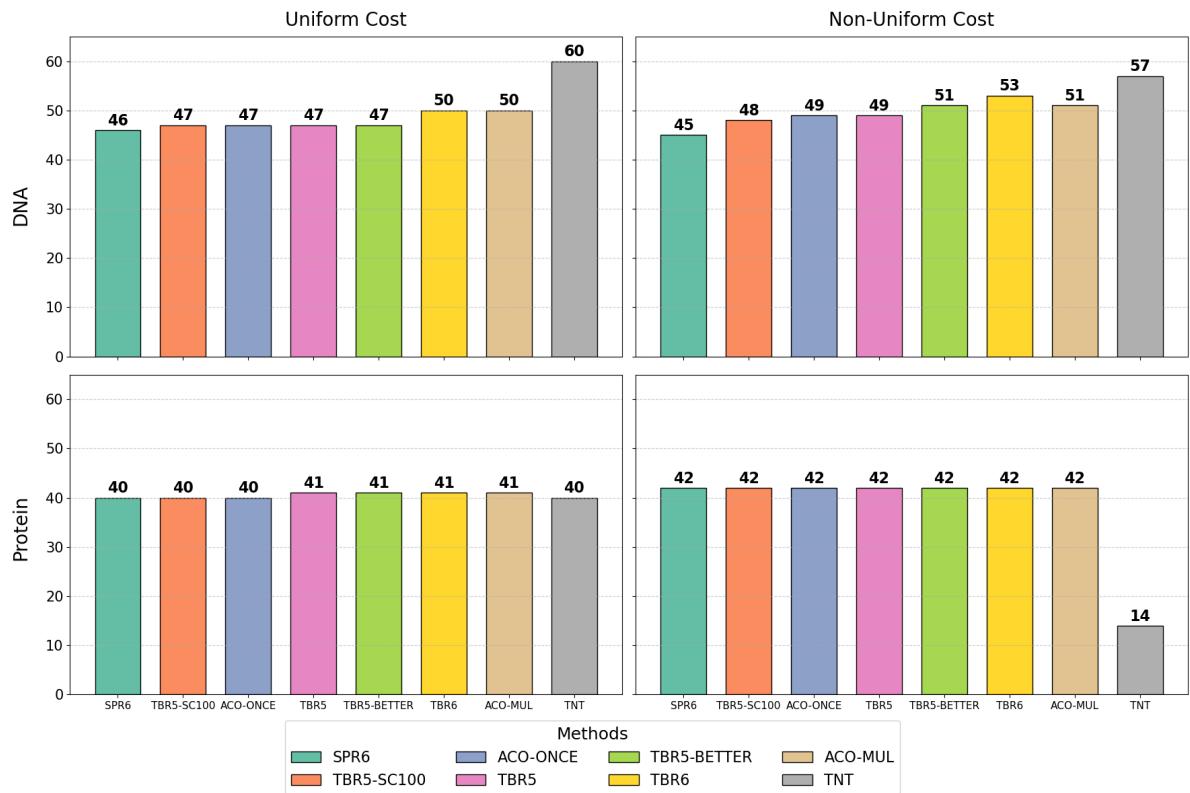
Các thí nghiệm được tiến hành bằng dữ liệu mô phỏng và dữ liệu sinh học (70 bộ DNAs và 45 bộ protein) như đề cập ở [Chương 4.2.1](#) trên hệ thống tính toán hiệu suất cao tại trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Các tiêu chí đánh giá cũng đã được đề cập ở [Chương 4.2.3](#).

MPBoot-RL được công khai mã nguồn tại địa chỉ Github <https://github.com/HynDuf/mpboot/tree/mpboot-rl>.

## 5.2.2. Kết quả

### 5.2.2.1. Điểm MP

Kết quả về điểm MP của 2 phương pháp MPBoot-RL với các phương pháp khác được trình bày trong [Hình 5.3](#). ACO-MUL hầu hết đạt được điểm MP tốt nhất so với các phương pháp khác của MPBoot, chỉ thua TBR6 ở dữ liệu DNA với chi phí không đồng nhất. Trong khi đó, ACO-ONCE có số bộ dữ liệu đạt điểm MP tốt nhất ngang với TBR5-SC100, nhỉnh hơn TBR5-SC100 một bộ dữ liệu DNA với chi phí không đồng nhất.



**Hình 5.3** — Hiệu suất của các phương pháp đánh giá trên bộ dữ liệu DNA và protein từ TreeBASE. Các biểu đồ cột thể hiện số lượng bộ dữ liệu (trong tổng số 115 bộ) mà phương pháp đạt được điểm số tốt nhất trong số các phương pháp khảo sát.

### 5.2.2.2. Thời gian thực thi

Trong [Bảng 5.1](#), dễ thấy ACO-MUL và ACO-ONCE là hai trong những phiên bản có thời gian thực thi nhanh nhất (ACO-ONCE nhanh hơn ACO-MUL nhưng không nhiều). Với điều kiện chi phí đồng nhất, hai phiên bản này chỉ chậm hơn một ít so với TBR5-SC100 (31,5 giờ (ACO-MUL) và 30,6 giờ (ACO-ONCE) so với 30,2 giờ (TBR5-SC100)). Không những thế, với điều kiện chi phí không đồng nhất, ACO-MUL và ACO-ONCE là hai phiên bản có thời gian thực thi nhanh nhất. Khi tính tỷ số thời gian chạy với phiên bản SPR6, cả hai phương pháp ACO đều cho thấy tốc độ tính toán nhanh hơn ( $\approx 0,78$  với điều kiện chi phí đồng nhất và  $\approx 0,84$  với điều kiện chi phí không đồng nhất).

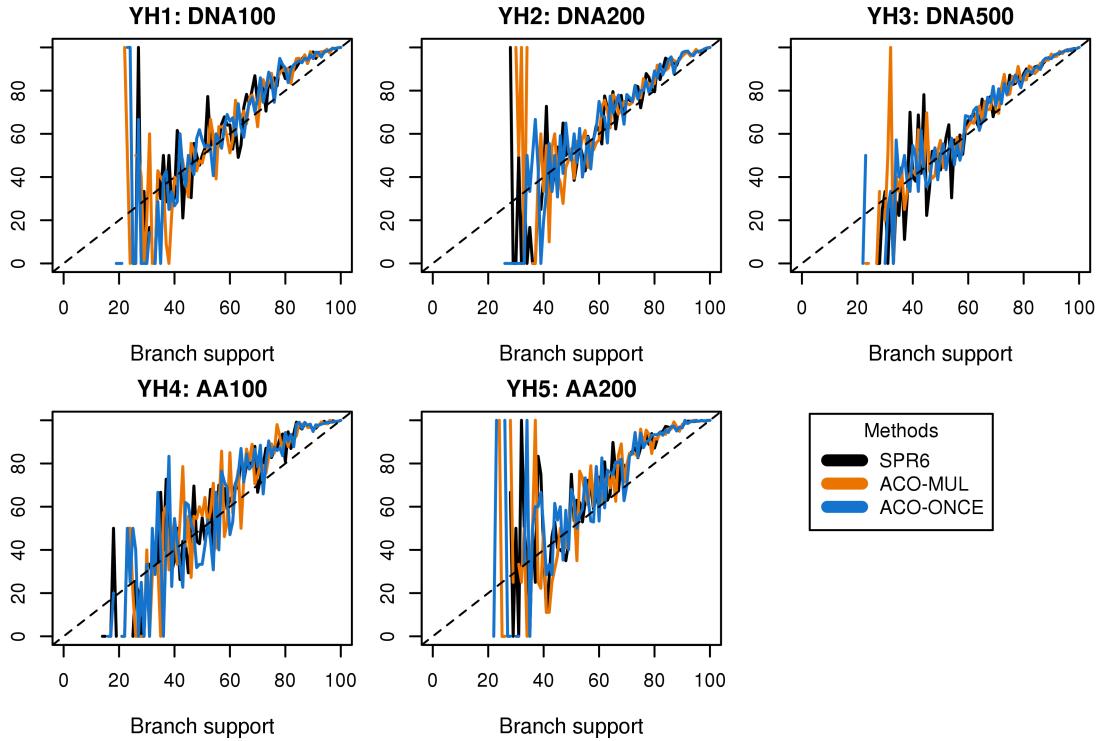
**Bảng 5.1** — Thời gian chạy tổng cộng (giờ) và tỷ lệ thời gian (so với SPR6) của các phương pháp trên 115 bộ dữ liệu từ bộ dữ liệu TreeBASE

Method	Uniform cost			Non-uniform cost		
	Time (hours)	Mean	Median	Time (hours)	Mean	Median
<b>ACO-MUL</b>	<b>31.5</b>	<b>0.79</b>	<b>0.79</b>	<b>144.7</b>	<b>0.85</b>	<b>0.84</b>
<b>ACO-ONCE</b>	<b>30.6</b>	<b>0.78</b>	<b>0.79</b>	<b>141.0</b>	<b>0.85</b>	<b>0.84</b>
SPR6	37.2	1.00	1.00	146.8	1.00	1.00
TBR5-SC100	30.2	1.04	0.96	192.8	1.26	1.17
TBR5	54.5	1.42	1.43	240.1	1.52	1.52
TBR5-BETTER	67.3	1.62	1.60	269.3	1.72	1.71
TBR6	78.2	1.98	1.96	338.8	2.21	2.18
TNT	75.5	1.23	0.47	682.3	6.47	3.55

Từ các thông số trên, có thể thấy ACO-MUL có khả năng tối ưu điểm MP tốt ngang ngửa với TBR6 nhưng tốc độ thực thi nhanh gấp  $\approx 2.3$  lần so với TBR6. Chi tiết các phân tích về điểm số và thời gian giữa các phương pháp được trình bày bổ sung ở [Phụ lục B](#).

### 5.2.2.3. Độ chính xác bootstrap

Hàm  $f_{\text{SPR6}}(v)$  (đường màu đen), hàm  $f_{\text{ACO-MUL}}(v)$  (đường màu cam) và hàm  $f_{\text{ACO-ONCE}}(v)$  (đường màu xanh) cho 5 bộ YuleHarding được minh họa ở [Hình 5.4](#). Trong cả 5 đồ thị, 2 đường cong của 2 hàm ACO này nằm sát nhau và cùng nằm phía trên đường chéo cho thấy phiên bản mới cho độ chính xác bootstrap tương đương MPBoot.



