Local Search Algorithms

(ARTIFICIAL INTELLIGENCE)

Local Search

- Trong nhiều vấn đề, đường dẫn đến trạng thái mục tiêu là không liên quan.
- Nếu đường dẫn đến mục tiêu không quan trọng, chúng ta có thể xem xét một lớp thuật toán khác không quan tâm đến đường dẫn chút nào.

=> thuật toán tìm kiếm cục bộ (Local search)

Local Search

- Trong nhiều vấn đề, đường dẫn đến trạng thái mục tiêu là không quan trọng, bản thân trạng thái mục tiêu là giải pháp.
- Nếu đường dẫn đến mục tiêu không quan trọng, chúng ta có thể xem xét một lớp thuật toán khác không quan tâm đến đường dẫn chút nào.

=> thuật toán tìm kiếm cục bộ (Local search)

- Local search sẽ duy trì một trạng thái "hiện tại" duy nhất, cố gắng cải thiện trạng thái đó => Không gian trạng thái sẽ là hằng số

Hill Climbing Algorithm

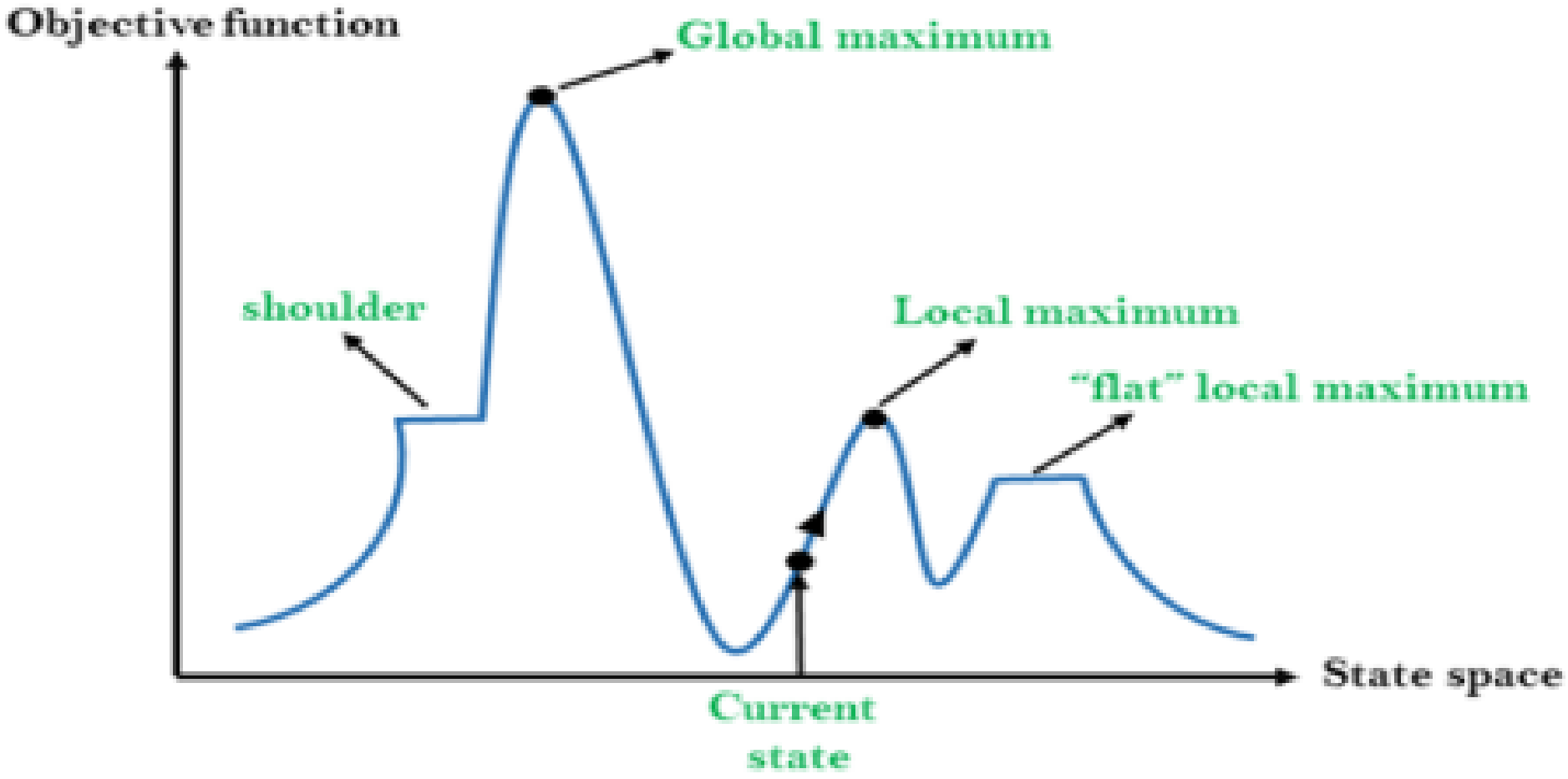
- Là thuật toán tìm kiếm cục bộ
- Trạng thái liên tục di chuyển theo hướng tăng độ cao/giá trị để tìm đỉnh núi hoặc giải pháp tốt nhất cho bài toán.
- Thuật toán kết thúc khi đạt đến giá trị đỉnh mà không có hàng xóm nào có giá trị cao hơn.
- Ví dụ: Bài toán người bán hàng du lịch trong đó chúng ta cần giảm thiểu khoảng cách mà người bán hàng di chuyển.

Hill Climbing Algorithm

- Thuật toán này còn được gọi là tìm kiếm cục bộ tham lam vì nó chỉ tìm kiếm trạng thái lân cận tốt của nó chứ không phải vượt ra ngoài trạng thái đó.
- Một nút của thuật toán leo đồi có hai thành phần là trạng thái và giá trị.
- Thuật toán leo đồi chủ yếu được sử dụng khi có phương pháp tìm kiếm tốt.
- Trong thuật toán này, chúng ta không cần phải duy trì và xử lý cây tìm kiếm hoặc đồ thị vì nó chỉ giữ một trạng thái hiện tại duy nhất.

Hill Climbing

- Vấn đề: tùy thuộc vào trạng thái ban đầu, có thể bị kẹt ở cực đại cục bộ



Các loại Hill Climbing

- Simple hill Climbing (Leo đồi đơn giản)
- Steepest-Ascent hill-climbing (Leo đồi dốc nhất)
- Stochastic hill Climbing (Leo đồi ngẫu nhiên)

