ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO HỘI NGHỊ SINH VIÊN NGHIÊN CỨU KHOA HỌC

NĂM 2023

Tên đề tài:

COMBINING MULTIPLE TREE REARRANGEMENT OPERATORS FOR EFFICIENT PARSIMONY INFERENCE USING REINFORCEMENT LEARNING APPROACH

NHÓM SINH VIÊN THỰC HIỆN:

1. Huỳnh Tiến Dũng – 21020007 – K66C-CLC
2. Vũ Quốc Tuấn – 21020033 – K66C-CLC
3. Nguyễn Việt Dũng – 21020043 – K66C-A-CLC2

NGƯỜI HƯỚNG DẪN KHOA HỌC:

1. TS. Hoàng Thị Điệp

Năm 2023

Combining Multiple Tree Rearrangement Operators for Efficient Parsimony Inference Using Reinforcement Learning Approach

Tóm tắt

Maximum parsimony phylogenetic boostrapping là bài toán quan trọng trong tin sinh với nhiều ứng dụng trong tiến hóa sinh học và được giải quyết hiệu quả nhờ tiếp cận xấp xỉ trong MPBoot. Trong nghiên cứu này, chúng tôi đề xuất bổ sung phép biến đổi tree bisection and reconnection (TBR) vào MPBoot nhằm cải thiện hiệu năng lấy mẫu trong không gian tìm kiếm và đề xuất thuật toán MPBoot-TBR. Do TBR có tập lân cận độ phức tạp lập phương, chúng tôi đề xuất các kĩ thuật tối ưu thuật toán để đánh giá nhanh một phép TBR, tìm kiếm nhanh lân cận ứng với một cạnh cắt cho trước và leo đồi sử dụng TBR khi tích hợp leo đồi TBR vào MPBoot. Ngoài ra, chúng tôi đề xuất thuật toán MPBoot-ACO kết hợp leo đồi 3 phép biến đổi NNI, SPR, TBR sử dụng giải thuật tối ưu đàn kiến (ACO) nhằm tăng hiệu năng của chương trình. Điều kiện dừng của framework được đề xuất hiệu chỉnh do khi so với subtree pruning and regrafting (SPR), TBR yêu cầu ít lượt lặp tìm kiếm hơn để hội tụ tới một điểm MP đủ tốt. MPBoot-TBR tương đương MPBoot về độ chính xác bootstrap. Đáng chú ý, MPBoot-ACO vượt trội so với bản MPBoot gốc về điểm MP và có ưu thế so với MPBoot-TBR về thời gian tính toán. Chúng tôi đã cài đặt thuật toán đề xuất MPBoot-TBR được công khai mã nguồn tại địa chỉ https://github.com/HynDuf/mpboot/tree/Huynh_Tien_Dung và MPBoot-ACO tại địa chỉ https://github.com/HynDuf/mpboot/tree/ant-colony-optimization.

1 Giới thiệu

1.1 Khái quát chung

Có nhiều phân tích trong sinh học tính toán, ví dụ các phân tích dựa trên đa dạng sinh học, các phân tích về tiến hóa và lây lan của dịch bệnh nguy hiểm do virus gây ra, chỉ có thể đạt hiệu năng tốt nhất khi tác vụ xây dựng cây tiến hóa ở phần lõi được làm tốt nhất [1]. Trên một cây nhị phân biểu diễn lịch sử tiến hóa cho một tập các loài đang sống cho trước, hai loài gần nhau sẽ được đặt ở hai nút lá gần nhau. Mỗi đỉnh trong của cây ứng với một sự kiện phân loài, cũng được hiểu là biểu diễn loài tổ tiên chung gần nhất của các lá trong cây con tương ứng.

Nhu cầu phổ biến trong sinh học là xây dựng cây tiến hóa dựa trên đầu vào có dạng sắp hàng (MSA) của các trình tự sinh học của tập hợp gồm n loài. Nó là một bài toán NP-Complete tìm kiếm tối ưu [2] trong không gian các cây nhị phân n lá. Maximum parsimony là một tiêu chuẩn được chấp nhận rộng rãi để đánh giá độ tốt của cây tiến hóa, theo đó cây tốt nhất là cây giải thích sắp hàng (MSA) đầu vào với chi phí biến đổi thấp nhất [3].

Suy luận tiến hóa hiện đại tiến hành xây dựng cây thường có phân tích bootstrap [4] theo sau nhằm đánh giá độ tin cậy cho từng cạnh theo tần suất của nó trong tập cây bootstrap. Để ngắn gọn, sau đây chúng tôi gọi bài toán xây dựng cây có phân tích bootstrap là phylogenetic bootrapping. Với một phân tích bootstrap B bản sao (thường $B = 1000$) thì phương pháp bootstrap chuẩn (SBS) xây dựng tập cây bootstrap nhờ tiến hành độc lập 1 lượt xây dựng cây tiến hóa cho mỗi bản sao giả lập của sắp hàng đầu vào, do đó, tốn kém về chi phí tính toán. SBS cho parsimony được cài đặt trong TNT [5] và PAUP* [6]. Phương pháp MPBoot 2018 [7] là một tiếp cận mới và hiệu quả để giải bài toán phylogenetic bootstrapping chỉ với 1 lượt xây dựng cây. Trên dữ liệu giả lập, độ tin cậy tính bởi MPBoot cho thấy ít biased hơn SBS. Trong bài báo MPBoot, chúng tôi đã tiến hành thực nghiệm đánh giá hiệu năng về điểm parsimony của TNT theo phương pháp intensive, TNT theo phương pháp fast, MPBoot và PAUP*. Do MPBoot chỉ sử dụng NNI, SPR và ratchet để biến đổi cấu trúc cây trong quá trình tìm kiếm, hiệu năng điểm parsimony của nó đứng thứ hai trong khi intensive TNT đứng thứ nhất.

1.2 Đóng góp

Trong bài báo này, chúng tôi đề xuất bổ sung TBR vào tập các phép biến đổi cây của MPBoot để cải thiện khả năng lấy mẫu không gian cây cho bài toán MP phylogenetic bootstrapping. Trên cơ sở đề xuất và thiết kế 2 thuật toán leo đồi dùng TBR, chúng tôi trình bày thuật toán MPBoot-TBR với 2 tùy chọn leo đồi TBR tích hợp vào thuật toán tree search, thuật toán xấp xỉ bootstrap của MPBoot và giải thuật tối ưu đàn kiến MPBoot-ACO kết hợp các phép biến đổi trong thuật leo đồi. Chúng tôi cài đặt

MPBoot-TBR và MPBoot-ACO nhờ phát triển thư viện PLL [8] và cung cấp miễn phí mã nguồn bản cài đặt của MPBoot-TBR trên Github tại địa chỉ https://github.com/HynDuf/mpboot/tree/Huynh_Tien_Dung, MPBoot-ACO tại địa chỉ https://github.com/HynDuf/mpboot/tree/ant-colony-optimization. Về mặt thực nghiệm, chúng tôi đánh giá các phiên bản mới và MPBoot phiên bản 2018 (sau đây sẽ được gọi là MPBoot gốc) trên 3 thang đo tiêu chuẩn là điểm parsimony, độ chính xác bootstrap và thời gian tính toán trên các bộ dữ liệu benchmark.