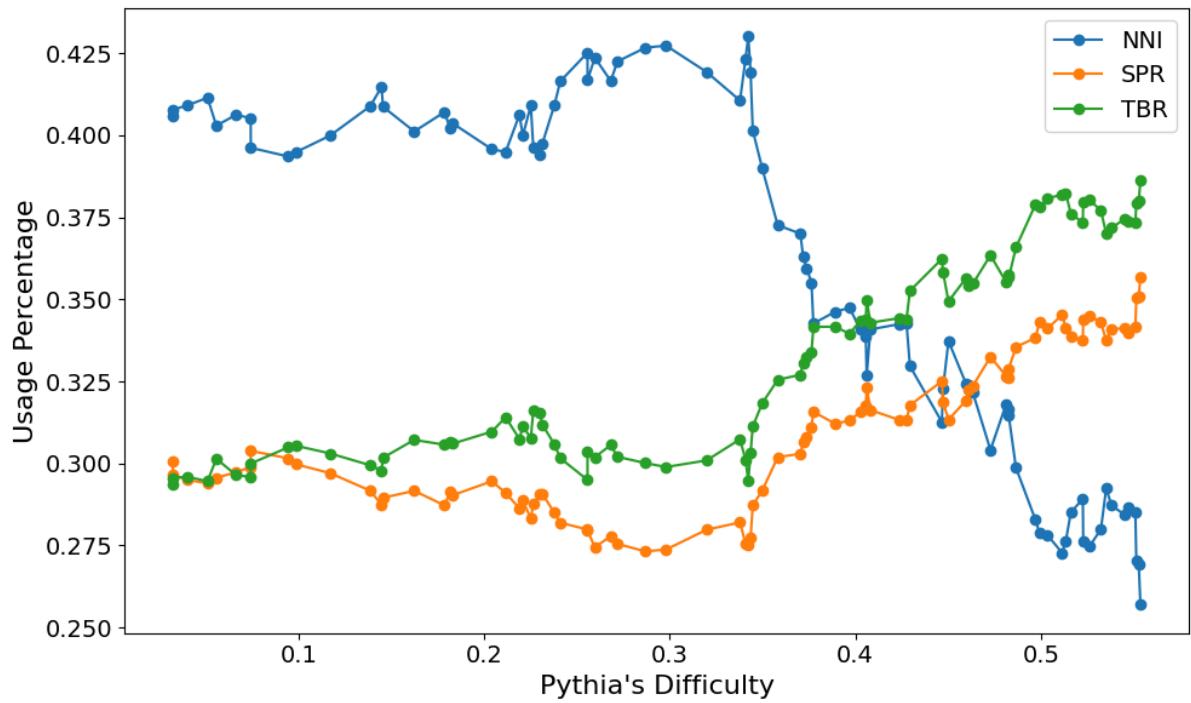
**Hình 5.4** — Độ chính xác bootstrap của các phương pháp ACO-MUL (đường màu cam), ACO-ONCE (đường màu xanh) và SPR6 (đường màu đen - của phiên bản MPBoot)

### 5.2.3. Khảo sát giữa các phép biến đổi cây với độ khó của dữ liệu

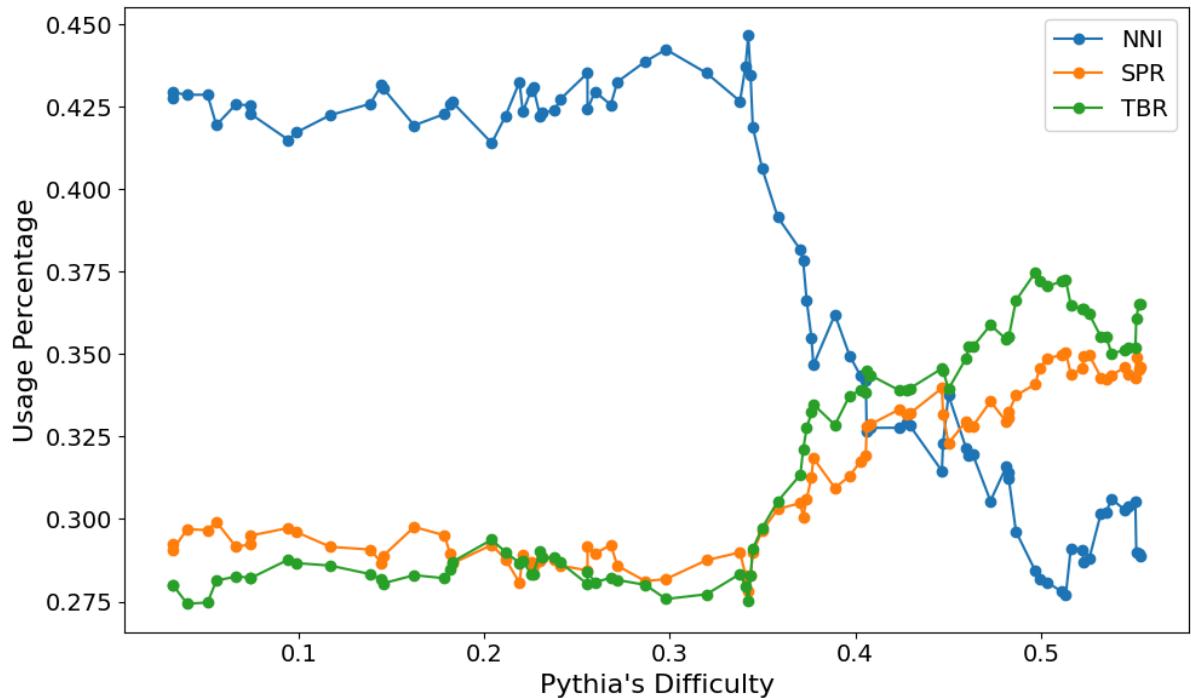
Liên quan đến độ khó của bộ dữ liệu, các nghiên cứu gần đây của [29] và [28] đã giới thiệu Pythia, một mô hình hồi quy Random Forest nhằm giải quyết vấn đề độ khó của bộ dữ liệu trong suy luận tiến hóa, và Adaptive RAxML-NG, một chiến lược thích ứng dựa trên mô hình Maximum Likelihood. Chiến lược thích ứng của họ điều chỉnh mức độ kỹ lưỡng trong việc tìm kiếm cây dựa trên độ khó được dự đoán. Pythia cho chỉ số độ khó trong khoảng từ 0.0 (dễ nhất) đến 1.0 (khó nhất).

Chúng tôi khảo sát tỉ lệ sử dụng 3 phép biến đổi cây NNI, SPR và TBR trên 115 bộ dữ liệu TreeBASE với độ khó tương ứng của từng bộ được Pythia dự đoán ở **Hình 5.5**. Dễ thấy rằng bộ dữ liệu mà Pythia đánh giá càng “khó” (chỉ số càng lớn), thì phép biến đổi SPR và TBR sẽ được sử dụng càng nhiều. Không những thế, trong những bộ dữ liệu khó đó, tỷ lệ tìm kiếm sử dụng phép TBR thường sẽ lớn hơn so với SPR.

(a) ACO-MUL



(b) ACO-ONCE



**Hình 5.5** — Khảo sát liên quan giữa ba phép biến đổi cây với độ khó Pythia của ACO-MUL và ACO-ONCE (đã làm mượt với window size = 30)

Khảo sát trên gợi ý cho việc đánh giá độ khó bộ dữ liệu sử dụng thông số sử dụng các phép biến đổi cây là khả thi. Một đề xuất đơn giản có thể là khi tỉ lệ sử dụng NNI lớn hơn so với 2 phép biến đổi còn lại thì bộ dữ liệu sẽ được đánh giá là “dễ” (tương ứng với độ khó Pythia  $\in (0.0; 0.35)$ ), trong trường hợp 3 tỉ lệ ngang nhau sẽ được đánh giá là “trung bình” (tương ứng với độ khó Pythia  $\in (0.35; 0.45)$ ) và các trường hợp còn lại sẽ được đánh giá là “khó” (độ khó Pythia  $\in (0.45; 1.0)$ ). Tuy nhiên, để việc đánh giá độ khó được chính xác và khoa học hơn cần nghiên cứu thêm kết hợp một số đặc tính khác của bộ dữ liệu.

Chú ý rằng, phương pháp Pythia khác biệt so với phương pháp của chúng tôi, vì Adaptive RAxML-NG yêu cầu huấn luyện mô hình trên một bộ dữ liệu tự gán nhãn. Sau khi huấn luyện, mô hình được sử dụng để dự đoán độ khó và triển khai một thuật toán chiến lược dựa trên độ khó đã dự đoán. Ngược lại, nghiên cứu của chúng tôi tập trung vào khả năng tự điều chỉnh và lựa chọn các phép biến đổi cây một cách tự nhiên bằng cách sử dụng thuật toán tối ưu hóa đàm kiến (Ant Colony Optimization). Sau đó, chúng tôi có thể phân tích sâu vào quy trình tự điều chỉnh này để ước lượng độ khó của bộ dữ liệu.

# Chương 6

## Kết luận

### 6.1. Kết quả đạt được

Qua quá trình nghiên cứu, cài đặt, thực nghiệm và hiệu chỉnh tham số, khóa luận đã đạt được những kết quả như sau:

- Trình bày những khái niệm cơ bản về bài toán xây dựng cây bootstrap tiến hóa, thuật toán bootstrap chuẩn và xấp xỉ, các tiêu chuẩn đánh giá cây tiến hóa, các kĩ thuật biến đổi cây và các công trình liên quan.
- Trình bày những kiến thức về mô hình giải thuật tối ưu đàm kiến (ACO), các thành phần và các biến thể của thuật toán.
- Trình bày đề xuất tích hợp phép biến đổi cây TBR vào khung thuật toán MPBoot, chi tiết phương pháp cài đặt, tối ưu tính toán và sử dụng leo đồi TBR vào MPBoot. Hai chiến lược tìm kiếm với TBR được đề xuất bao gồm chiến lược tìm kiếm “tốt nhất” và chiến lược tìm kiếm “tốt hơn”.
- Trình bày đề xuất MPBoot2 – chính thức tích hợp TBR vào MPBoot cùng với cài đặt nhiều tính năng mới cho MPBoot như checkpoint, khả năng xử lý dữ liệu lớn, dữ liệu morphology, dữ liệu nhị phân...
- Trình bày đề xuất MPBoot-RL – kết hợp các phép biến đổi cây NNI, SPR và TBR vào pha khám phá leo đồi của MPBoot sử dụng giải thuật tối ưu đàm kiến. Hai phiên bản được đề xuất bao gồm ACO-MUL (sử dụng quy tắc cập nhật mì SMMAS-multiple) và ACO-ONCE (sử dụng quy tắc cập nhật mì SMMAS gốc).
- Kết quả thực nghiệm cho thấy các phiên bản TBR cho điểm MP tốt hơn so với phiên bản MPBoot cũ (SPR6) nhưng hầu hết có thời gian thực thi lâu hơn, trừ phiên bản TBR5-SC100 có thời gian tương đương với SPR6. So với TNT, các phiên bản của MPBoot2, tuy không bằng TNT với điều kiện

chi phí đồng nhất, áp đảo TNT với điều kiện chi phí không đồng nhất. Với các phiên bản MPBoot-RL, ACO-MUL và ACO-ONCE cho thấy kết quả điểm số tốt hơn hầu hết các phiên bản MPBoot khác và thời gian thực thi áp đảo.

- Thông qua thực nghiệm trên dữ liệu sinh theo mô hình Yule-Harding cho thấy độ chính xác bootstrap được đảm bảo tương đương với thuật toán gốc.
- Mã nguồn của MPBoot2 và MPBoot-RL được công bố lần lượt tại <https://github.com/HynDuf/mpboot/tree/mpboot2> và <https://github.com/HynDuf/mpboot/tree/mpboot-rl>.

## 6.2. Các định hướng phát triển

Các định hướng phát triển thuật toán trong tương lai bao gồm:

- Thủ nghiệm MPBoot-RL với các quy tắc cập nhật mùi khác.
- Phân tích độ khó bộ dữ liệu dựa trên thông số sử dụng các phép biến đổi cây kết hợp với các đặc tính khác của bộ dữ liệu.
- Ý tưởng áp dụng giải thuật đòn kiến có thể áp dụng tương tự vào quá trình phá cây (các thuật toán phá cây như ratchet, random NNIs, IQP...)
- Tái cấu trúc phần cài đặt nhằm tối ưu việc thêm một phép biến đổi cây khác và kết hợp chúng sử dụng ACO trong tương lai.

# Công bố liên quan

- T. D. Huynh, Q. T. Vu, V. D. Nguyen and D. T. Hoang, “Employing tree bisection and reconnection rearrangement for parsimony inference in MPBoot,” 2022 14th International Conference on Knowledge and Systems Engineering (KSE), Nha Trang, Vietnam, 2022, pp. 1-6, doi: 10.1109/KSE56063.2022.9953773.
- Giải Nhì Hội nghị Sinh viên Nghiên cứu Khoa học cấp Đại học Quốc gia (VNU), Vietnam, 08/2023, đề tài “Kết hợp các phép biến đổi cây để tối ưu hóa việc tính điểm MP sử dụng tiếp cận học tăng cường”.