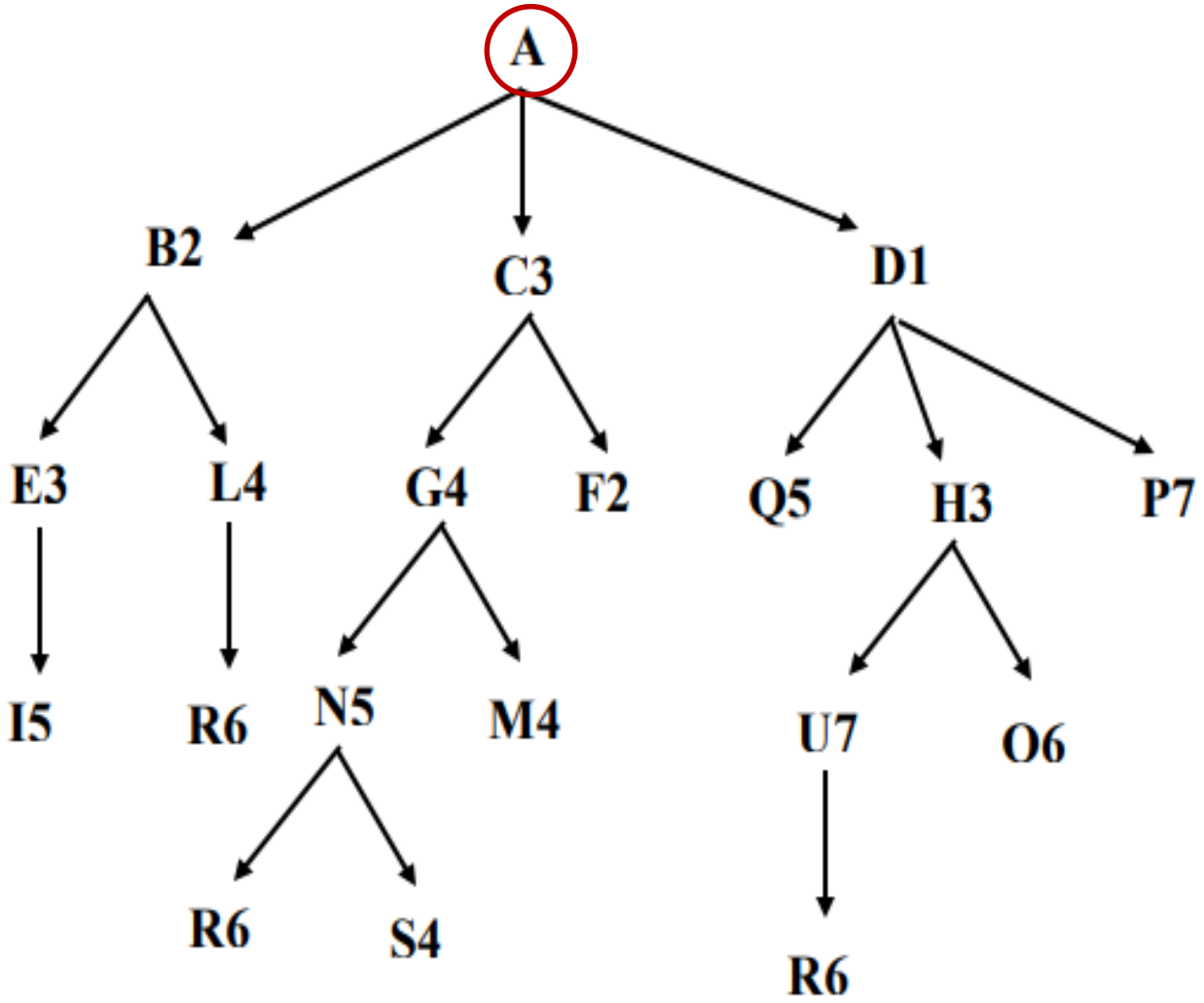
Simple hill Climbing (Leo đồi đơn giản)

- Là cách đơn giản nhất để triển khai thuật toán leo đồi
- Chỉ kiểm tra từng trạng thái lận cận của nó và nếu nó tìm thấy trạng thái tốt hơn trạng thái hiện tại thì di chuyển

Simple hill Climbing (Leo đồi đơn giản)

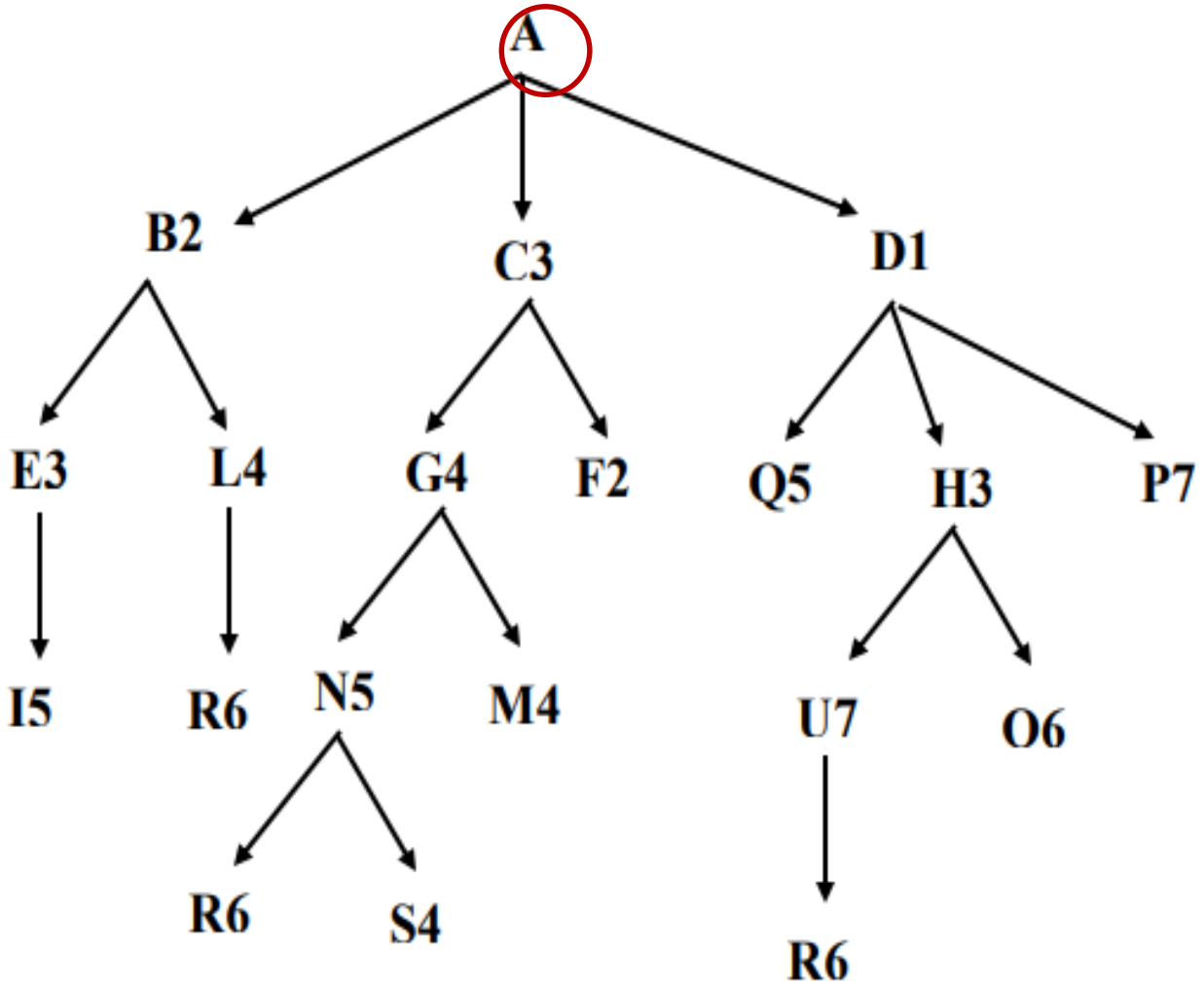
BƯỚC 1: Chọn trạng thái hiện tại
BƯỚC 2: Tạo một hàng xóm của trạng thái hiện tại
BƯỚC 3: Đánh giá hàm mục tiêu tại hàng xóm được đề xuất
BƯỚC 4: Nếu giá trị của hàm mục tiêu tại hàng xóm được đề xuất tốt hơn tại trạng thái hiện tại, thì trạng thái hàng xóm được đề xuất sẽ trở thành trạng thái hiện tại
BƯỚC 5: Lặp lại BƯỚC 2-BƯỚC 4 trong n lần lặp hoặc cho đến khi giá trị của mục tiêu tại trạng thái hiện tại cao hơn tất cả các hàng xóm.
BƯỚC 6: Trả về trạng thái hiện tại và giá trị hàm mục tiêu của nó