2 Cơ sở lý thuyết

2.1 Bài toán MP Phylogenetic Bootstrapping

Trong bài toán xây dựng cây bootstrap tiến hóa theo tiêu chuẩn MP, ta được cho ma trận A^{data} kích thước $n \times m$ lưu sắp hàng (MSA) của n chuỗi và số B là số lượng bản sao bootstrap. Ta cần đi tìm cây T^{best} tốt nhất trên A^{data} và tập hợp B các cây tốt nhất trên các sắp hàng bootstrap A_b $(b = 1, \dots, B)$ theo tiêu chuẩn MP.

Nhờ việc xem mỗi cạnh trên cây tiến hóa là một phân hoạch nhị phân cho tập hợp nút lá, có thể tính cho mỗi cạnh trên cây T^{best} tần suất gặp nó trong tập B. Đây chính là giá trị hỗ trợ bootstrap.

Theo tiêu chuẩn MP, hàm mục tiêu, hay hàm đánh giá độ tốt một cây T, là điểm parsimony, được định nghĩa là chi phí biến đổi tối thiểu để thu được MSA A^{data}, sau đây kí hiệu là $MP(T|A^{data})$ (xem Fig. 1) hay $MP(T)$ nếu bối cảnh của MSA là rõ ràng. Điểm số MP có thể được tính hiệu quả nhờ thuật toán Fitch [9] hoặc thuật toán quy hoạch động Sankoff [10]. Ở đây ta làm việc với các cây nhị phân không gốc, khi tính điểm ta có thể chọn vị trí của gốc trên 1 cạnh bất kì sao cho thuận tiện về mặt tính toán. Việc cạnh nào được chọn không ảnh hưởng điểm số MP.

Tìm kiếm trong không gian các cây nhị phân n lá ra cây T^{best} làm cực tiểu hàm mục tiêu là bài toán NP-Complete [2].

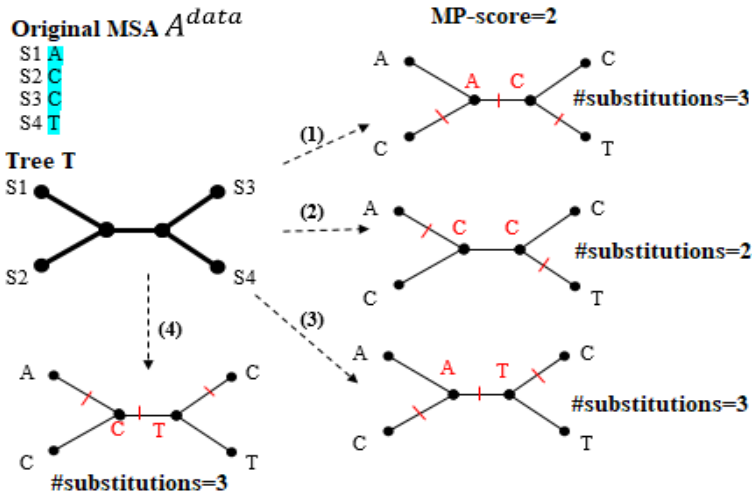

Fig. 1 Minh họa cho ít lượt thay thế nhất (được gọi là Điểm số MP) cần để biểu diễn A^{data} bằng cây T

2.2 Những phép biến đổi cây thường gặp

Với n lớn, các thuật toán khác nhau tuy sử dụng các chiến lược gần đúng khác nhau để lấy mẫu hiệu quả không gian các cây nhị phân n lá. Tuy nhiên, chúng hầu hết dựa trên khai thác 3 phép biến đổi cây với tập lân cận từ nhỏ nhất tới lớn nhất là nearest neighbor interchange (NNI), subtree pruning and regrafting (SPR) và tree bisection and reconnection (TBR) [11].

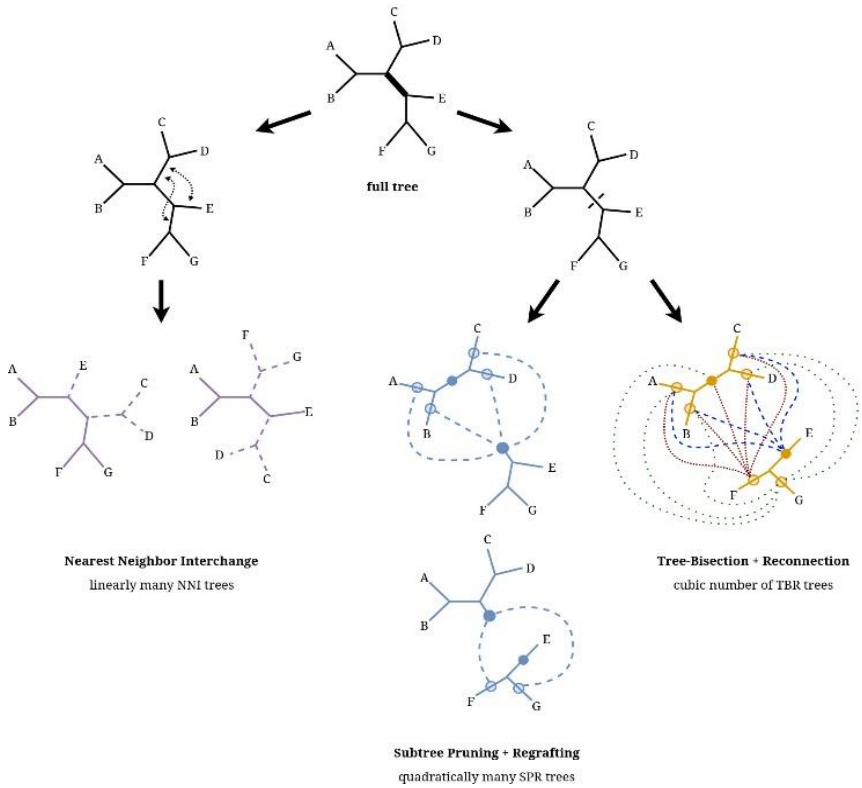

Fig. 2 Ba phép biến đổi cây trên một cây ví dụ gồm 7 taxa

Trên một cạnh xác định, phép NNI sinh ra $O(1)$ lời giải lân cận nhờ hoán đổi các cặp cây con kề với 2 đầu mút của cạnh được chọn (xem Fig. 2). Trên một cạnh cắt xác định, phép SPR sinh ra $O(n)$ lời giải lân cận nhờ ghép gốc trên phần này của cây vào một trong các cạnh của phần kia (xem Fig. 2). Trên một cạnh cắt xác định, phép TBR sinh ra $O(n^2)$ lời giải lân cận nhờ thử đặt gốc trên từng cạnh của phần này của cây rồi ghép vào một trong các cạnh của phần kia (xem Fig. 2).

Để giảm độ phức tạp của SPR và TBR, các thuật toán gần đúng thường chỉ khảo sát điểm ghép trong ngưỡng bán kính nhất định so với điểm cắt. Bán kính SPR là số đỉnh nằm giữa điểm ghép và điểm cắt. Bán kính TBR là tổng của số đỉnh giữa điểm ghép và điểm cắt trên phần này với số đỉnh giữa điểm ghép và điểm cắt trên phần kia.