# Tài liệu tham khảo

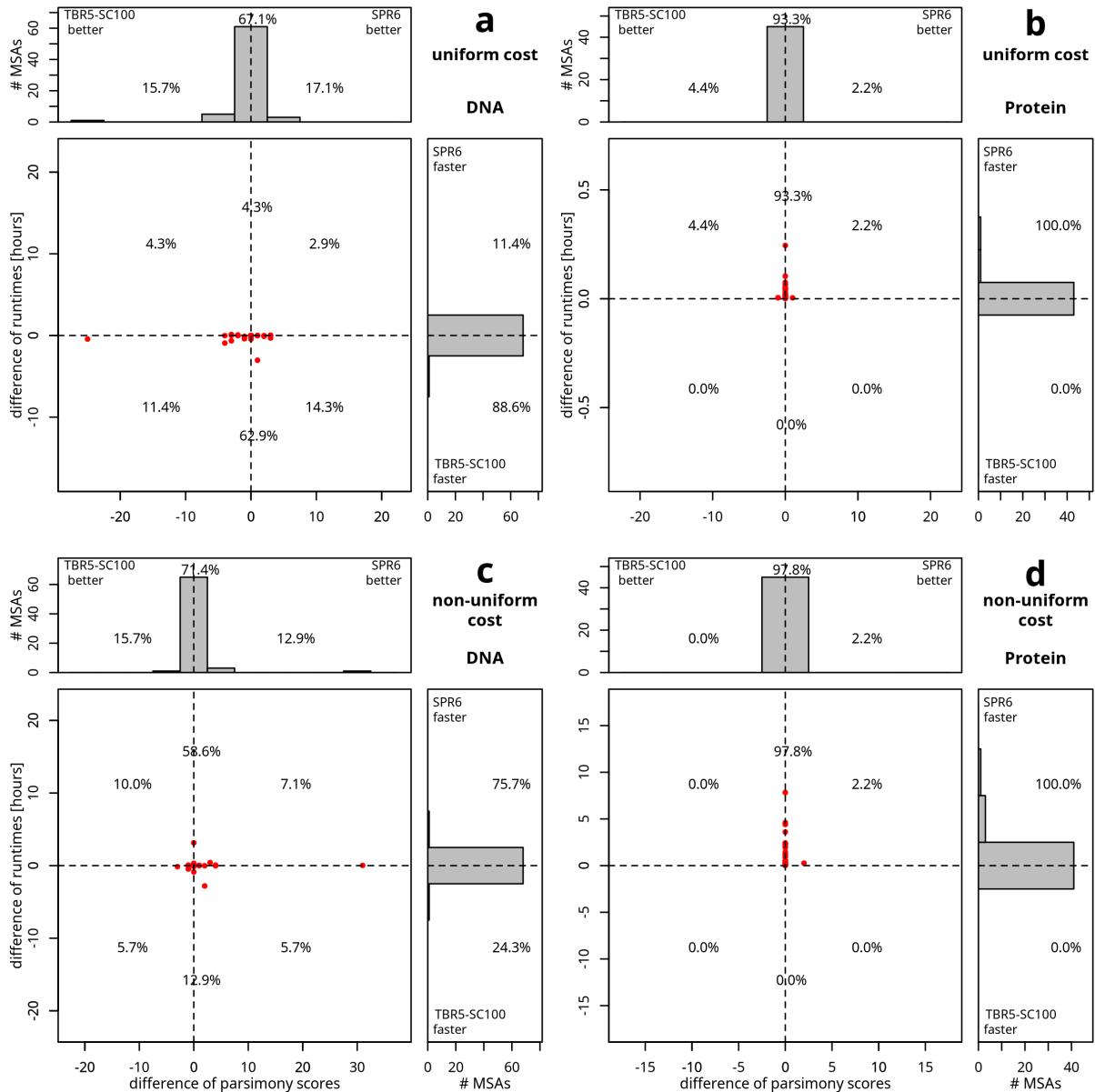
- [1] Darwin's C. On the origin of species 1859;24
- [2] Nayfach S, Roux S, Seshadri R, Udwary D, Varghese N, Schulz F, và c.s. A genomic catalog of Earth's microbiomes. *Nature biotechnology* 2021;39:499–509
- [3] Zheng J, Wittouck S, Salvetti E, Franz C M, Harris H M, Mattarelli P, và c.s. A taxonomic note on the genus Lactobacillus: Description of 23 novel genera, emended description of the genus Lactobacillus Beijerinck 1901, and union of Lactobacillaceae and Leuconostocaceae. *International journal of systematic and evolutionary microbiology* 2020;70:2782–858
- [4] Gonzalez-Reiche A S, Hernandez M M, Sullivan M J, Ciferri B, Alshammary H, Obla A, và c.s. Introductions and early spread of SARS-CoV-2 in the New York City area. *Science* 2020;369:297–301
- [5] Hodcroft E, others. Want to track pandemic variants faster. Fix the bioinformatics bottleneck không ngày;811:30–3
- [6] Wruck W, Adjaye J. Detailed phylogenetic analysis tracks transmission of distinct SARS-COV-2 variants from China and Europe to West Africa. *Scientific Reports* 2021;11:21108
- [7] Efron B. Bootstrap methods: another look at the jackknife. *Breakthroughs in statistics: Methodology and distribution* 1992:569–93
- [8] Graham R L, Foulds L R. Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Mathematical Biosciences* 1982;60:133–42
- [9] Hoang D T, Vinh L S, Flouri T, Stamatakis A, Haeseler A von, Minh B Q. MPBoot: fast phylogenetic maximum parsimony tree inference and bootstrap approximation. *BMC evolutionary biology* 2018;18:1–11
- [10] Felsenstein J. Confidence limits on phylogenies: an approach using the bootstrap. *evolution* 1985;39:783–91

- [11] Dhar A, Minin V N. Maximum Likelihood Methods for Phylogenetic Inference 2015
- [12] Fitch W M. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology* 1971;20:406–16
- [13] Sankoff D. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics* 1975;28:35–42
- [14] Lemey P, Salemi M, Vandamme A-M. The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing. Cambridge University Press; 2009
- [15] Goloboff P A, Morales M E. TNT version 1.6, with a graphical interface for MacOS and Linux, including new routines in parallel. *Cladistics* 2023;39:144–53
- [16] Swofford D L, Sullivan J, others. Phylogeny inference based on parsimony and other methods using PAUP\*. The phylogenetic handbook: a practical approach to DNA and protein phylogeny 2003;7:160–206
- [17] Tamura K, Dudley J, Nei M, Kumar S. MEGA4: molecular evolutionary genetics analysis (MEGA) software version 4.0. *Molecular biology and evolution* 2007;24:1596–9
- [18] Hillis D M, Bull J J. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Systematic biology* 1993;42:182–92
- [19] Minh B Q, Nguyen M A T, Von Haeseler A. Ultrafast approximation for phylogenetic bootstrap. *Molecular biology and evolution* 2013;30:1188–95
- [20] Colorni A, Dorigo M, Maniezzo V, others. Distributed optimization by ant colonies. trong:. Proceedings of the first European conference on artificial life, vol. 142, 1991, tr 134–42
- [21] Do Duc D, Dinh H Q, Hoang Xuan H. On the pheromone update rules of ant colony optimization approaches for the job shop scheduling problem. trong:. Intelligent Agents and Multi-Agent Systems: 11th Pacific Rim International Conference on Multi-Agents, PRIMA 2008, Hanoi, Vietnam, December 15-16, 2008. Proceedings 11, 2008, tr 153–60

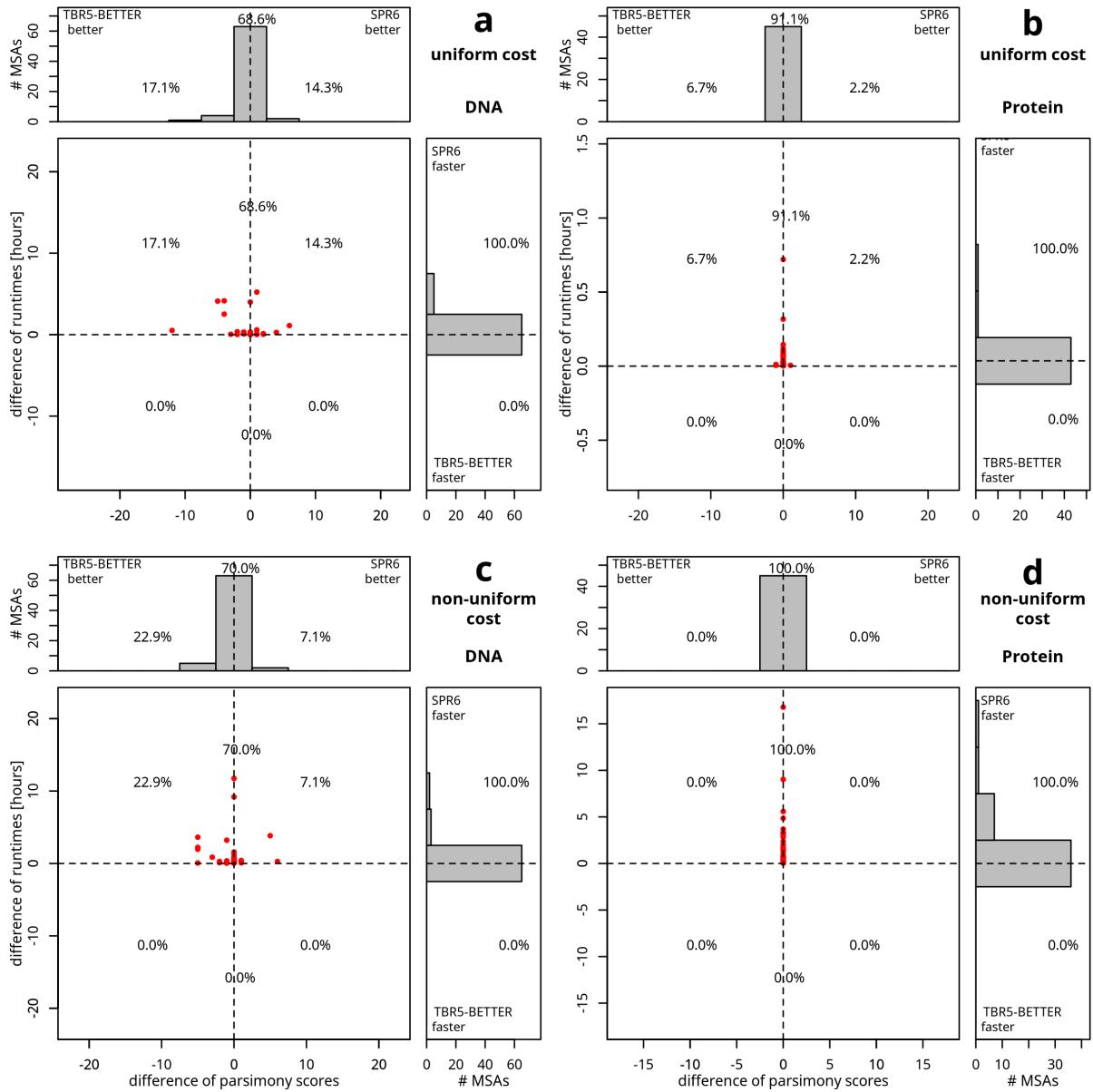
- [22] Minh B Q, Schmidt H A, Chernomor O, Schrempf D, Woodhams M D, Von Haeseler A, và c.s. IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution* 2020;37:1530–4
- [23] Flouri T, Izquierdo-Carrasco F, Darriba D, Aberer A J, Nguyen L-T, Minh B, và c.s. The phylogenetic likelihood library. *Systematic biology* 2015;64:356–62
- [24] Piel W H, Donoghue M, Sanderson M, Netherlands L. TreeBASE: a database of phylogenetic information. trong:. Proceedings of the 2nd International Workshop of Species, vol. 2000, 2000
- [25] Nguyen L-T, Schmidt H A, Von Haeseler A, Minh B Q. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution* 2015;32:268–74
- [26] Chifman J, Kubatko L. Quartet inference from SNP data under the coalescent model. *Bioinformatics* 2014;30:3317–24
- [27] Minh B Q, Dang C C, Vinh L S, Lanfear R. QMaker: fast and accurate method to estimate empirical models of protein evolution. *Systematic Biology* 2021;70:1046–60
- [28] Haag J, Höhler D, Bettsworth B, Stamatakis A. From easy to hopeless —predicting the difficulty of phylogenetic analyses. *Molecular Biology and Evolution* 2022;39:msac254
- [29] Togkousidis A, Kozlov O M, Haag J, Höhler D, Stamatakis A. Adaptive RAxML-NG: accelerating phylogenetic inference under maximum likelihood using dataset difficulty. *Molecular Biology and Evolution* 2023;40:msad227