Simple hill Climbing (Leo đồi đơn giản)



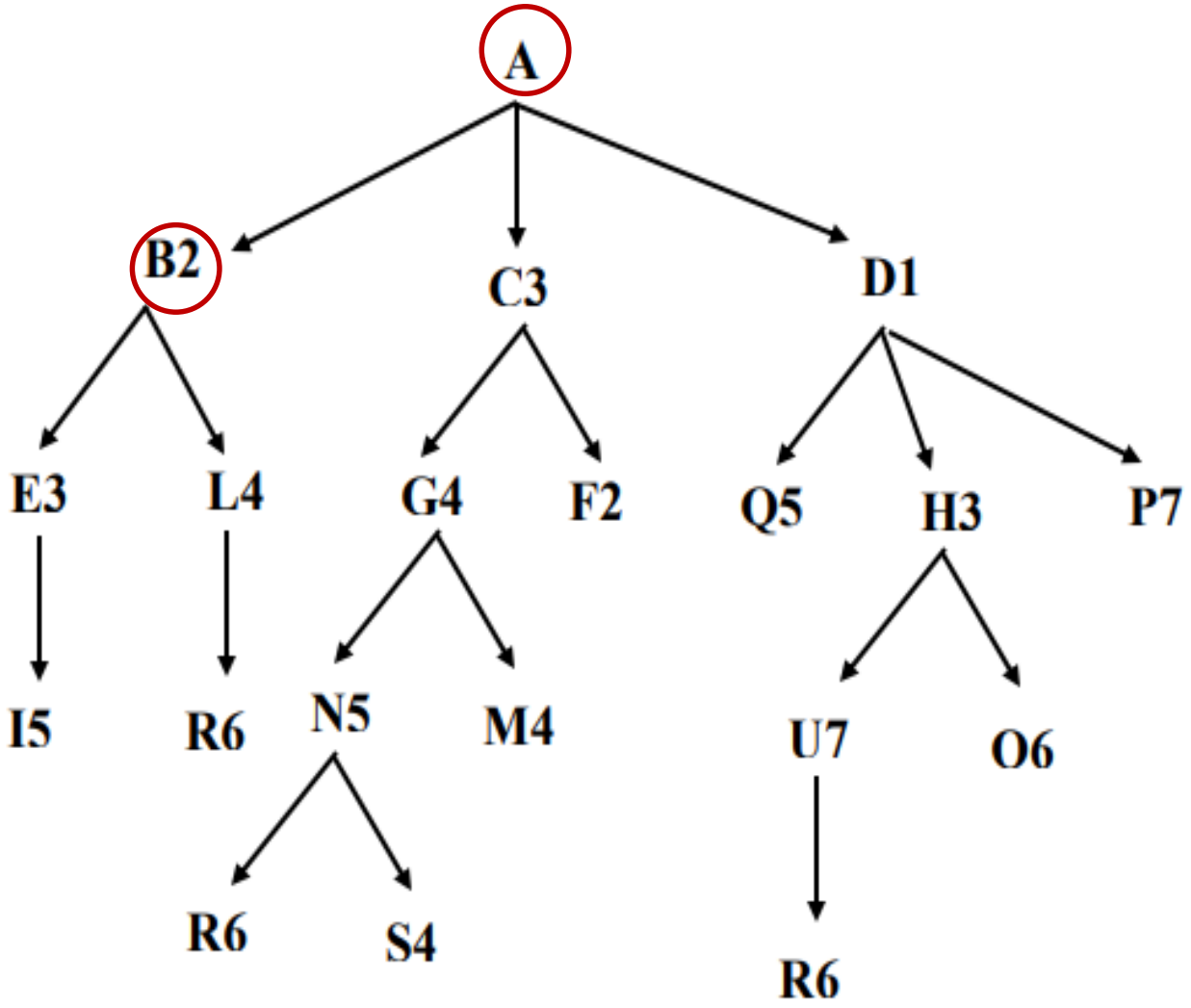
Nút được xét	Nút lân cận được chọn
(0,A)	

Simple hill Climbing (Leo đồi đơn giản)



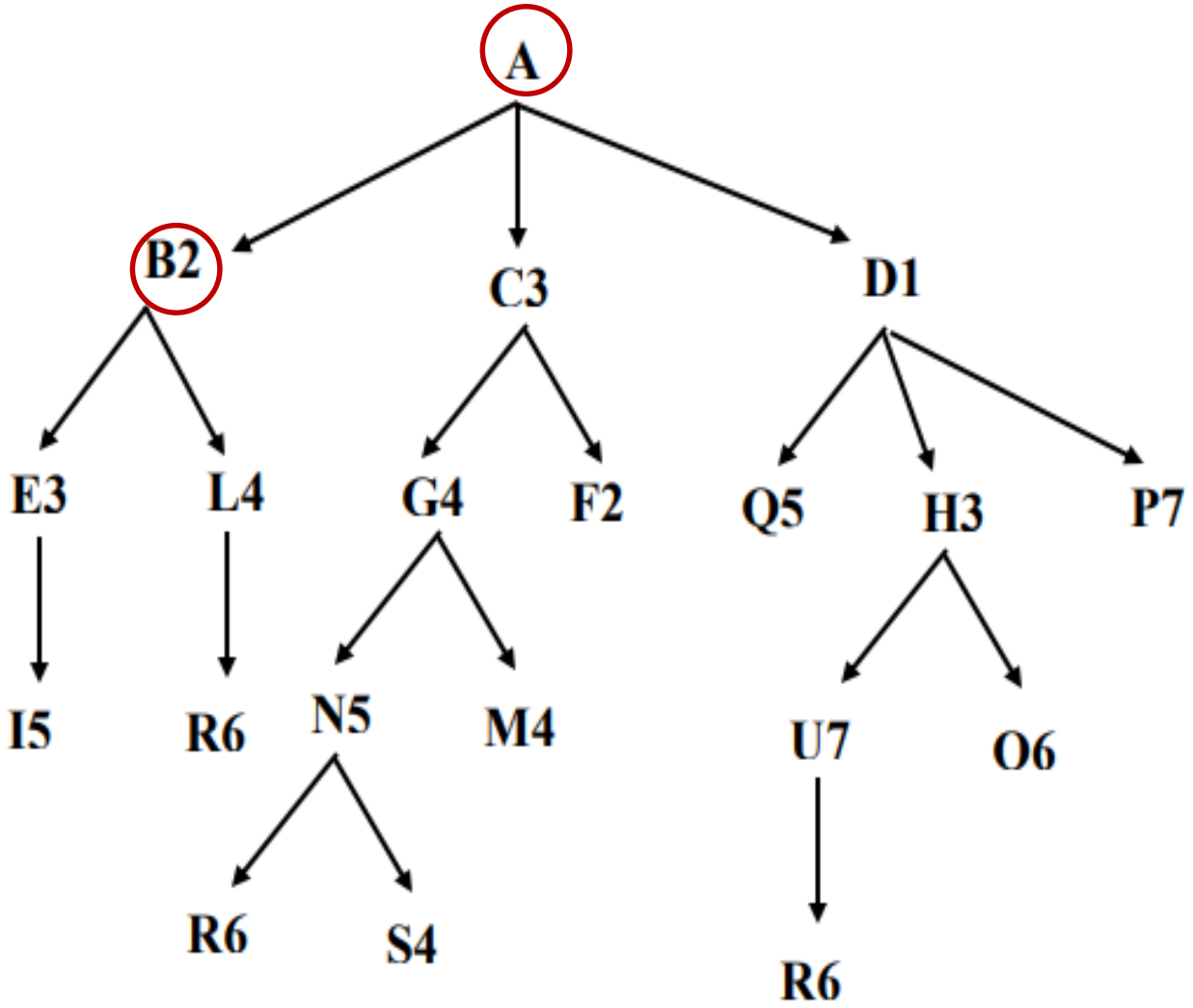
Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)