2.3 Giải thuật tối ưu đàn kiến

Thuật toán ACO (Ant Colony Optimization) - tối ưu đàn kiến - là phương pháp nghiên cứu lấy cảm hứng từ việc mô phỏng hành vi của đàn kiến trong tự nhiên nhằm mục đích giải quyết các bài toán tối ưu phức tạp trong thực tế. Các cá thể kiến trao đổi thông tin trên đường đi thông qua vết mùi (Pheromone) để lại trên đường đi. Các đường đi có nồng độ mùi ít hơn sẽ được loại bỏ, cuối cùng tất cả đàn kiến sẽ đi trên con đường có khả năng trở thành con đường ngắn nhất từ tổ đến nguồn thức ăn.

2.4 Nghiên cứu liên quan

Về độ chính xác bootstrap, SBS như được cài đặt trong TNT, PAUP* và MEGA có xu hướng đánh giá thấp khả năng đúng của một phân hoạch nhị phân [7]; còn phương pháp MPBoot cho giá trị hỗ trợ bootstrap sát với xác suất cạnh đó thuộc về cây đúng. Do đó, giá trị hỗ trợ bootstrap theo MPBoot dễ sử dụng hơn. Một cạnh trên T^{best} được xem là tin cậy khi giá trị hỗ trợ bootstrap cao hơn 70% nếu sử dụng SBS; cao hơn 95% nếu sử dụng MPBoot.

Về điểm số parsimony của cây T^{best}, thực nghiệm trên dữ liệu TreeBASE cho thấy trong các phương pháp hiện tại, MPBoot đứng ngay sau intensive TNT [7]. Điều này là giải thích được vì intensive routine của TNT sử dụng TBR và nhiều kĩ thuật nâng cao như ratchet, sectorial searches, tree fusing, và tree drifting trong việc khám phá không gian cây. MPBoot sử dụng NNI, SPR trong giới hạn bán kính xác định kết hợp với ratchet.

3 Phương pháp

Chúng tôi đề xuất bổ sung phép TBR vào tập các phép biến đổi cây của MPBoot. Việc tích hợp TBR với MPBoot framework được thực hiện qua giải quyết các tác vụ cụ thể: đánh giá nhanh một phép TBR, tìm kiếm lân cận với 2 tham số là cận trên bán kính và cận dưới bán kính TBR, leo đồi sử dụng TBR, tích hợp leo đồi TBR vào MPBoot và

kết hợp 3 phép leo đồi sử dụng NNI, SPR, TBR sử dụng giải thuật tối ưu đàn kiến (ACO).

3.1 Việc áp dụng các phép biến đổi cây trong thuật toán MPBoot gốc

Để giải bài phylogenetic bootstrapping, thuật toán MPBoot duy trì một tập cây C gồm n_C cây tốt nhất tìm được cho tập MSA ban đầu. Tập hợp này được sinh ở pha khởi tạo (pha 1) nhờ chạy 100 lần thủ tục thêm từng bước ngẫu nhiên rồi tối ưu bằng leo đồi SPR và chọn ra n_C cây tốt nhất. Tập hợp C tiếp tục được cải thiện qua pha khám phá (pha 2) nhờ chiến lược lặp phá cây chọn ngẫu nhiên trong C rồi leo đồi SPR trên kết quả. Việc phá cây ở pha khám phá được thực hiện nhờ luân phiên (i) random NNI và (ii) ratchet dùng leo đồi SPR. Ngoài ra, tập cây bootstrap 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.

Như vậy phép biến đổi cây xuất hiện ở cả 3 pha. Việc thay 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^{best} có điểm số MP tốt hơn vừa tìm được tập cây bootstrap tốt hơn.

3.2 Đề xuất tính toán nhanh một phép biến đổi TBR

Xét một phép biến đổi TBR trên cây T^{lst} (xem Fig. 3A). Gọi cạnh R là cạnh cắt của phép TBR. Trong trường hợp tổng quát, cắt cây T^{lst} tại R tạo ra hai cây con T_1, T_2 và cạnh R tạm thời tách biệt với nhau (xem Fig. 3B). Cạnh R sau đó sẽ được dùng làm cạnh trung gian để nối T_1 và T_2 (xem Fig. 3C). 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 (straightforward) thì việc đánh giá lại điểm của cây T^* được thực hiện thuần bằng việc duyệt post-order lại toàn bộ cây và tính theo thuật toán Fitch hoặc Sankoff. Tuy nhiên, trước và sau khi nối tạo thành cây T^* điểm parsimony của nhiều cây con không thay đổi.

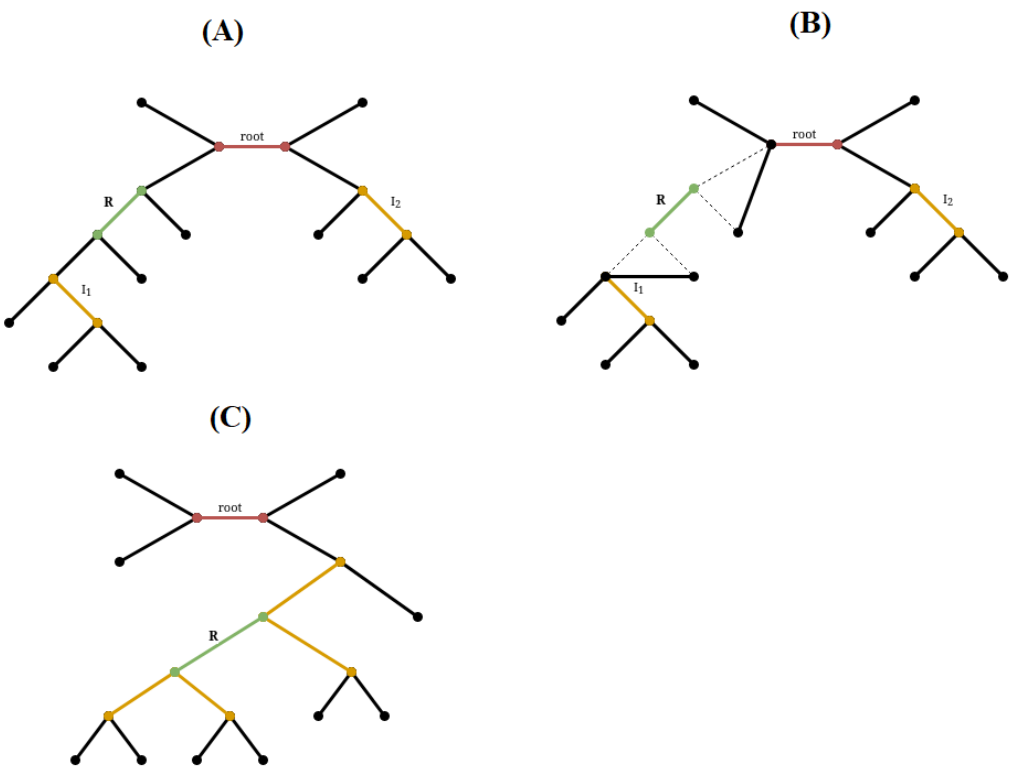

Fig. 3 Một phép biến đổi TBR với cạnh cắt R và 2 cạnh nối I_1 và I_2

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^{lst} đã được tính toán điểm parsimony cho từng đỉnh sử dụng gốc đặt trên cạnh $root$ nối 2 đỉnh $root_1$ và $root_2$. Mỗi đỉnh sẽ lưu điểm parsimony của cây con tương ứng (xem Fig. 4A)**.** 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^{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 Fig. 4B). Để xác định được những đỉnh cần tính lại điểm như trên, với mỗi đỉnh, ta lưu thêm một biến con trỏ tới đỉnh cha ứng với cây T^{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^{lst}. Cuối cùng, trên cây T^* xét gốc ở cạnh R, ta thực hiện tính toán điểm và chỉnh sửa biến con trỏ 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^{lst} cho lượt thử tiếp theo.