# Phụ lục

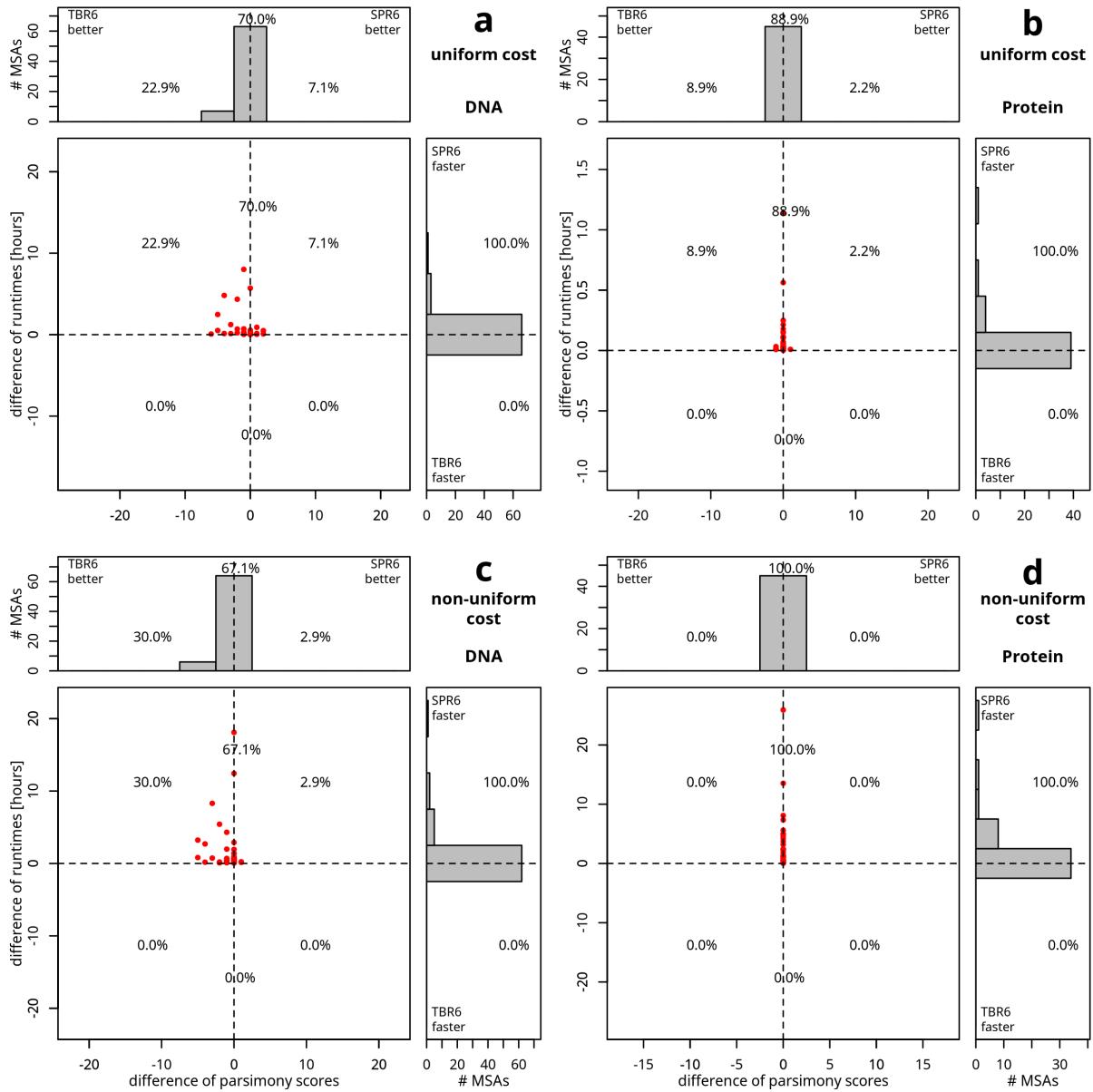
## A. Kết quả phân tích bổ sung của các phiên bản MPBoot2



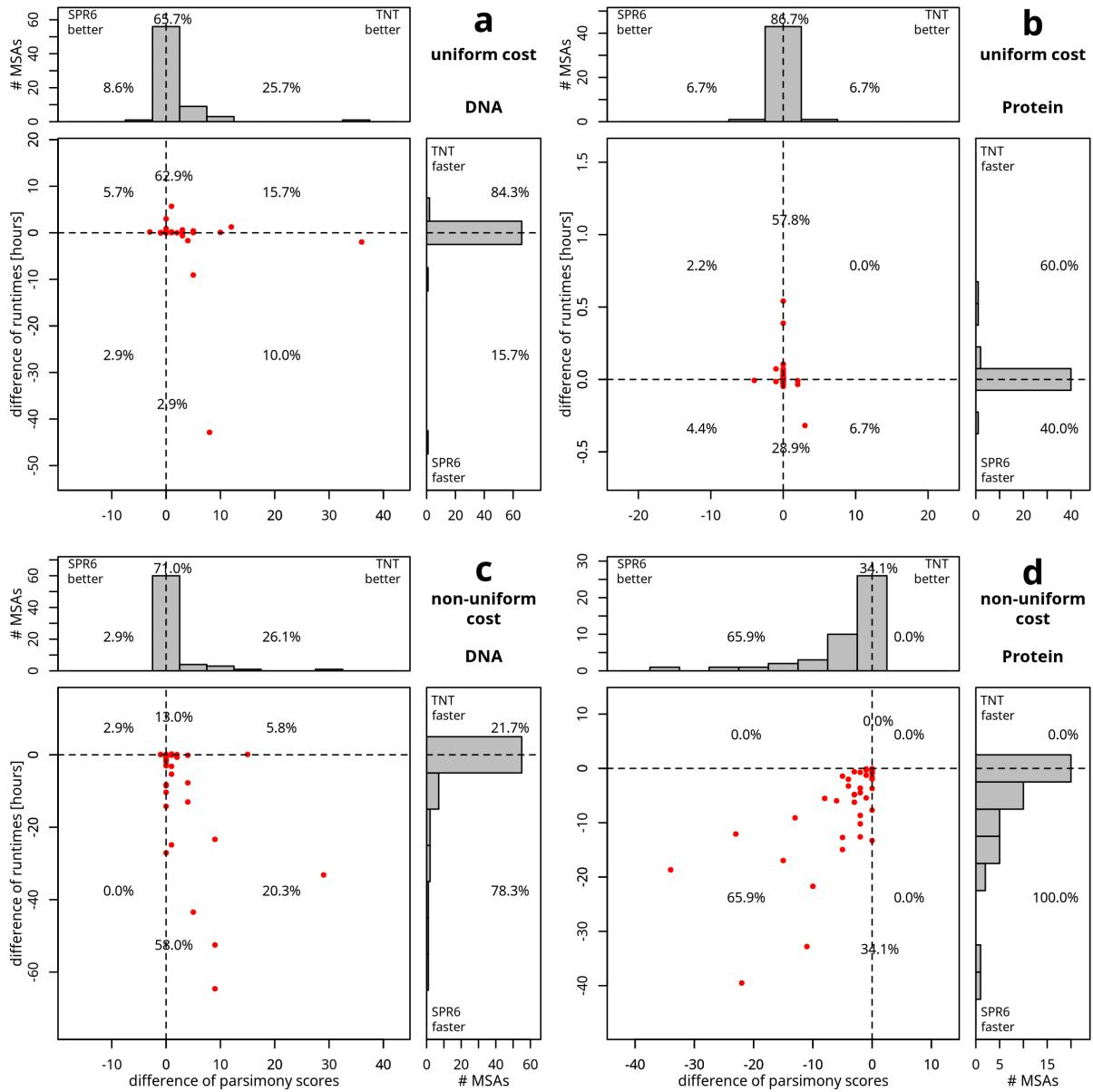
**Hình A.1** — So sánh kết quả của TBR5-SC100 với SPR6 trên 115 bộ dữ liệu TreeBASE



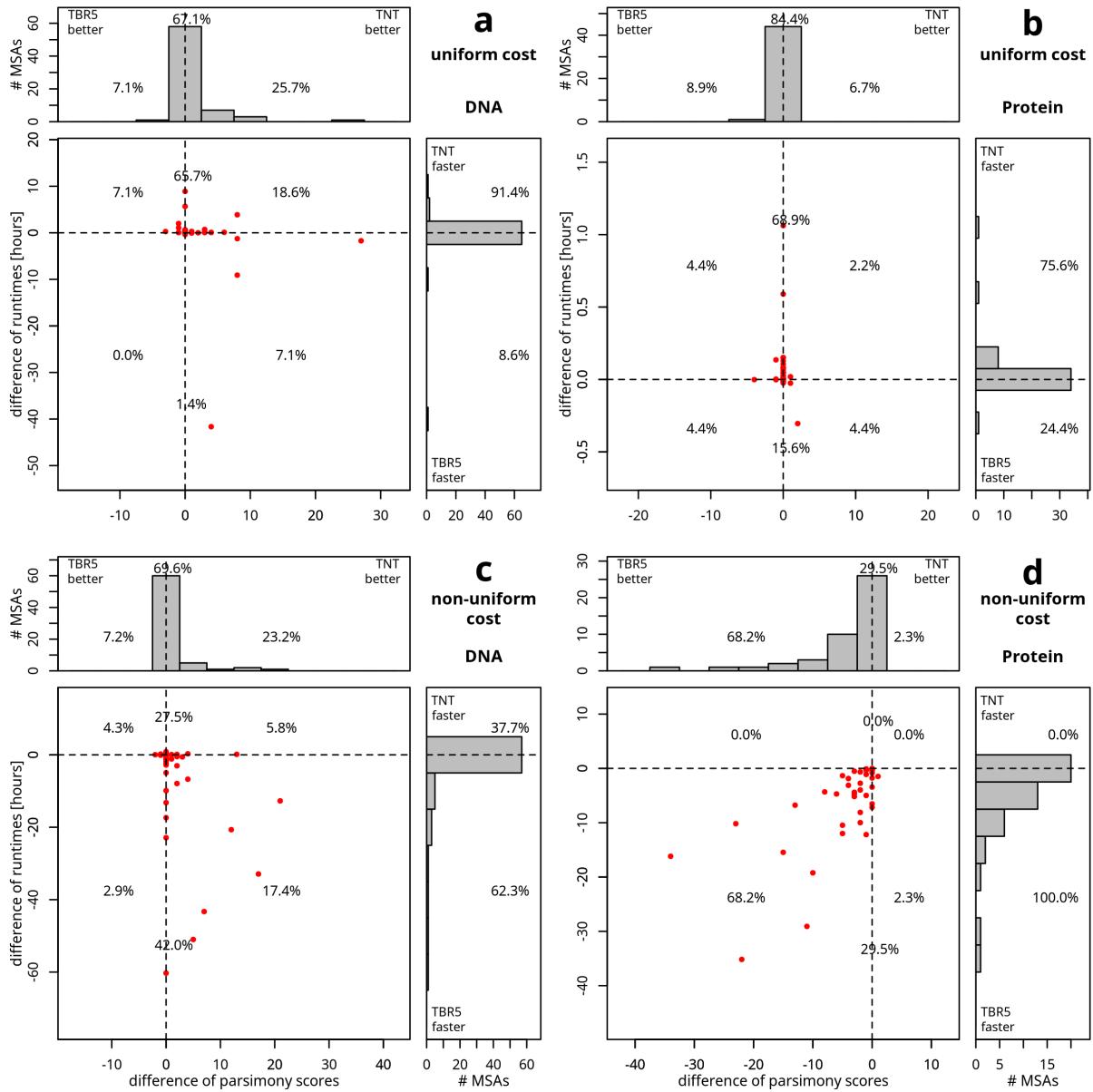
**Hình A.2** — So sánh kết quả của TBR5-BETTER với SPR6 trên 115 bộ dữ liệu TreeBASE