Simple hill Climbing (Leo đồi đơn giản)



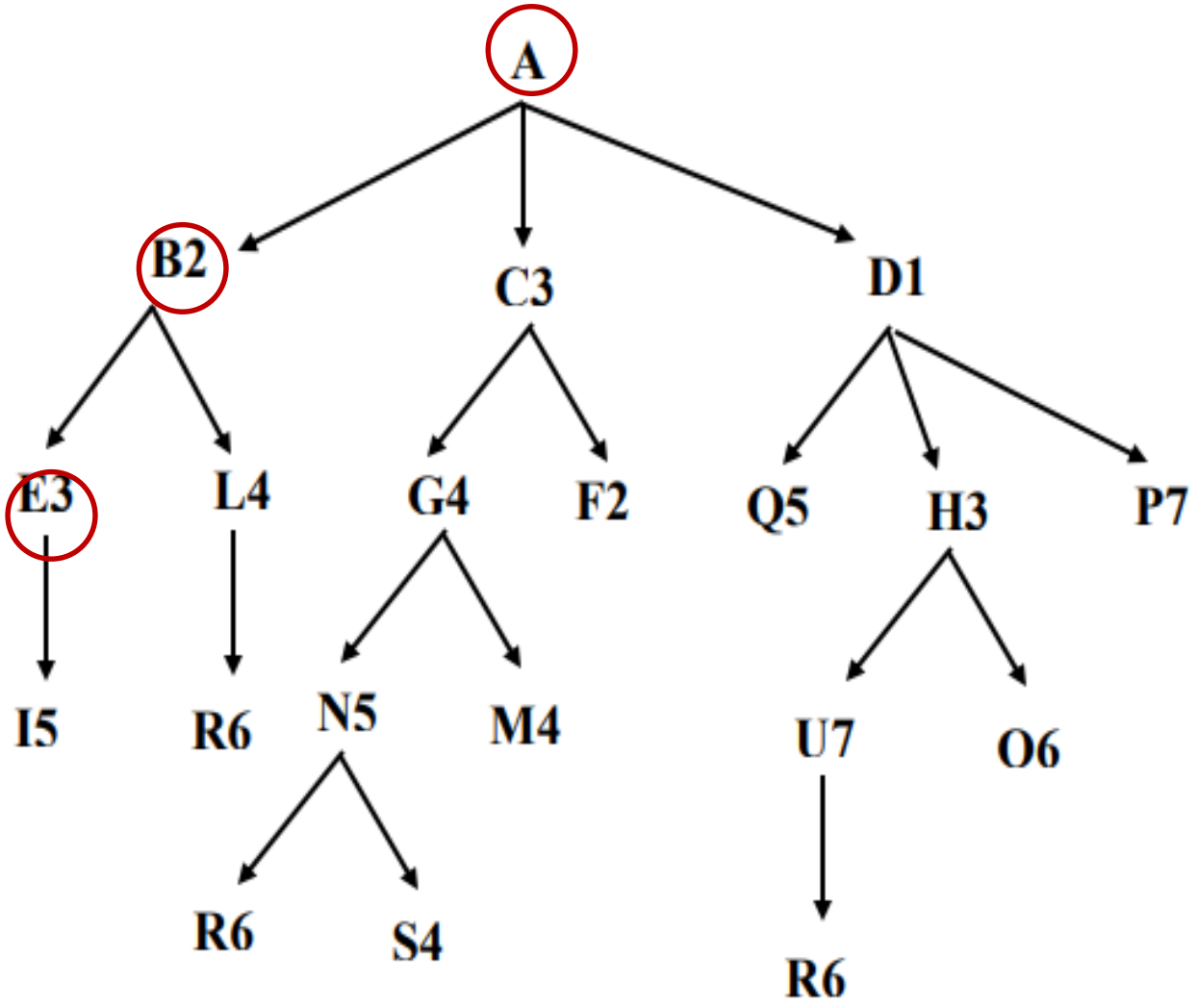
Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)
(2,BA)	

Simple hill Climbing (Leo đồi đơn giản)



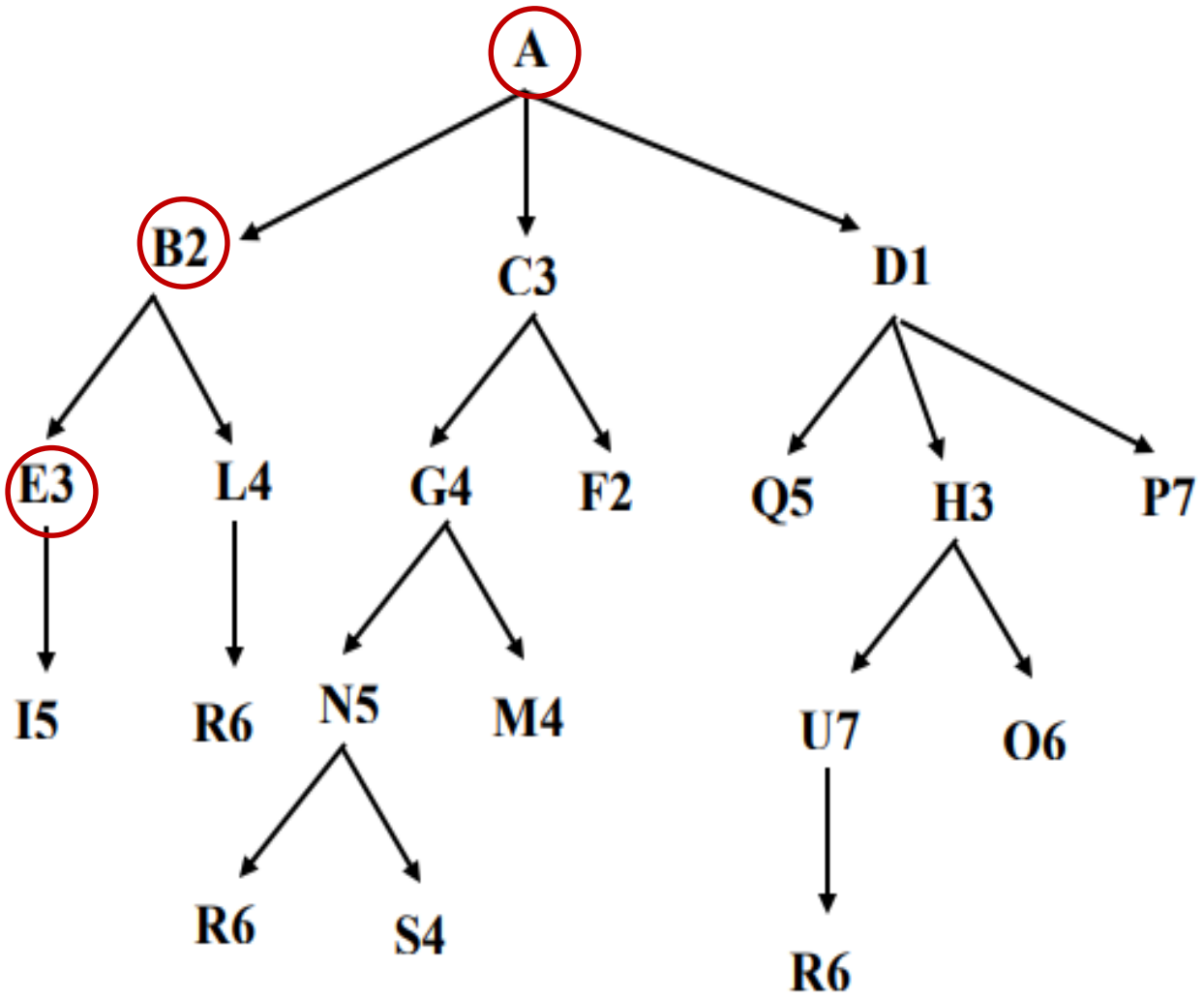
Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)
(2,BA)	(3,EBA)