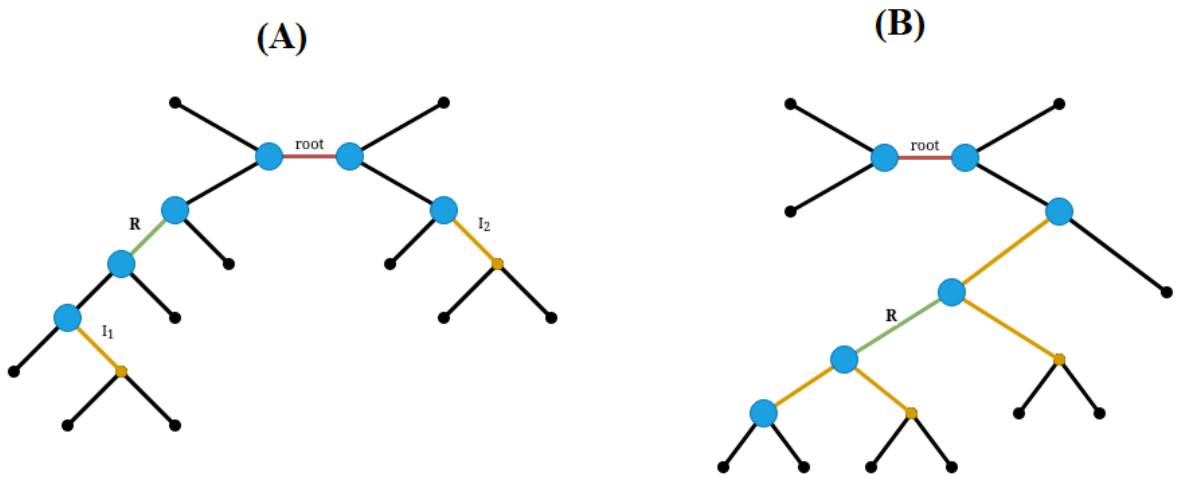

Fig. 4 Nhận diện những đỉnh cần phải tính lại điểm với cạnh cắt R và cạnh nối I_1 và I_2

3.3 Đề xuất tìm kiếm cây lân cận sử dụng TBR với cạnh cắt và bán kính cho trước

3.3.1 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 Fig. 3C)
  - 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 **Algorithm 1** sau đây.

Algorithm 1 Evaluating TBR moves with given remove-branch R on tree T

Input: Tree T,
Remove-branch R,
Radius criteria mintrav and maxtrav for insert-branch I_1 and I_2

Output: Best found tree $T^{bestNei}$ (best (I_1,I_2)) with remove-branch R

```
computeTBR(R, mintrav, maxtrav)
Function testTBRMove(I1,I2)
    # T is already cut into T1, T2 and R when called testTBRMove()
    Connect branch I1 and I2 using R, result in T*
    Evaluate parsimony score MP(T*) of T*
    if MP(T*) < MP(T^bestNei) then
        T^bestNei := T*
        I1^bestNei := I1
        I2^bestNei := I2
    end if
    Remove branch R, rollback the changes
Function computeTBR(R, mintrav, maxtrav)
    Remove branch R from tree T
    # Find all valid (I1,I2) can be done recursively via DFS
    for each (I1,I2) satisfied
        testTBRMove(I1,I2)
    end for
    Reconnect branch R, rollback to T
    T = applyTBR(R, I1^bestNei, I2^bestNei)
```

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 1 đỉnh lá) được xử lí 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 $root$ có thể khác R).

3.3.2 Chiến lược tìm kiếm tốt hơn

Chúng tôi cũng đề xuất một thuật toán tìm kiếm lân cận TBR tương tự **Algorithm 1** nhưng với Chúng tôi cũng đề xuất một thuật toán tìm kiếm lân cận TBR tương tự **Algorithm 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^{best} cho kết quả tốt hơn). Ở thuật toán thay đổi này **(Algorithm 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^{best} và cập nhật cho cây hiện tại nếu cho kết quả tốt hơn.

Algorithm 2 Better improvement strategy

```
1: Function computeTBR(R, mintrav, maxtrav)
2:     for each I1 satisfied
3:         Remove branch R from tree T
4:         for each I2 satisfied
5:             testTBRMove(I1,I2)
6:         end for
7:         Reconnect branch R, rollback to T
8:         T = applyTBR(R, I1, I2^bestNei)
9:     end for
```

3.4 Đề xuất thuật toán leo đồi TBR

Thuật toán leo đồi TBR thực hiện leo đồi cập nhật cây T bằng cây T^{best} tìm được (nếu T^{best} cho kết quả tốt hơn) với mỗi cạnh cắt R khảo sát bằng thuật toán được mô tả ở **Algorithm 1** và **Algorithm 2**. Vòng lặp leo đồi sẽ tiếp tục trong khi cây T vẫn được cập nhật bởi một cây tối ưu hơn.

Thuật toán leo đồi TBR được mô tả bằng mã giả trong **Algorithm 3** sau đây.

Algorithm 3 Hill-climbing algorithm with TBR moves on tree T

Input: Tree T,
Radius criteria mintrav and maxtrav

Output: Tree T updated to best found neighbor tree $T^{bestNei}$ consider every remove-branch R

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

Khi leo đồi sử dụng hai cách tìm kiếm lân cận TBR khác nhau (sau đây gọi tắt là 2 cách leo đồi) thì mẫu không gian cây khảo sát được cũng khác nhau.

3.5 Đề xuất thuật toán MPBoot-TBR

Chúng tôi đề xuất MPBoot-TBR **(Algorithm 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^{best}, thì lượt lặp sẽ được coi là unsuccessful (thất bại). Thuật toán duy trì biến $n_{unsuccess}$ lưu số lượt lặp tìm kiếm liên tiếp unsuccessful (thất bại). MPBoot gốc dừng nếu $n_{unsuccess}$ 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 $n_{unsucess}$ thành $n' = 100$ giống với IQ-TREE [12].