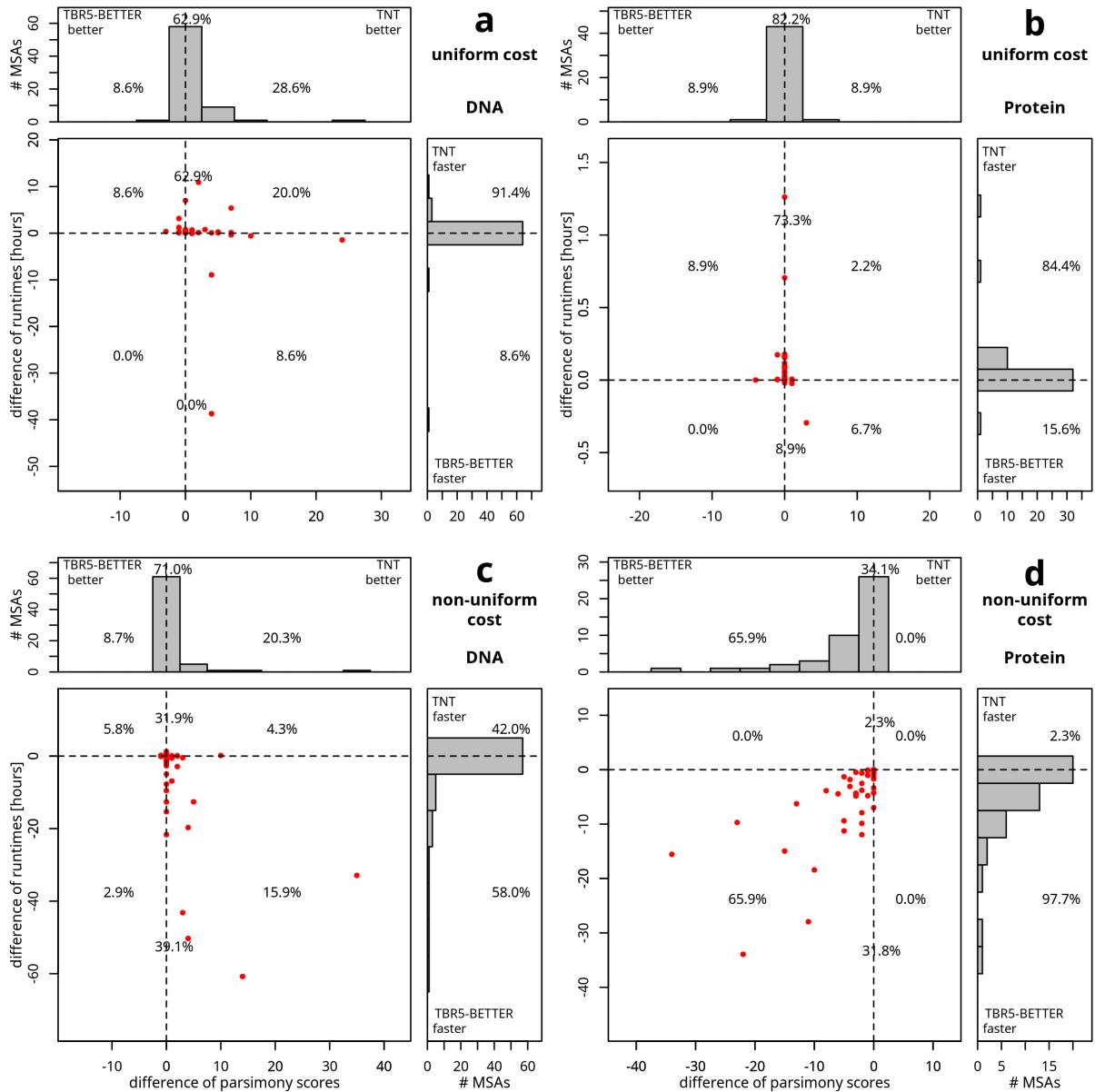
**Hình A.3** — So sánh kết quả của TBR6 với SPR6 trên 115 bộ dữ liệu TreeBASE



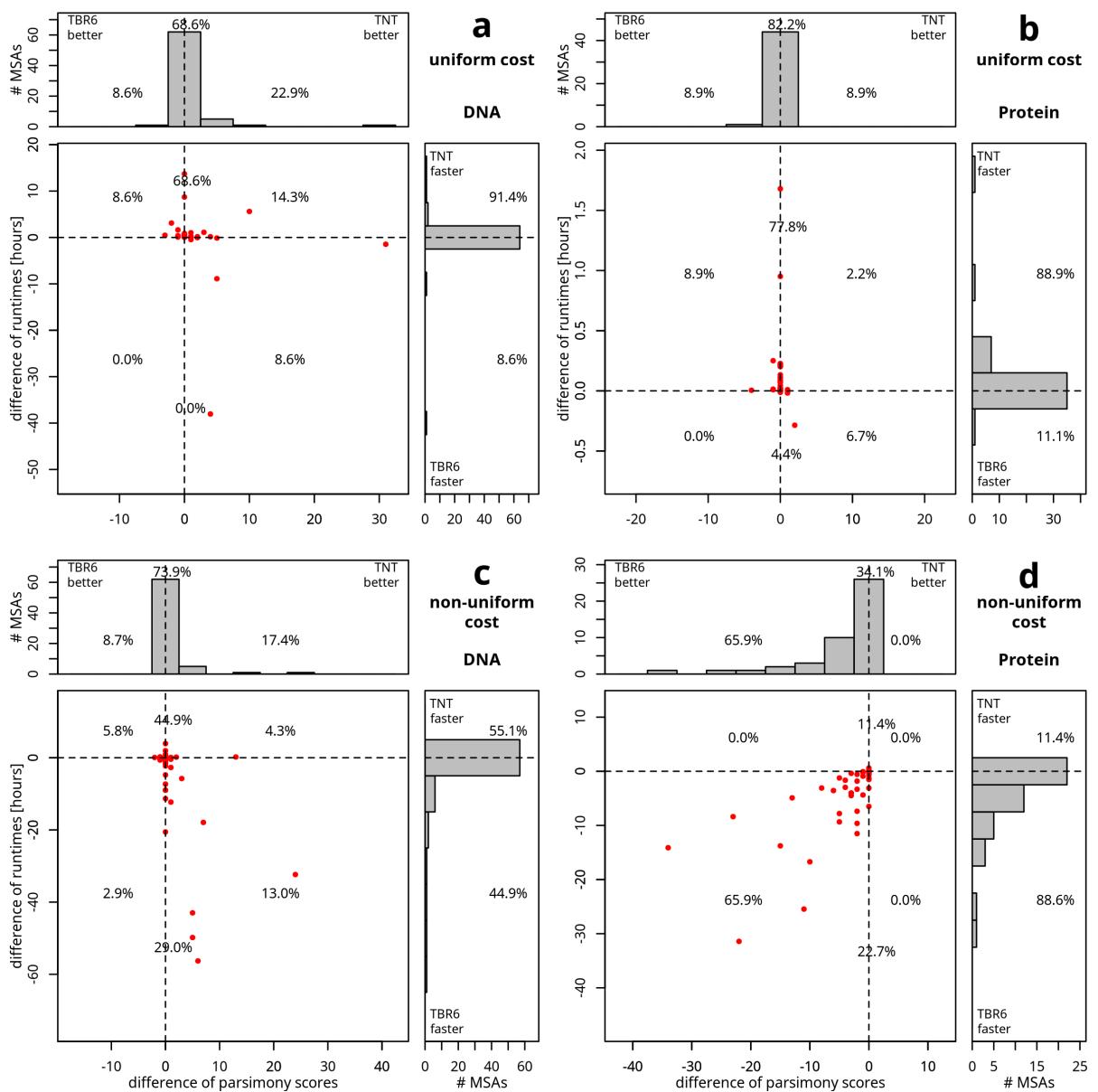
**Hình A.4** — So sánh kết quả của SPR6 với TNT trên 115 bộ dữ liệu TreeBASE