Simple hill Climbing (Leo đồi đơn giản)



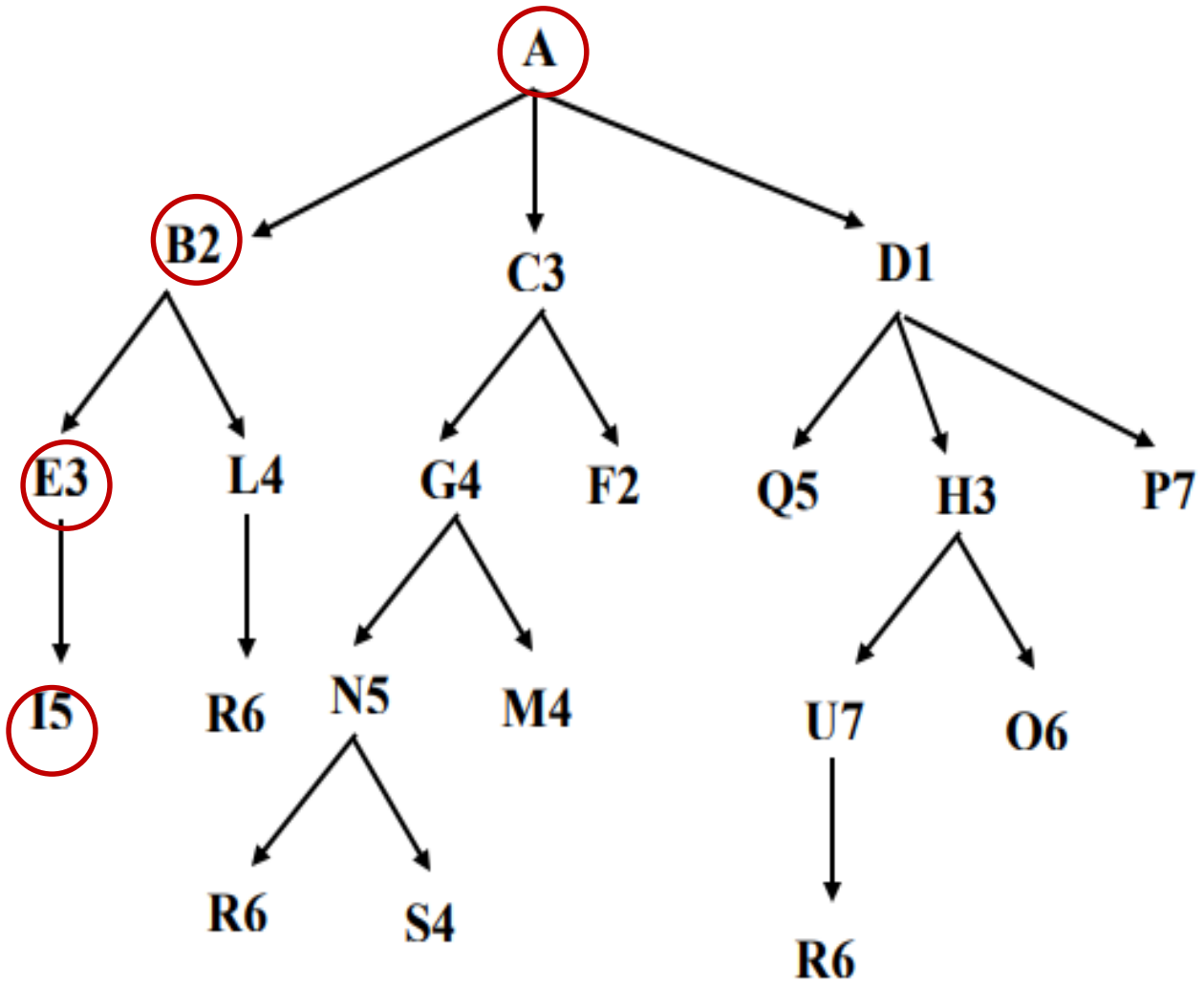
Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)
(2,BA)	(3,EBA)
(3,EBA)	

Simple hill Climbing (Leo đồi đơn giản)



Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)
(2,BA)	(3,EBA)
(3,EBA)	(5, IEBA)

Simple hill Climbing (Leo đồi đơn giản)



Nút được xét	Nút lân cận được chọn
(0,A)	(2,BA)
(2,BA)	(3,EBA)
(3,EBA)	(5, IEBA)
(5, IEBA)	DỪNG

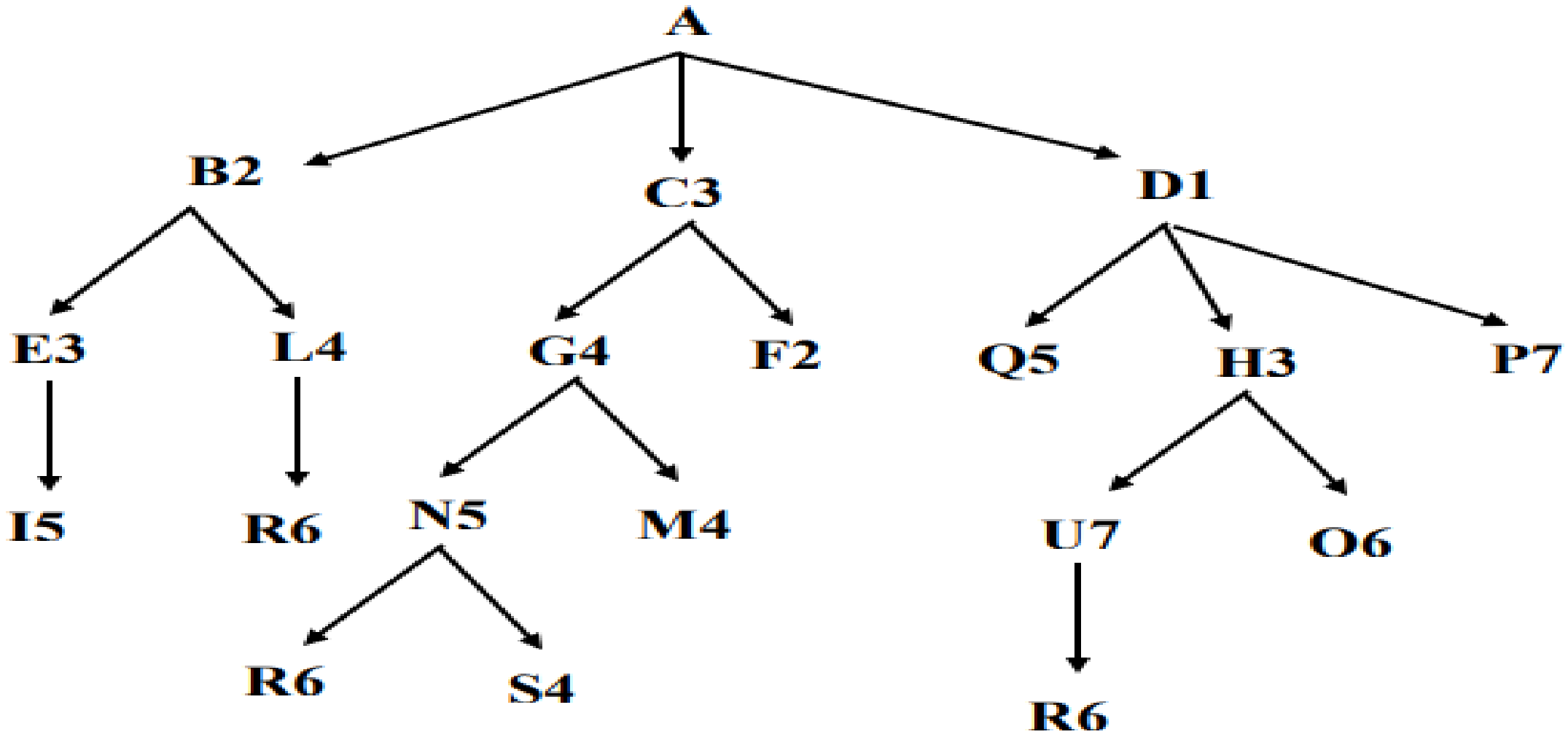
Steepest-Ascent hill climbing (Leo đồi dốc nhất)