Algorithm 4 The MPBoot-TBR method for bootstrap approximation

Input: an MSA A^{data} with n sequences,
the number of bootstrap MSAs B,
upperbound for TBR radius $maxtrav$

Output: A tree T^{best} with best found $MP(T^{best}|A^{data})$ and a set $\mathcal{B}$ of bootstrap trees $\{T_1, T_2, \dots, T_B\}$

Phase 1. Initialization

Generate bootstrap MSAs and initialize bootstrap tree set $\mathcal{B}$.
Initialize the threshold $MP_{max} := +\infty$.
Initialize the candidate set C for A^{data} with 100 random stepwise addition procedures followed by TBR hill-climbing

Phase 2. Exploration

do
  Improve C by performing the perturbation on a randomly selected tree from the candidate set C and a subsequent TBR hill-climbing step. Every time a new tree, T, with $MP(T|A^{data}) < MP_{max}$ is encountered, execute REPS to update bootstrap tree set $\mathcal{B}$.
  Update T^{best}, MP_{max}, $n_{unsuccess}$
while $n_{unsuccess} < n'$
end do

Phase 3. Bootstrap refinement

For each MP-tree T_b , do a hill-climbing TBR search and replace T_b by the new MP tree, if a better parsimony score is found.
output T^{best} the best MP tree that was found for A^{data}.
output set $\mathcal{B}$ and/or map the support values onto T^{best}.

3.6 Đề xuất thuật toán MPBoot-ACO

Chúng tôi đề xuất MPBoot-ACO thay thế việc leo đồi chỉ sử dụng duy nhất SPR hay TBR bằng giải thuật đàn kiến (ACO) xác định phép leo đồi phù hợp (trong các phép NNI, SPR, TBR) ở mỗi lượt leo đồi. Thuật toán duy trì 3 giá trị "pheromone" cho 3 phép leo đồi, lần lượt là $\tau_{NNI}, \tau_{SPR}, \tau_{TBR}$. Giá trị "pheromone" này lớn đồng nghĩa với phép leo đồi tương ứng đang có hiệu quả cao trong các lượt leo đồi gần nhất. Ngoài ra, 3 tham số heuristic $\eta_{NNI}, \eta_{SPR}, \eta_{TBR}$ được khởi tạo giá trị trước thể hiện sự ưu tiên ban đầu đối với từng phép leo đồi. Khi đó, ở mỗi lượt leo đồi, việc chọn phép biến đổi cây tương ứng tuân theo hàm xác suất sau:

$$p_{NNI} = \frac{\tau_{NNI}\,\eta_{NNI}}{\Sigma\,\tau_i\,\eta_i}; \qquad p_{SPR} = \frac{\tau_{SPR}\,\eta_{SPR}}{\Sigma\,\tau_i\,\eta_i}; \qquad p_{TBR} = \frac{\tau_{TBR}\,\eta_{TBR}}{\Sigma\,\tau_i\,\eta_i}$$

Sau mỗi UPDATE_ITERS lượt leo đồi, pheromone của 3 phép biến đổi sẽ được cập nhật như sau: $\tau_i = (1-\rho)\tau_i + \Delta\tau_i$ với ρ là hệ số bay hơi pheromone và $\Delta\tau_i$ là lượng pheromone thay đổi sau UPDATE_ITERS lượt leo đồi. Giả sử lượt leo đồi sử dụng SPR (tương tự với các phép biến đổi khác) và kết quả leo đồi tìm được một cây có điểm số MP tốt hơn điểm số tốt nhất. Khi đó, $\Delta\tau_{SPR} = \Delta\tau_{SPR} + 1$, ngược lại khi SPR không cho điểm số tốt hơn, cập nhật pheromone cho 2 phép còn lại $\Delta\tau_{TBR} = \Delta\tau_{TBR} + 0.5$, $\Delta\tau_{NNI} = \Delta\tau_{NNI} + 0.5$. Các $\Delta\tau_i$ được đặt lại về 0 sau mỗi lần cập nhật pheromone (hay là sau mỗi UPDATE_ITERS lượt leo đồi).

3.7 Cài đặt

Chúng tôi cài đặt MPBoot-TBR 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 [8].

Dòng lệnh chạy MPBoot-TBR tìm kiếm cây MP trên sắp hàng gốc lưu tại <alignment_file> sử dụng leo đồi TBR-best có cú pháp như sau:

mpboot -s <alignment_file> -tbr_pars

Cú pháp để tìm kiếm với phương án leo đồi TBR-better:

mpboot -s <alignment_file> -tbr_pars -tbr_restore_ver2

Cú pháp để tìm kiếm với thuật toán MPBoot-ACO (mặc định sử dụng TBR-best):

mpboot -s <alignment_file> -do_ant_colony

Để thực hiện xấp xỉ bootstrap B bản sao đồng thời với tìm kiếm cây MP, hãy thêm vào dòng lệnh chỉ định "-bb <B>".

4 Kết quả thực nghiệm

4.1 Cài đặt thực nghiệm

Chúng tôi so sánh 2 thuật toán MPBoot-TBR dựa trên chiến lược tìm kiếm lân cận tốt nhất (kí hiệu MPBoot-TBR-best) và dựa trên chiến lược tìm kiếm lân cận tốt hơn (kí hiệu MPBoot-TBR-better) đề xuất với thuật toán MPBoot gốc. Việc 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 ĐH Công Nghệ, ĐHQG HN. MPBoot-TBR được khảo sát với maxtrav=5 và điều kiện dừng cũ (như trong MPBoot gốc) bởi vì chúng tôi quan sát thấy cài đặt bán kính này có thời gian chạy tương đương với MPBoot gốc. Ngoài ra, chúng tôi cũng chạy MPBoot-TBR với $n' = 100$. Để thuận tiện, chúng tôi nối thêm "sc100" vào kí hiệu phương pháp để chỉ việc dùng điều kiện dừng này. Bên cạnh đó, chúng tôi cũng chạy MPBoot-TBR với điều kiện dừng của MPBoot gốc. MPBoot gốc sử dụng

chiến lược leo đồi với SPR bán kính 6. Việc phá cây bằng parsimony ratchet trong những iteration chẵn được dùng như nhau trong tất cả các phương pháp.

Ngoài ra, chúng tôi so sánh MPBoot-ACO với MPBoot-TBR-best (gọi tắt là MPBoot-TBR trong phần kết quả) và MPBoot gốc. Việc thực nghiệm được thực hiện trên bộ dữ liệu sinh học TreeBASE, lấy kết quả trung bình về điểm số và thời gian thực thi sau 5 lần chạy ngẫu nhiên mỗi phương pháp. Chúng tôi chạy thuật toán tree search (không phải bootstrap) và sử dụng điều kiện dừng của MPBoot gốc. Các tham số chạy MPBoot-ACO đặt ban đầu bao gồm số lượt leo đồi trước mỗi lần cập nhật UPDATE_ITERS=30, hệ số bay hơi pheromone ρ = EVAPORATION_RATE = 0.6, độ ưu tiên $\eta_{NNI}, \eta_{SPR}, \eta_{TBR} = 0.3, 0.35, 0.35$. Chúng tôi đã khảo sát MPBoot-ACO với nhiều bộ giá trị khác nhau và các giá trị trên cho kết quả và thời gian phù hợp nhất. Các tham số chạy MPBoot-TBR-best (MPBoot-TBR) và MPBoot gốc được đặt như đã đề cập ở phần so sánh trên.