**Hình A.5** — So sánh kết quả của TBR5 với TNT trên 115 bộ dữ liệu TreeBASE

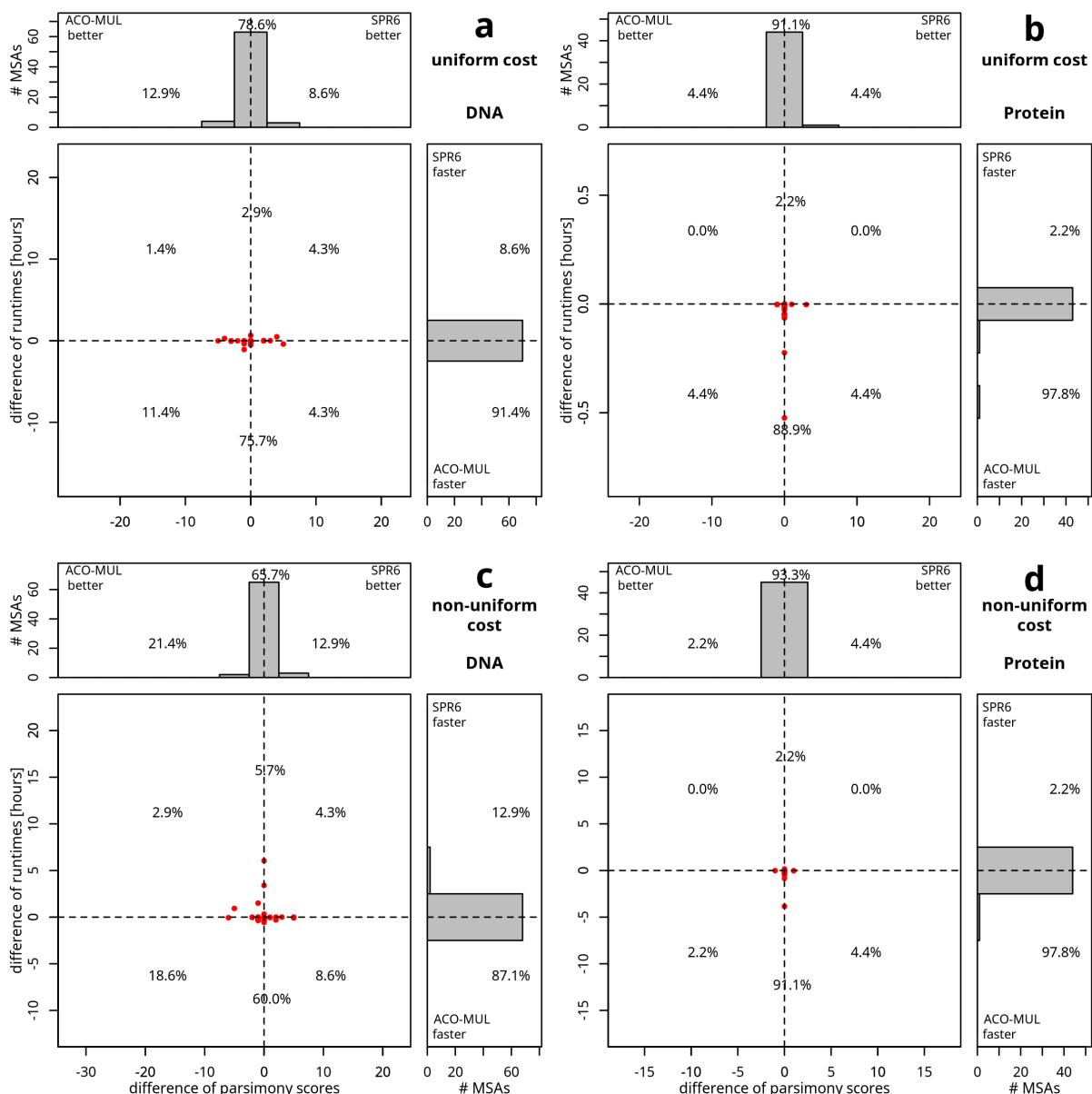


**Hình A.6** — So sánh kết quả của TBR5-BETTER với TNT trên 115 bộ dữ liệu TreeBASE

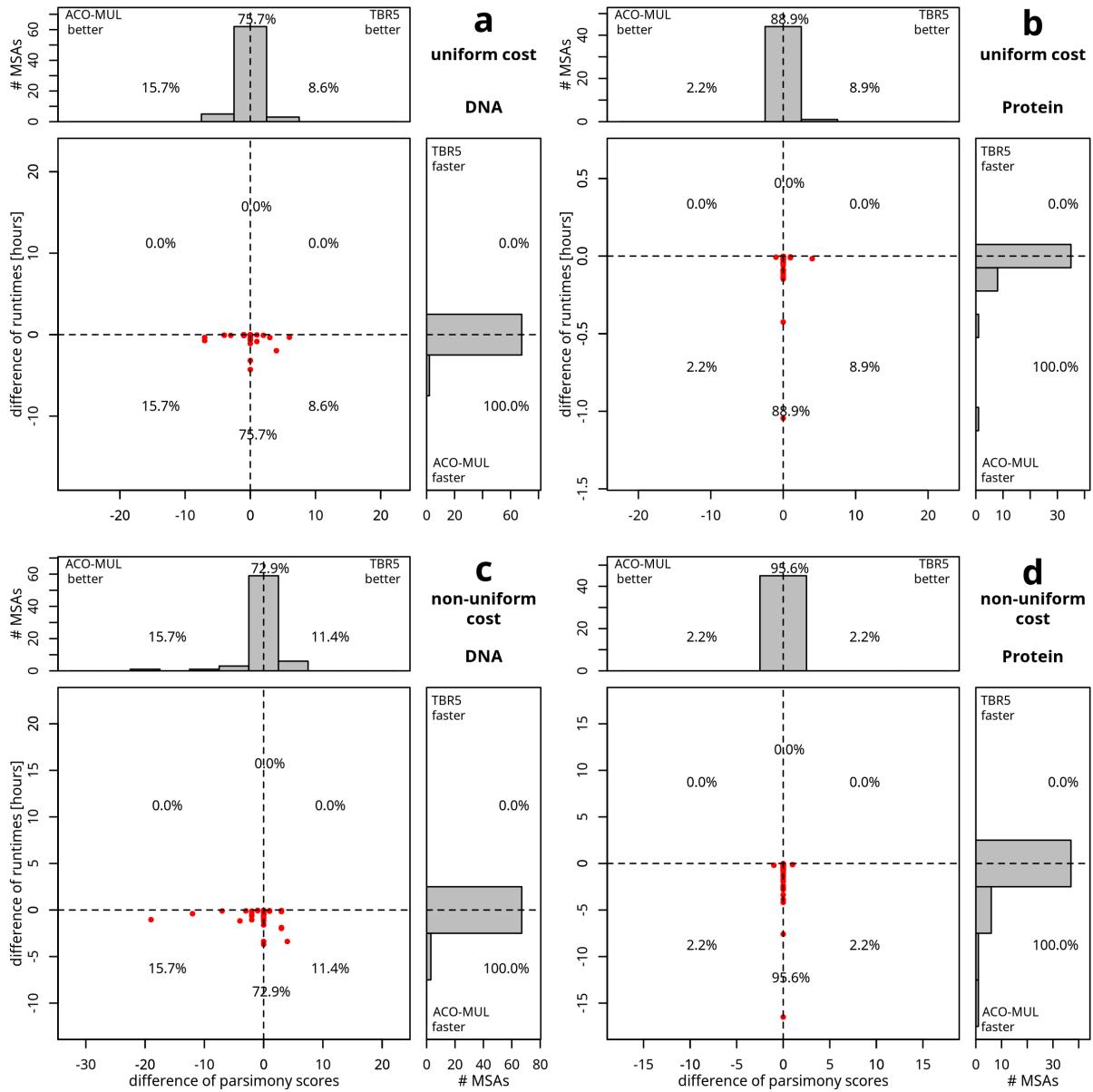


**Hình A.7** — So sánh kết quả của TBR6 với TNT trên 115 bộ dữ liệu TreeBASE

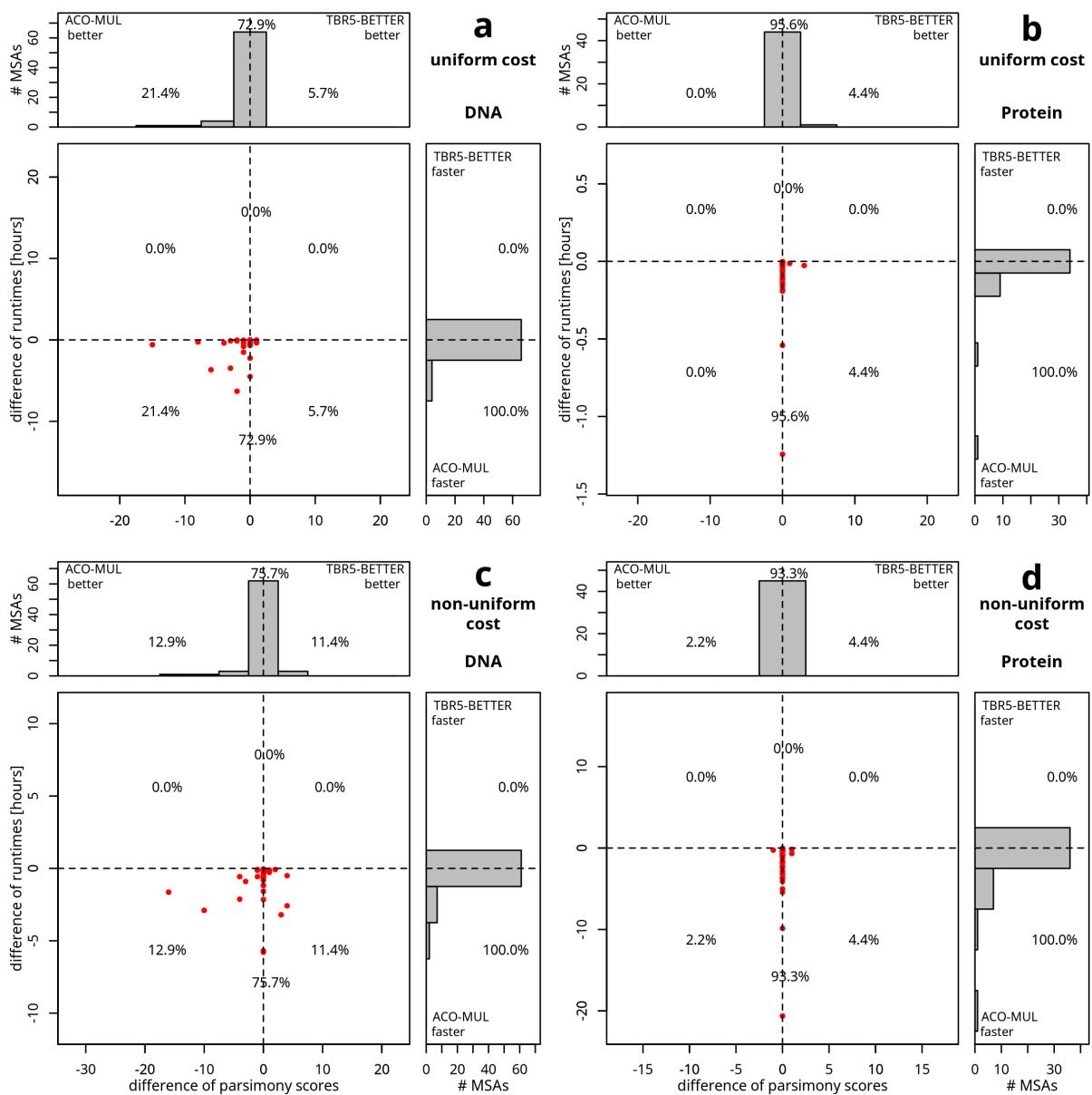
## B. Kết quả phân tích bổ sung của các phiên bản MPBoot-RL



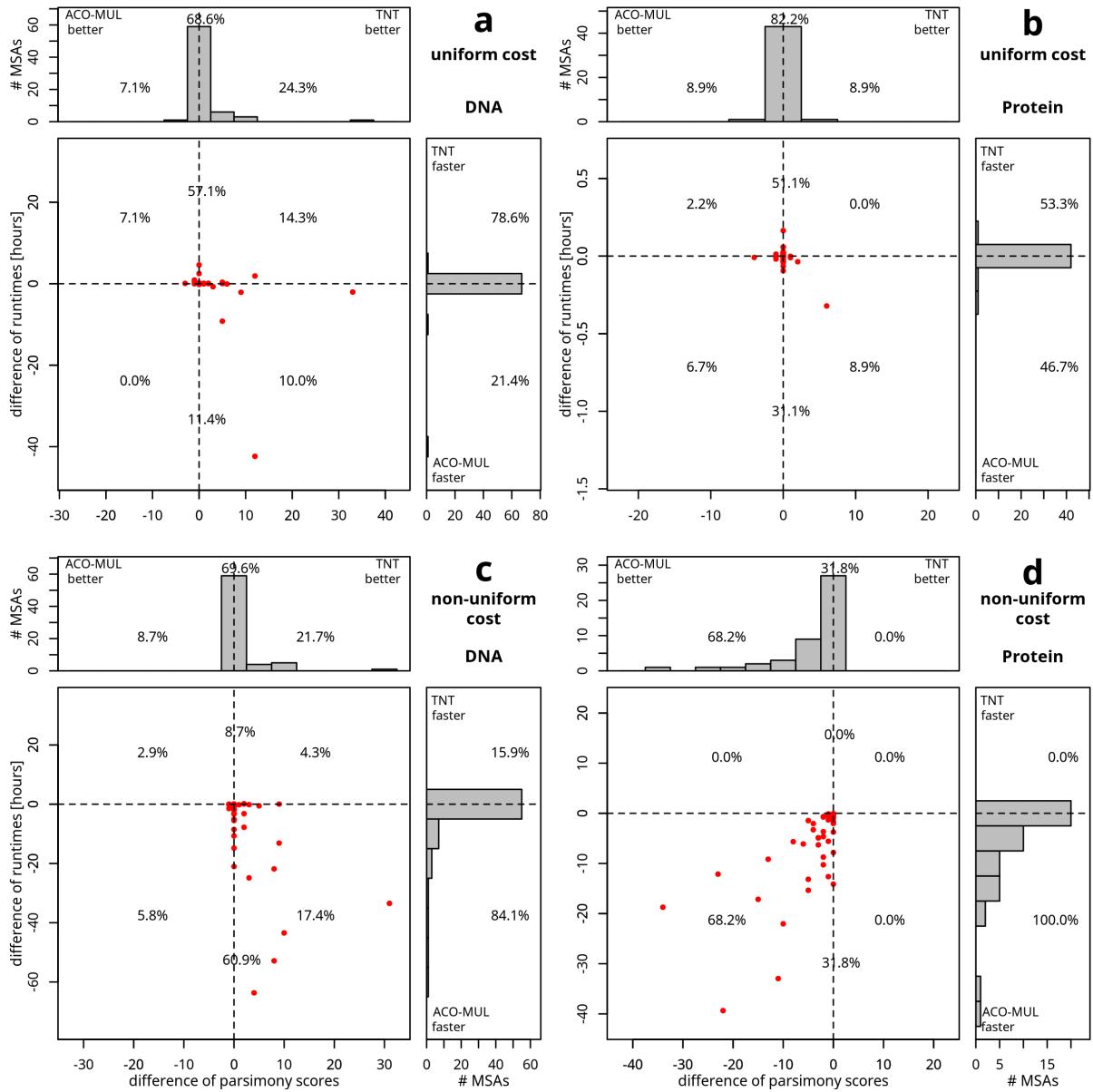
**Hình B.8** — So sánh kết quả của ACO-MUL với SPR6 trên 115 bộ dữ liệu TreeBASE