- Là một biến thể của thuật toán leo đồi đơn giản.
- Thuật toán này kiểm tra tất cả các nút lân cận của trạng thái hiện tại và chọn một nút lân cận gần nhất với trạng thái mục tiêu.

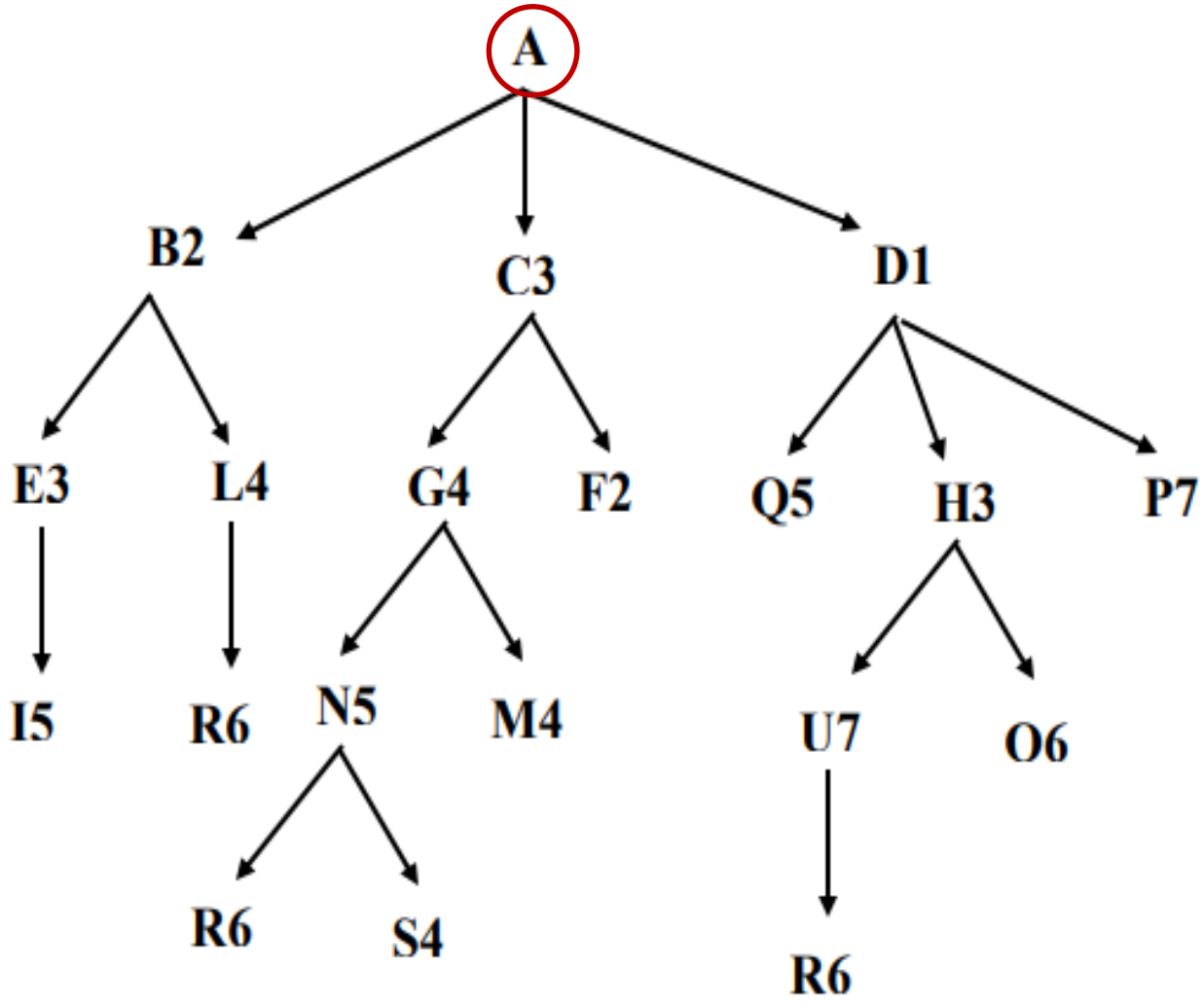
Steepest-Ascent hill climbing (Leo đồi dốc nhất)

- BƯỚC 1: Chọn trạng thái hiện tại
- BƯỚC 2: Tạo tất cả các trạng thái lân cận của trạng thái hiện tại
- BƯỚC 3: Đánh giá hàm mục tiêu tại tất cả các lân cận
- BƯỚC 4: Nếu hàm mục tiêu tại trạng thái hiện tại có giá trị cao hơn tất cả các trạng thái lân cận, thì trạng thái hiện tại đã là giá trị tối đa: KẾT THÚC TÌM KIẾM và nhảy đến BƯỚC 6. Nếu không, lân cận có cải thiện cao nhất của hàm mục tiêu sẽ trở thành trạng thái hiện tại
- BƯỚC 5: Lặp lại BƯỚC 2-BƯỚC 4 trong n lần lặp
- BƯỚC 6: Trả về trạng thái hiện tại và giá trị hàm mục tiêu của nó

Steepest-Ascent hill climbing (Leo đồi dốc nhất)