4.2 Dữ liệu

4.2.1 Dữ liệu mô phỏng

Chúng tôi sử dụng bộ dữ liệu mô phỏng benchmark, gồm 3 bộ dữ liệu DNA và 2 bộ dữ liệu protein, mỗi bộ chứa 200 MSAs (xem Table 1), được sinh từ cây theo mô hình Yule Harding [13]. Đây là phương tiện đánh giá độ chính xác bootstrap đã được dùng trong các nghiên cứu xây dựng cây bootstrap trước đây [14]–[16]

4.2.2 Dữ liệu sinh học

Chúng tôi sử dụng bộ dữ liệu TreeBASE gồm 70 sắp hàng DNA và 45 MSA protein (xem Table 1) được thu thập và chọn lọc từ cơ sở dữ liệu trực tuyến TreeBASE [17] bởi Nguyen và cộng sự [18].

Table 1 Tóm tắt các bộ dữ liệu

Dataset	Sub-dataset	Data type	#MSAs	#sequences	#sites
YuleHarding	YH1	DNA	200	100	500
	YH2		200	200	1000
	YH3		200	500	1000
	YH4	Protein	200	100	300
	YH5		200	200	500
TreeBASE	TreeBASE-dna	DNA	70	201-767	976-61199

	TreeBASE-prot	Protein	45	50-194	126-22426

4.3 Tiêu chí đánh giá

4.3.1 Điểm MP

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

4.3.2 Độ chính xác Bootstrap

Chúng tôi cũng so sánh chất lượng tập 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 $\mathrm{T}^{\mathrm{best}}$) [19].

4.3.3 Thời gian thực thi

Với thước đo thứ 3 là thời gian thực thi, chúng tôi quan sát tổng thời gian (tiếng) phân tích bootstrap và thời gian cho mỗi vòng lặp tìm kiếm của từng phương pháp.

4.4 Kết quả

4.4.1 Điểm MP

Như đã tổng hợp ở Table 2, trên dữ liệu mô phỏng YuleHarding thì MPBoot-TBR* và MPBoot cho điểm số MP giống hệt nhau.

Trên dữ liệu thực TreeBASE thì thuật toán MPBoot-TBR cho điểm MP tốt hơn. Đáng chú ý MPBoot-TBR-sc100 cho kết quả tương đương MPBoot-TBR.

Table 2 Tần số đạt được điểm MP tốt nhất của các phiên bản MPBoot

Dataset	Sub-dataset	MPBoot	MPBoot-TBR-best	MPBoot-TBR-better	MPBoot-TBR-best-sc100	MPBoot-TBR-better-sc100
YuleHarding	YH1	100%	100%	100%	100%	100%
	YH2	100%	100%	100%	100%	100%
	YH3	100%	100%	100%	100%	100%

	YH4	100%	100%	100%	100%	100%
	YH5	100%	100%	100%	100%	100%
TreeBASE	TreeBASE-dna	63%	80%	74%	70%	74%
	TreeBASE-prot	87%	89%	96%	98%	93%

4.4.2 Độ chính xác Bootstrap

Hàm $f_{MPBoot}(v)$ (đường màu đen), hàm $f_{MPBoot-TBR-best-sc100}(v)$ (đường màu xanh) và hàm $f_{MPBoot-TBR-best-sc100}(v)$ (đường màu vàng) cho 5 bộ YuleHarding được minh họa ở Fig. 5. 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.

4.4.3 Thời gian thực thi

Tổng thời gian chạy từng phương pháp trên mỗi dataset được tổng hợp trên Table 3. Đáng chú ý, MPBoot-TBR-best-sc100 và MPBoot-TBR-better-sc100 cho thời gian tốt hơn bản MPBoot gốc.

Table 3 Tổng thời gian thực thi của MPBoot và các phiên bản MPBoot-TBR

Dataset	Time summation over all MSAs (in hours)				
	Original MPBoot	TBR-best	TBR-better	TBR-best-sc100	TBR-better-sc100
YuleHarding	38.9	81.4	104.5	39	48.7
TreeBASE	27.2	56.1	59.9	22.7	28.1

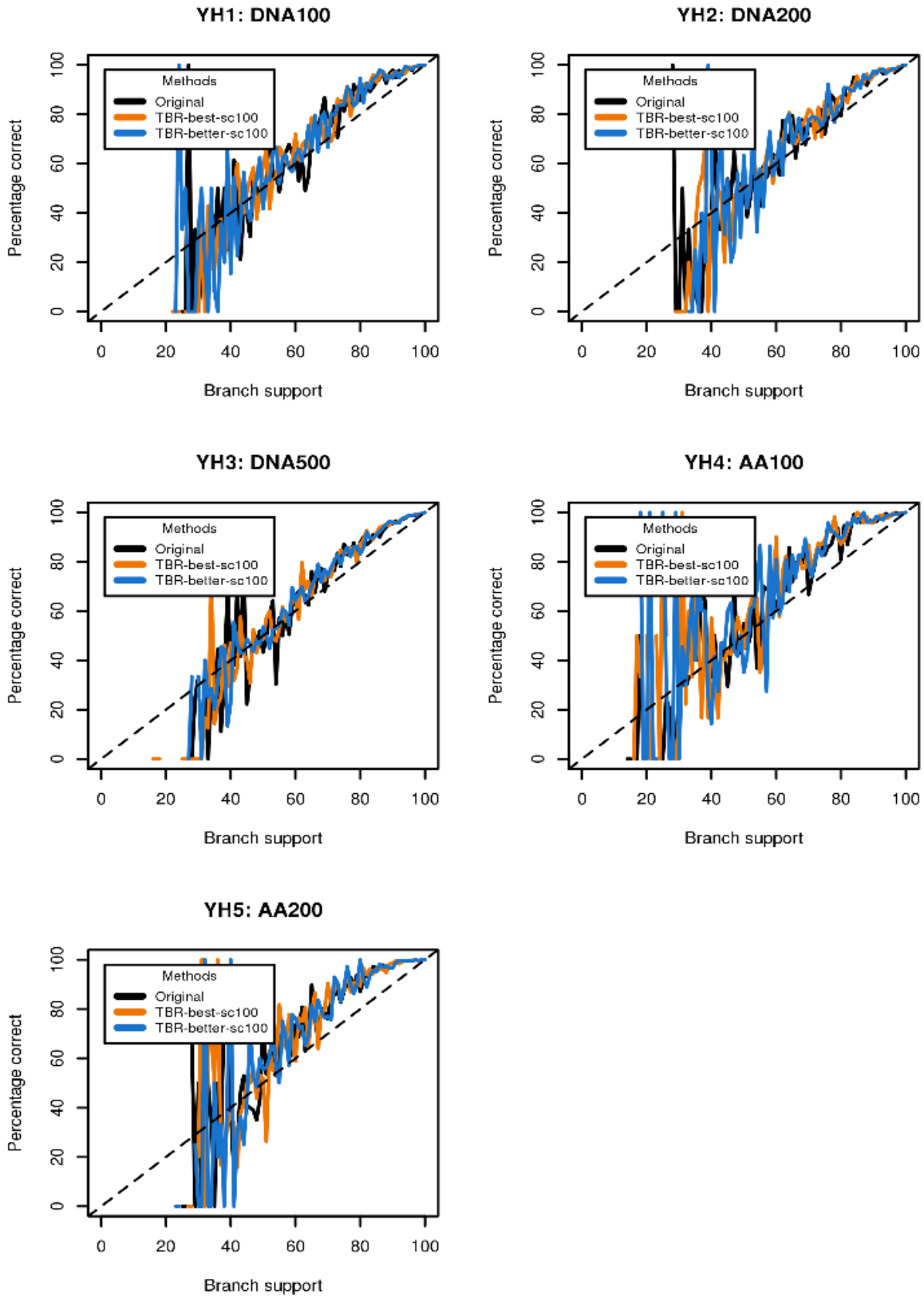

Fig. 5 Minh họa độ chính xác bootstrap trên 5 bộ dữ liệu Yule-Harding của MPBoot gốc và các MPBoot-TBR khác

4.4.4 So sánh MPBoot-ACO, MPBoot và MPBoot-TBR

Độ chính xác về điểm số MP và tổng thời gian của thời gian thực thi trung bình trên 115 bộ dữ liệu của TreeBASE với MPBoot và MPBoot-TBR-best (sau đây gọi tắt là MPBoot-TBR) được tổng hợp ở Table 4.