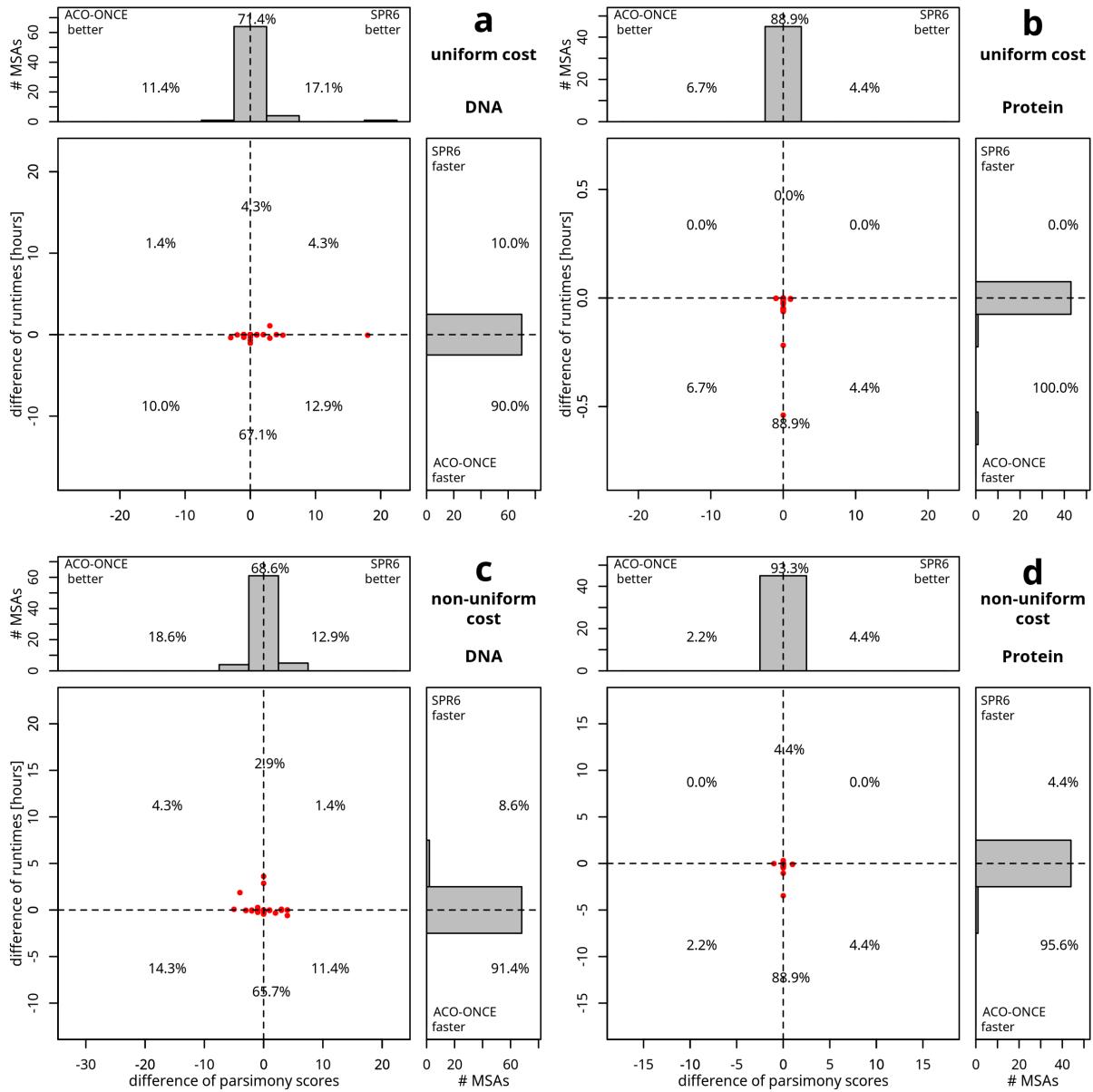
**Hình B.9** — So sánh kết quả của ACO-MUL với TBR5 trên 115 bộ dữ liệu TreeBASE



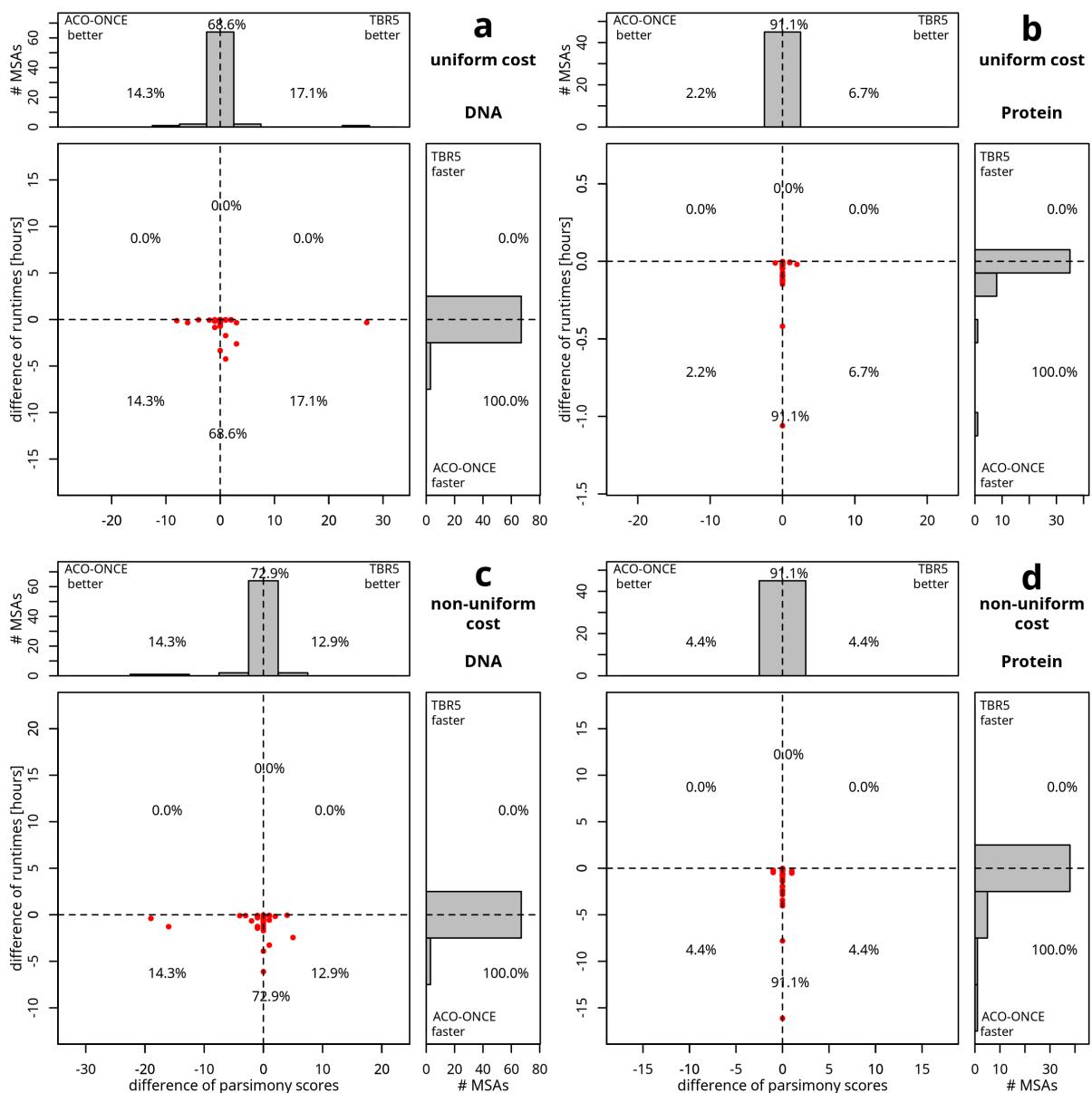
**Hình B.10** — So sánh kết quả của ACO-MUL với TBR5-BETTER trên 115 bộ dữ liệu TreeBASE