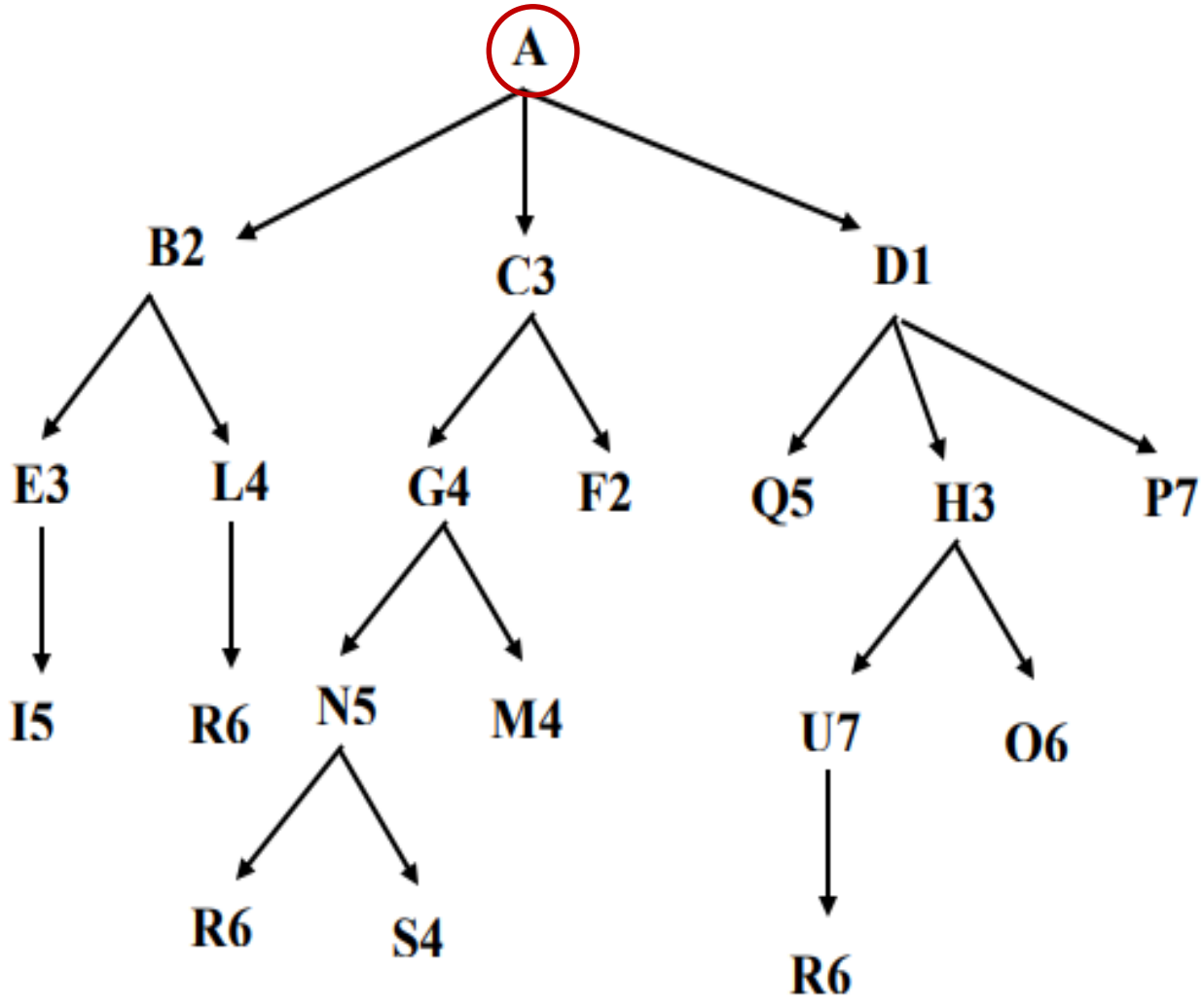


Steepest-Ascent hill climbing (Leo đồi dốc nhất)



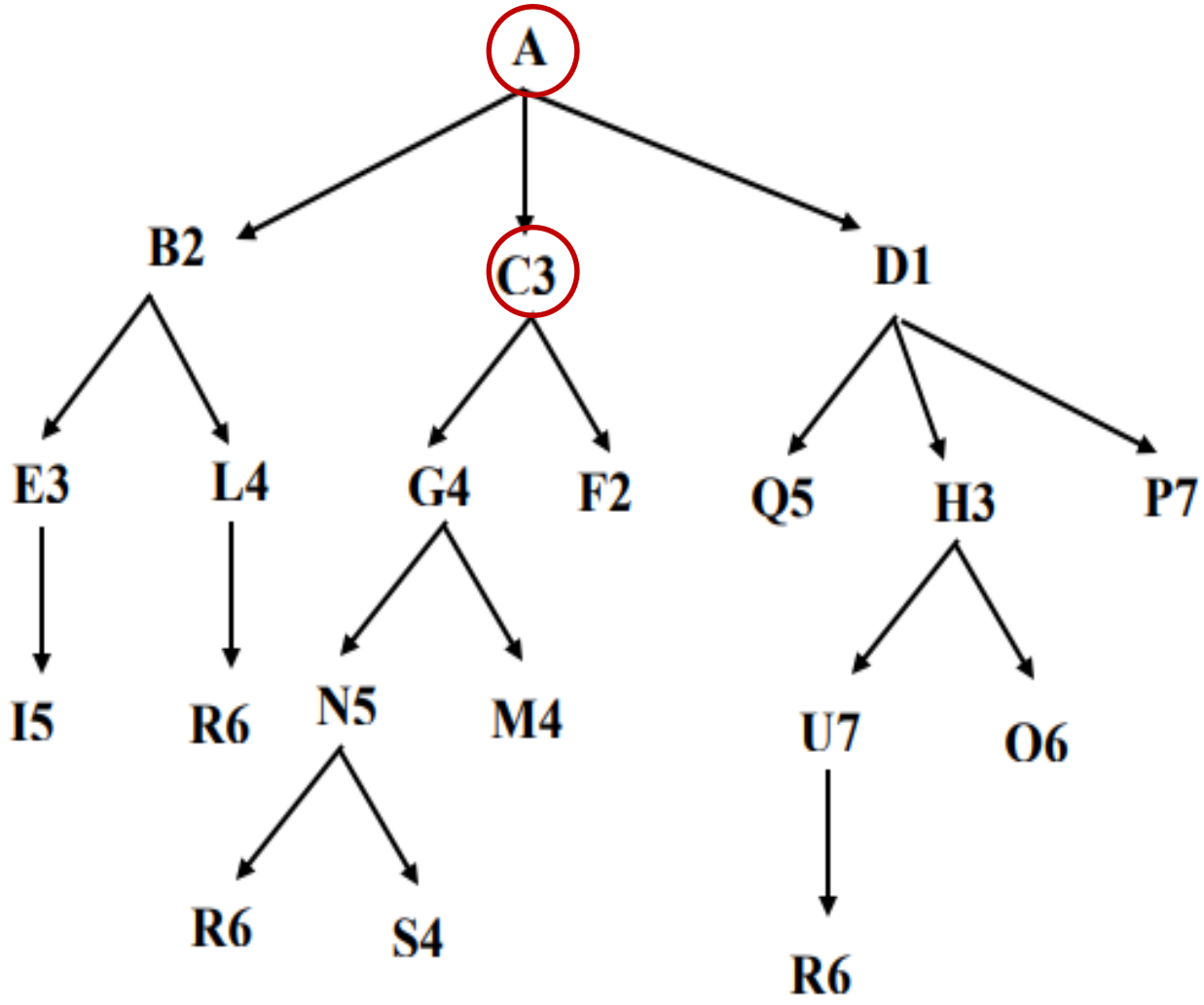
Nút được xét	Tập N
(0,A)	