MPBoot-ACO cho điểm tốt hơn MPBoot gốc và MPBoot-TBR với thời gian chạy chậm hơn MPBoot gốc nhưng nhanh hơn MPBoot-TBR thể hiện sự vượt trội so với MPBoot-TBR.

Table 4 Độ chính xác điểm số và tổng thời gian chạy trung bình trên bộ dữ liệu TreeBASE của MPBoot, MPBoot-TBR và MPBoot-ACO

	MPBoot	MPBoot-TBR	MPBoot-ACO
Số bộ dữ liệu đạt điểm trung bình tốt nhất (trên 115 bộ)	90	92	96
Tổng thời gian trung bình (phút)	227.1	279.1	249.8

Ở Table 5, thông qua việc thống kê tỉ lệ sử dụng các loại leo đồi của MPBoot-ACO ở những bộ liệu với đặc điểm khác nhau, ta có thể thấy được ưu điểm của MPBoot-ACO. Về trung bình, MPBoot-ACO sử dụng 16.4% NNI, 39.1% SPR và 44.4% TBR. Ở những bộ dữ liệu mà MPBoot-ACO cho điểm số tốt hơn MPBoot-TBR, MPBoot-ACO sử dụng nhiều lượt leo đồi SPR hơn và ít lượt NNI hơn. Ở những bộ dữ liệu điểm số bằng nhau và MPBoot-ACO có thời gian tính toán nhanh hơn MPBoot-TBR, tỉ lệ sử dụng leo đồi NNI tăng lên trong khi SPR và TBR giảm xuống. Đặc biệt, ở những bộ protein (những bộ đơn giản trong TreeBASE) mà MPBoot-ACO cho điểm số tương đương và thời gian nhanh hơn MPBoot-TBR, tỉ lệ sử dụng leo đồi NNI tăng lên nhiều so với trung bình. Qua đó cho thấy MPBoot-ACO lựa chọn leo đồi phù hợp với độ khó/dễ của dữ liệu, tăng hiệu năng của chương trình.

Table 5 Chi tiết tỉ lệ sử dụng các loại leo đồi trong bộ dữ liệu TreeBASE của MPBoot-ACO

Mẫu đánh giá	%NNI	%SPR	%TBR
Trên toàn bộ TreeBASE	16.4	39.1	44.4
Trên các bộ điểm ACO tốt hơn TBR	13.9	43.4	42.7
Trên các bộ điểm ACO bằng TBR và thời gian ACO nhanh hơn TBR	19.7	39.7	40.6

Trên các bộ protein điểm ACO bằng TBR và thời gian ACO nhanh hơn TBR	22.1	39.3	38.6

5 Bình luận và Kết luận

Chúng tôi đã đề xuất thuật toán MPBoot-TBR nhờ nghiên cứu kĩ phép biến đổi TBR và khai thác TBR để cải thiện hiệu năng lấy mẫu trong không gian tìm kiếm cho MPBoot framework [7]. Theo đó, thuật toán mới có mục tiêu cải thiện đồng thời điểm số cây MP và tập cây bootstrap. Để đảm bảo độ linh hoạt của framework hướng tới phân tích dữ liệu lớn, chúng tôi thiết kế phép tìm kiếm lân cận TBR có 2 tham số là cận trên bán kính và cận dưới bán kính. Ngoài ra, chúng tôi đề xuất thuật toán MPBoot-ACO dựa trên phép biến đổi TBR trong MPBoot-TBR và 2 phép biến đổi có sẵn NNI và SPR. Thuật toán kết hợp này nhằm cải thiện điểm số và thời gian chạy bằng cách khai thác điểm mạnh của các phương án leo đồi khác nhau.

Hiệu năng của thuật toán đề xuất được thực hiện thông qua các đề xuất nhằm tối ưu các tác vụ cụ thể, bao gồm: đánh giá nhanh một phép TBR, tìm kiếm nhanh lân cận ứng với một cạnh cắt cho trước, leo đồi sử dụng TBR, tích hợp leo đồi TBR và kết hợp 3 phương án leo đồi vào MPBoot. Để việc tính điểm cho cây sau khi làm TBR tránh được việc tính điểm của những cây con không thay đổi điểm số, chúng tôi đề xuất kết hợp ý tưởng đánh dấu đỉnh và ý tưởng đổi gốc. Ý tưởng đổi gốc cũng được vận dụng linh hoạt để tối thiểu số lượng đỉnh cần tính lại điểm giữa 2 lần thử ghép liên tiếp ứng với 1 cạnh cắt giảm số lượng đỉnh cần tính lại từ $O(n)$ xuống $O(maxtrav)$. Chúng tôi khảo sát 2 chiến lược tìm kiếm lân cận, gọi là TBR-best và TBR-better và hiệu chỉnh điều kiện dừng của phần mềm dựa trên sự hội tụ tới điểm MP tốt của TBR. Cuối cùng, chúng tôi kết hợp phương án leo đồi TBR-best với leo đồi SPR, NNI có sẵn ở MPBoot, sử dụng giải thuật đàn kiến (ACO) tạo ra thuật toán MPBoot-ACO. Do MPBoot gốc đã so sánh với những công cụ tính toán khác một cách chi tiết [7], bài báo này tập trung vào so sánh những phiên bản MPBoot-TBR, MPBoot-ACO với MPBoot gốc nhằm nhấn mạnh tác động của các đề xuất liên quan tới TBR.

Hiệu năng theo độ chính xác bootstrap của các phiên bản MPBoot-TBR tương đương với MPBoot trên tất cả các cấu hình mô phỏng YH. Về điểm MP, trên các bộ dữ liệu mô phỏng YuleHarding và bộ dữ liệu thực TreeBASE, các phiên bản MPBoot-TBR cho kết quả tốt hơn hoặc tương đương với bản MPBoot gốc. Các bộ dữ liệu YH được mô phỏng với mô hình tiến hóa đơn giản để nhấn mạnh xấp xỉ bootstrap với nhiều kiểu dữ liệu và kích thước đầu vào khác nhau hơn là để nhấn mạnh hiệu năng thuật toán tìm kiếm. Vì vậy, việc tìm kiếm cây có điểm số MP tốt nhất trên bộ dữ liệu YH dễ hơn so với bộ dữ liệu thực tế TreeBASE. Mọi phương án đề xuất đạt 100% điểm số MP tốt nhất trên các bộ YH thể hiện việc vượt qua kiểm tra sơ bộ. Đặc biệt, các phiên bản MPBoot-TBR chạy với $n' = 100$ và chiến thuật cải thiện tốt nhất (MPBoot-TBR-best)

cho thời gian tốt hơn bản gốc trên dữ liệu thực tế. Chúng tôi chọn maxtrav=5 để cân bằng hiệu năng giữa điểm số MP và thời gian chạy sau khi khảo sát TBR-best với các cài đặt bán kính khác nhau. Bán kính 3 xử lý bộ TreeBASE trong 11.7 tiếng, nhanh gấp 2 lần những điểm số tệ hơn MPBoot gốc. Bán kính 7 xử lý bộ TreeBASE trong 54.6 tiếng nhưng điểm số tương đương với bán kính 5 với điều kiện n' cũ.