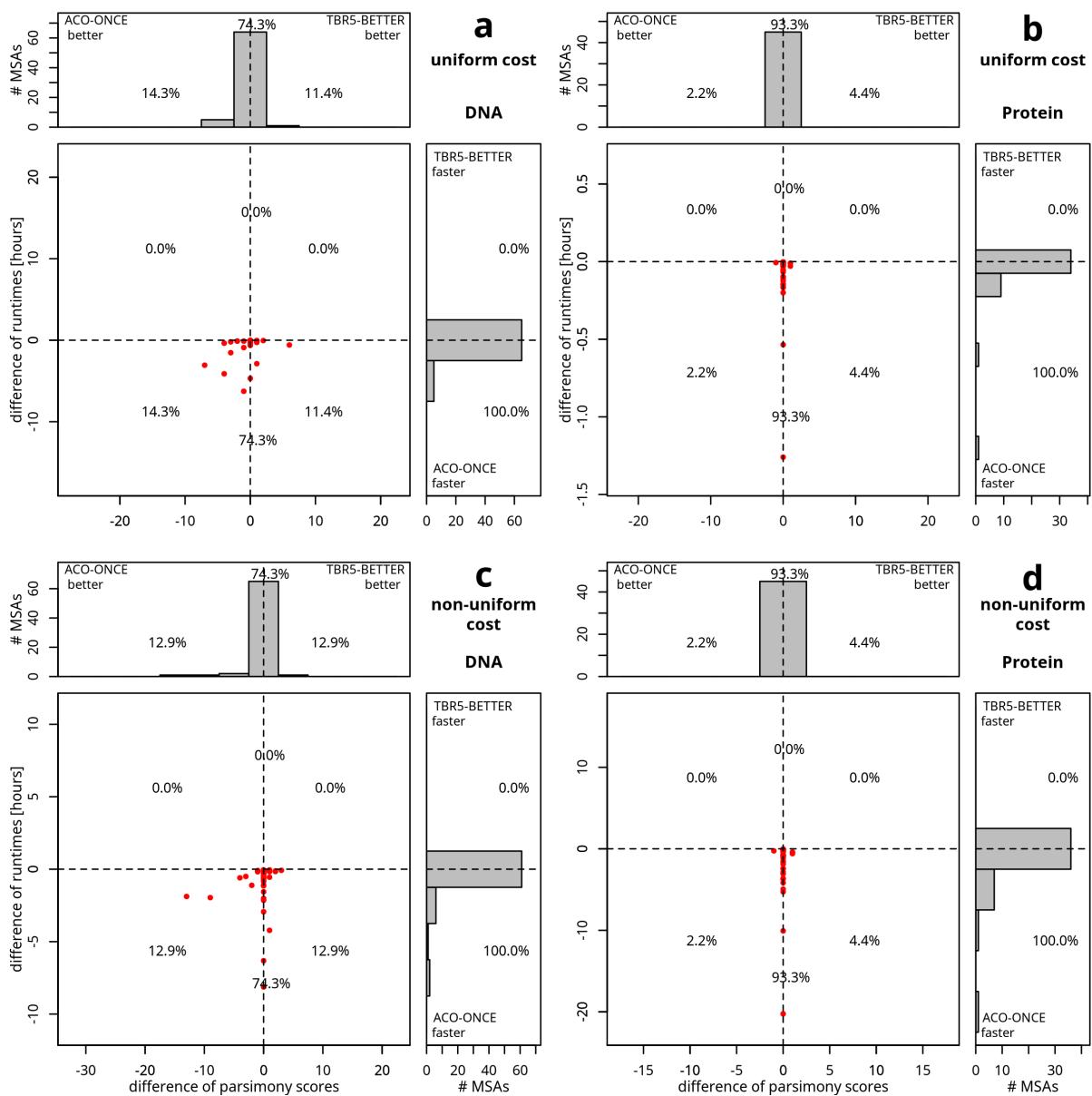
**Hình B.11** — So sánh kết quả của ACO-MUL với TNT trên 115 bộ dữ liệu TreeBASE



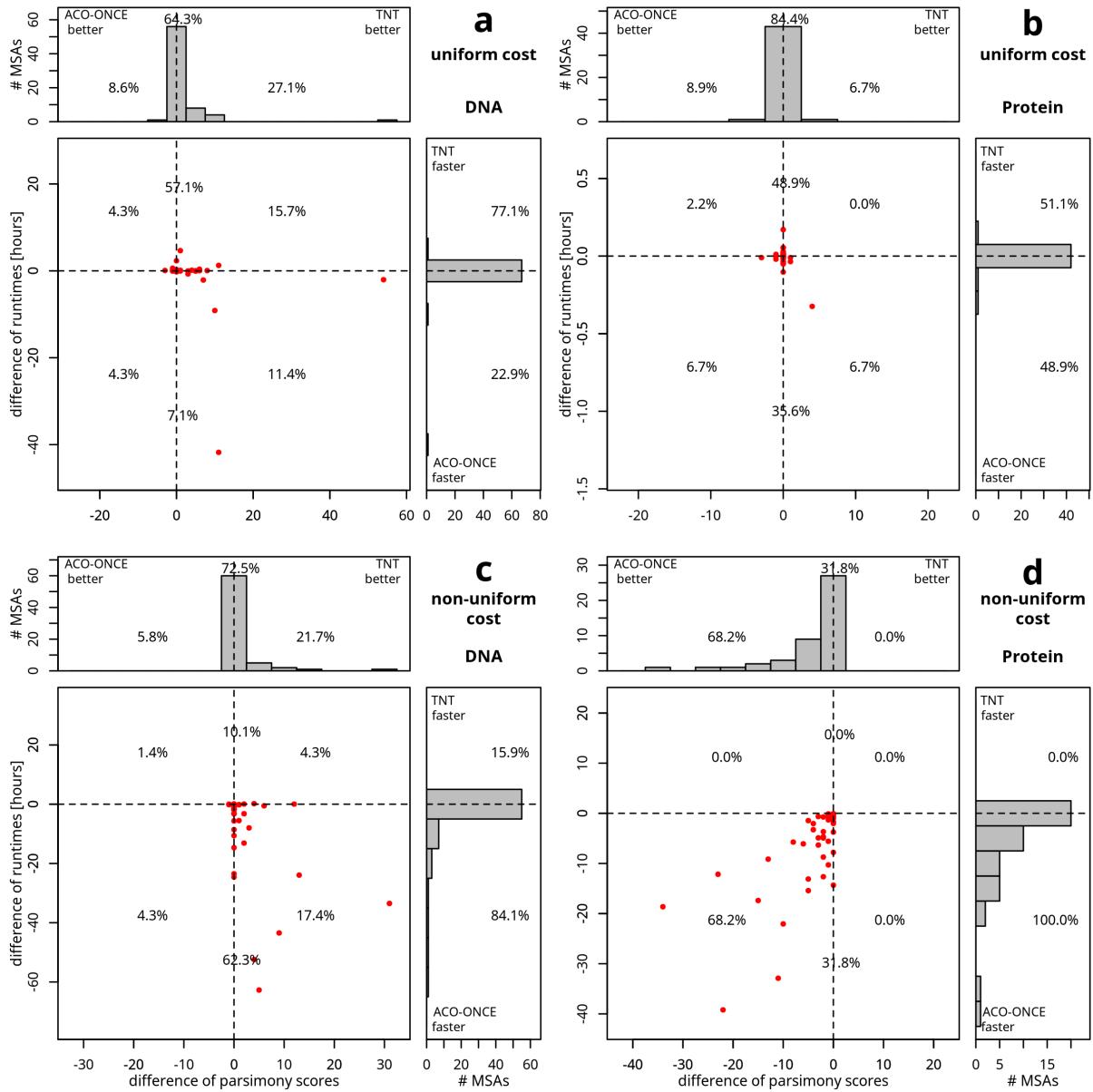
**Hình B.12** — So sánh kết quả của ACO-ONCE với SPR6 trên 115 bộ dữ liệu TreeBASE



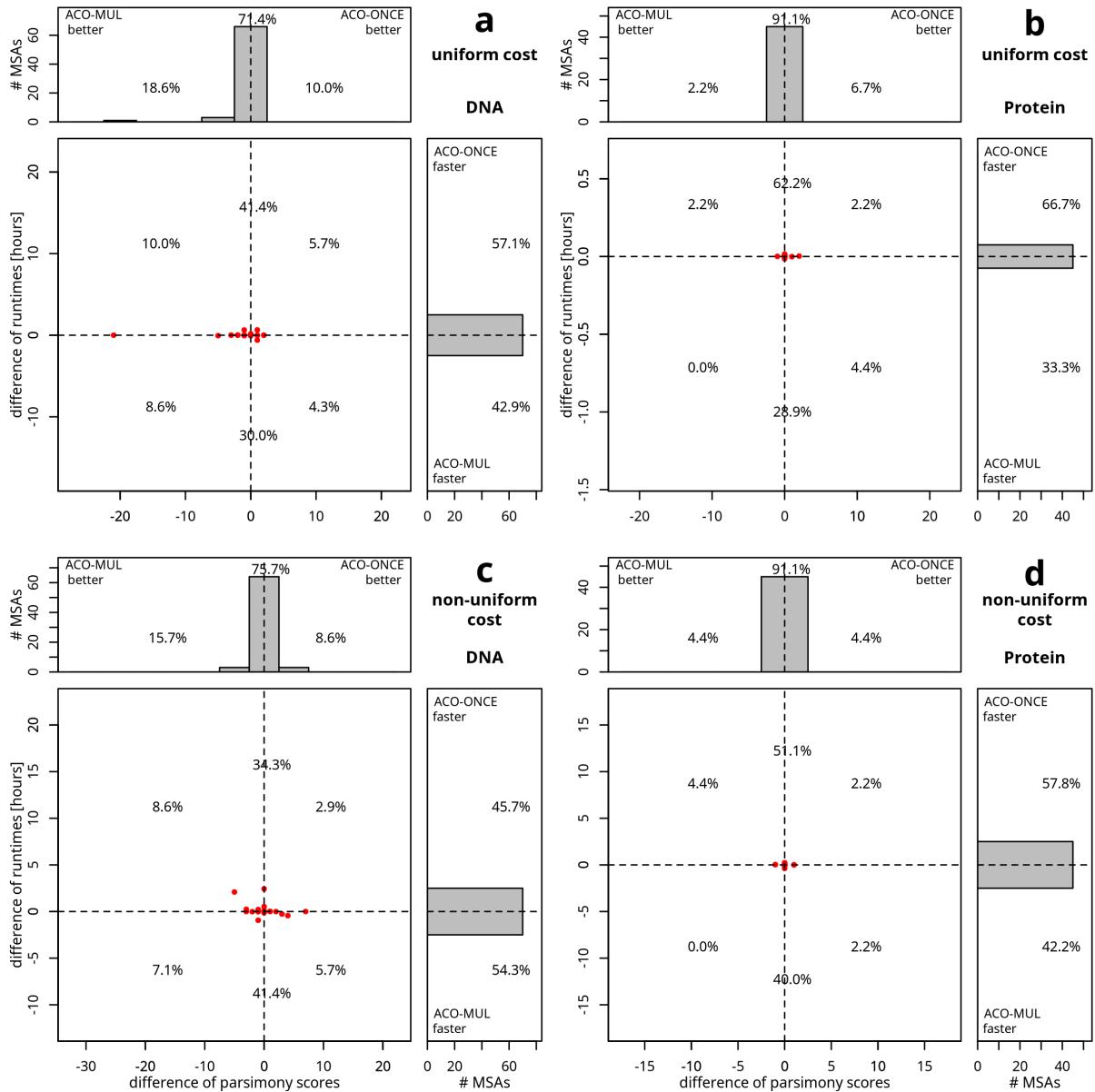
**Hình B.13 — So sánh kết quả của ACO-ONCE với TBR5 trên 115 bộ dữ liệu TreeBASE**



**Hình B.14** — So sánh kết quả của ACO-ONCE với TBR5-BETTER trên 115 bộ dữ liệu TreeBASE



**Hình B.15** — So sánh kết quả của ACO-ONCE với TNT trên 115 bộ dữ liệu TreeBASE



**Hình B.16** — So sánh kết quả của ACO-MUL với ACO-ONCE trên 115 bộ dữ liệu TreeBASE

## C. Kết quả bổ sung của các phiên bản MPBoot-RL với các bộ siêu tham số khác nhau

**Bảng C.1** — Thống kê số lượng bộ dữ liệu đạt được kết quả tốt nhất (trong 115 bộ TreeBASE) của các phương pháp MPBoot-RL với các bộ siêu tham số khác nhau

Cài đặt	Fitch		Sankoff	
	DNA	Protein	DNA	Protein
ACO-MUL, $\rho = 0.1$ , $L_0 = 5$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	48	38	49	40
ACO-MUL, $\rho = 0.15$ , $L_0 = 5$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	49	39	49	42
ACO-MUL, $\rho = 0.2$ , $L_0 = 5$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	46	40	50	42
ACO-MUL, $\rho = 0.25$ , $L_0 = 10$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	49	40	48	40
ACO-MUL, $\rho = 0.25$ , $L_0 = 15$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	50	41	51	42
ACO-MUL, $\rho = 0.3$ , $L_0 = 15$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	49	40	47	42
ACO-ONCE, $\rho = 0.1$ , $L_0 = 5$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	47	40	49	42
ACO-ONCE, $\rho = 0.25$ , $L_0 = 15$ $\eta_{NNI} = 0.3$ , $\eta_{SPR} = 0.4$ , $\eta_{TBR} = 0.4$	48	40	49	40