MPBoot-ACO cho kết quả vượt trội so với MPBoot gốc và MPBoot-TBR-best về điểm số MP. Không những thế, MPBoot-ACO cũng vượt trội hơn so với MPBoot-TBR-best về thời gian thực thi. Ở những bộ dữ liệu dễ, MPBoot-ACO lựa chọn các phép leo đồi nhằm tăng hiệu năng chương trình (do những phép leo đồi mạnh sau khi đã tìm được điểm MP tối ưu sẽ không phát huy được hiệu quả). Qua đó thể hiện sự khai thác thế mạnh hợp lý của các phép biến đổi cây của MPBoot-ACO.

Trong các nghiên cứu sắp tới, chúng tôi sẽ nghiên cứu cải tiến MPBoot-TBR ở cả 2 thang đo điểm MP và thời gian chạy. Thứ nhất, ta có thể tối ưu việc tính lại điểm của cây sao cho độ phức tạp tính toán không phụ thuộc vào bán kính maxtrav, từ đó có thể khảo sát được những cạnh nối ở xa cạnh cắt mà không tăng thêm chi phí, tối ưu có thể áp dụng cho cả TBR và SPR. Thứ hai, ta có thể nghiên cứu cải tiến cách duyệt khi leo đồi để tránh việc khảo sát nhiều phép TBR dẫn tới cùng một cấu trúc cây. Thứ ba, ta có thể nghiên cứu điều chỉnh hạn chế tập cây đưa vào làm REPS để vẫn đảm bảo độ chính xác bootstrap nhưng khảo sát ít cây hơn.

Chúng tôi đã cài đặt thuật toán đề xuất trong phần mềm MPBoot-TBR được công khai mã nguồn tại địa chỉ https://github.com/HynDuf/mpboot/tree/Huynh_Tien_Dung và MPBoot-ACO tại địa chỉ https://github.com/HynDuf/mpboot/tree/ant-colony-optimization.

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.

Tài liệu tham khảo

[1] J. Herron and S. Freeman, *Evolutionary Analysis*, 5th ed. Pearson, 2013. doi: 10.1163/156854009X409126.

[2] R. L. Graham and L. R. Foulds, "Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time," *Math Biosci*, vol. 60, no. 2, pp. 133–142, Aug. 1982, doi: 10.1016/0025-5564(82)90125-0.

[3] S. Wooding, "Inferring Phylogenies," *Am J Hum Genet*, vol. 74, p. 1074, May 2004.

[4] J. Felsenstein, "CONFIDENCE LIMITS ON PHYLOGENIES: AN APPROACH USING THE BOOTSTRAP," *Evolution (N Y)*, vol. 39, no. 4, pp. 783–791, Jul. 1985, doi: 10.1111/j.1558-5646.1985.tb00420.x.

[5] P. A. Goloboff, J. S. Farris, and K. C. Nixon, "TNT, a free program for phylogenetic analysis," *Cladistics*, vol. 24, no. 5, pp. 774–786, Oct. 2008, doi: 10.1111/j.1096-0031.2008.00217.x.

[6] D. Swofford, "PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods)," 2002, doi: 10.1111/j.0014-3820.2002.tb00191.x.

[7] D. T. Hoang, L. S. Vinh, T. Flouri, A. Stamatakis, A. von Haeseler, and B. Q. Minh, "MPBoot: fast phylogenetic maximum parsimony tree inference and bootstrap approximation," *BMC Evol Biol*, vol. 18, no. 1, p. 11, Dec. 2018, doi: 10.1186/s12862-018-1131-3.

[8] T. Flouri *et al.*, "The Phylogenetic Likelihood Library," *Syst Biol*, vol. 64, no. 2, pp. 356–362, Mar. 2015, doi: 10.1093/sysbio/syu084.

[9] W. M. Fitch, "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology," *Syst Zool*, vol. 20, no. 4, p. 406, Dec. 1971, doi: 10.2307/2412116.

[10] D. Sankoff, "Minimal Mutation Trees of Sequences," *SIAM J Appl Math*, vol. 28, no. 1, pp. 35–42, Jan. 1975, doi: 10.1137/0128004.

[11] P. Lemey, *The Phylogenetic Handbook*, 2nd ed. Cambridge: Cambridge University Press, 2009. doi: 10.1017/CBO9780511819049.

[12] B. Q. Minh *et al.*, “IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era,” *Mol Biol Evol*, vol. 37, no. 5, pp. 1530–1534, May 2020, doi: 10.1093/molbev/msaa015.

[13] E. F. Harding, “The probabilities of rooted tree-shapes generated by random bifurcation,” *Adv Appl Probab*, vol. 3, no. 1, pp. 44–77, Jul. 1971, doi: 10.2307/1426329.

[14] B. Q. Minh, M. A. T. Nguyen, and A. von Haeseler, “Ultrafast Approximation for Phylogenetic Bootstrap,” *Mol Biol Evol*, vol. 30, no. 5, pp. 1188–1195, May 2013, doi: 10.1093/molbev/mst024.

[15] D. T. Hoang, L. S. Vinh, T. Flouri, A. Stamatakis, A. von Haeseler, and B. Q. Minh, “A new phylogenetic tree sampling method for maximum parsimony bootstrapping and proof-of-concept implementation,” in *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, Oct. 2016, pp. 1–6. doi: 10.1109/KSE.2016.7758020.

[16] N. Pham and D. T. Hoang, “A distributed algorithm for the parsimony bootstrap approximation,” in *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*, Nov. 2021, pp. 1–6. doi: 10.1109/KSE53942.2021.9648648.

[17] M. J. Sanderson, M. J. Donoghue, W. Piel, and T. Eriksson, “TreeBASE: a prototype database of phylogenetic analyses and an interactive tool for browsing the phylogeny of life,” *Am J Bot*, vol. 81, p. 183, 1994.

[18] L.-T. Nguyen, H. A. Schmidt, A. von Haeseler, and B. Q. Minh, “IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies,” *Mol Biol Evol*, vol. 32, no. 1, pp. 268–274, Jan. 2015, doi: 10.1093/molbev/msu300.

[19] D. M. Hillis and J. J. Bull, “An Empirical Test of Bootstrapping as a Method for Assessing Confidence in Phylogenetic Analysis,” *Syst Biol*, vol. 42, no. 2, pp. 182–192, Jun. 1993, doi: 10.1093/sysbio/42.2.182.