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



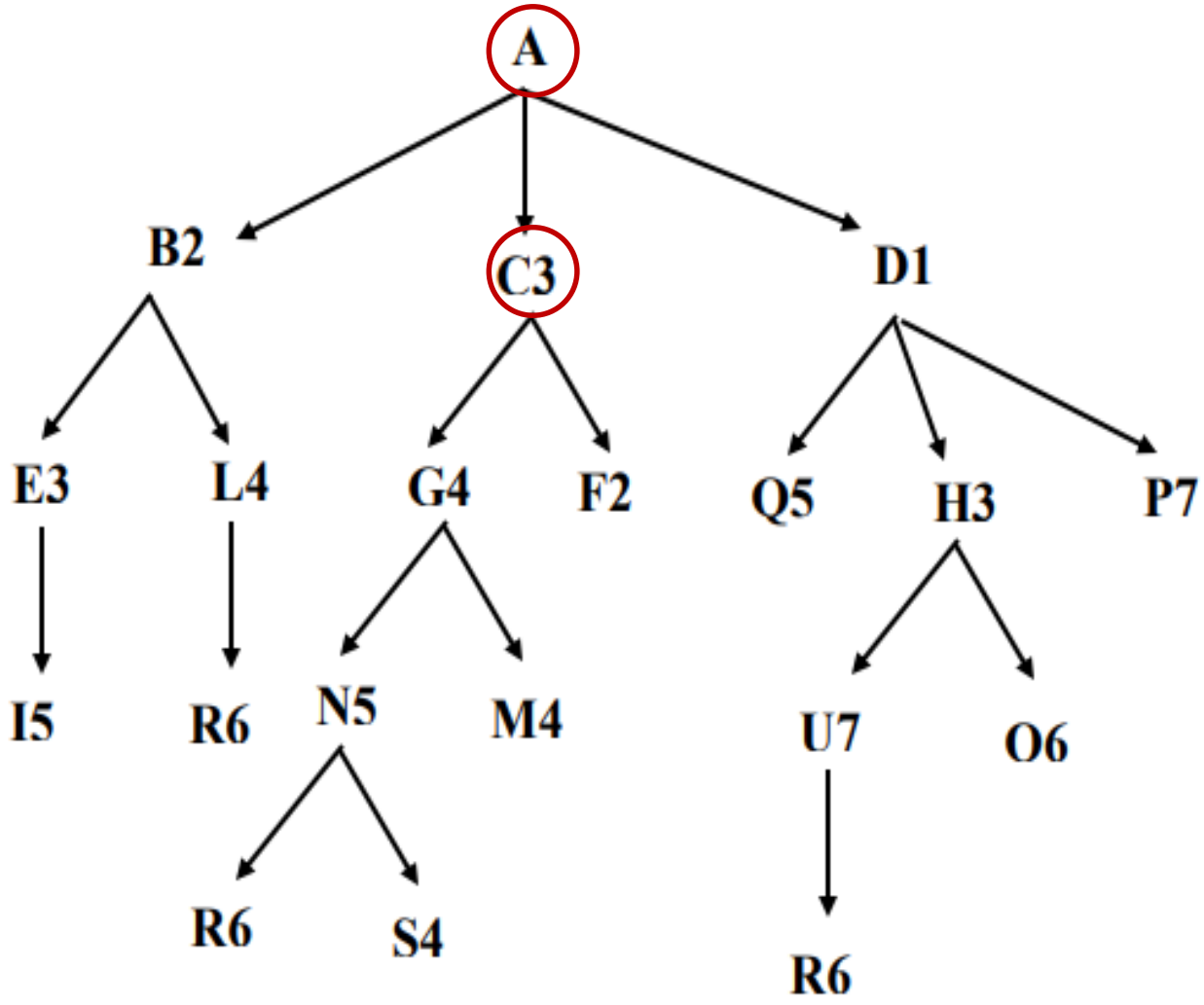
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



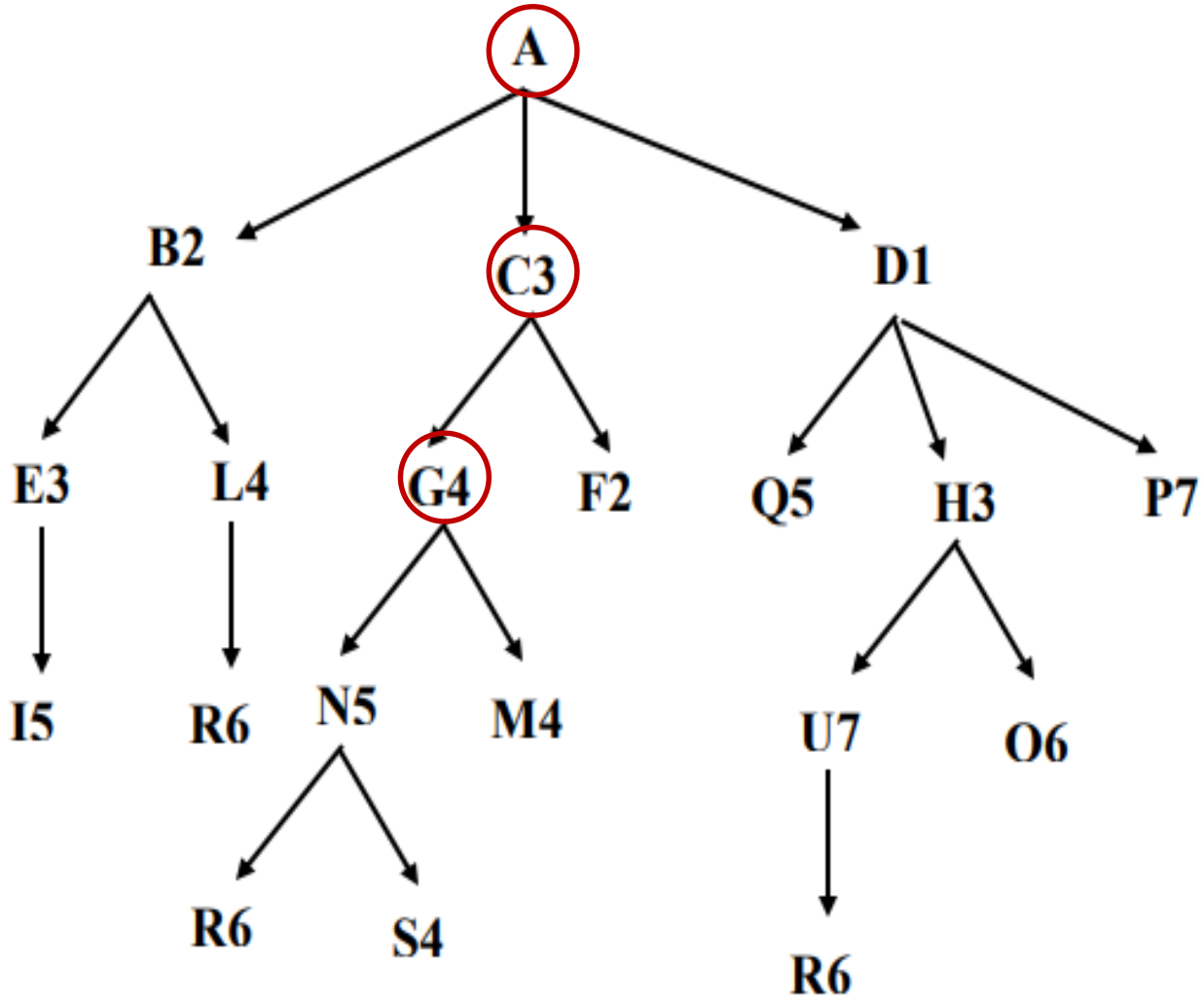
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



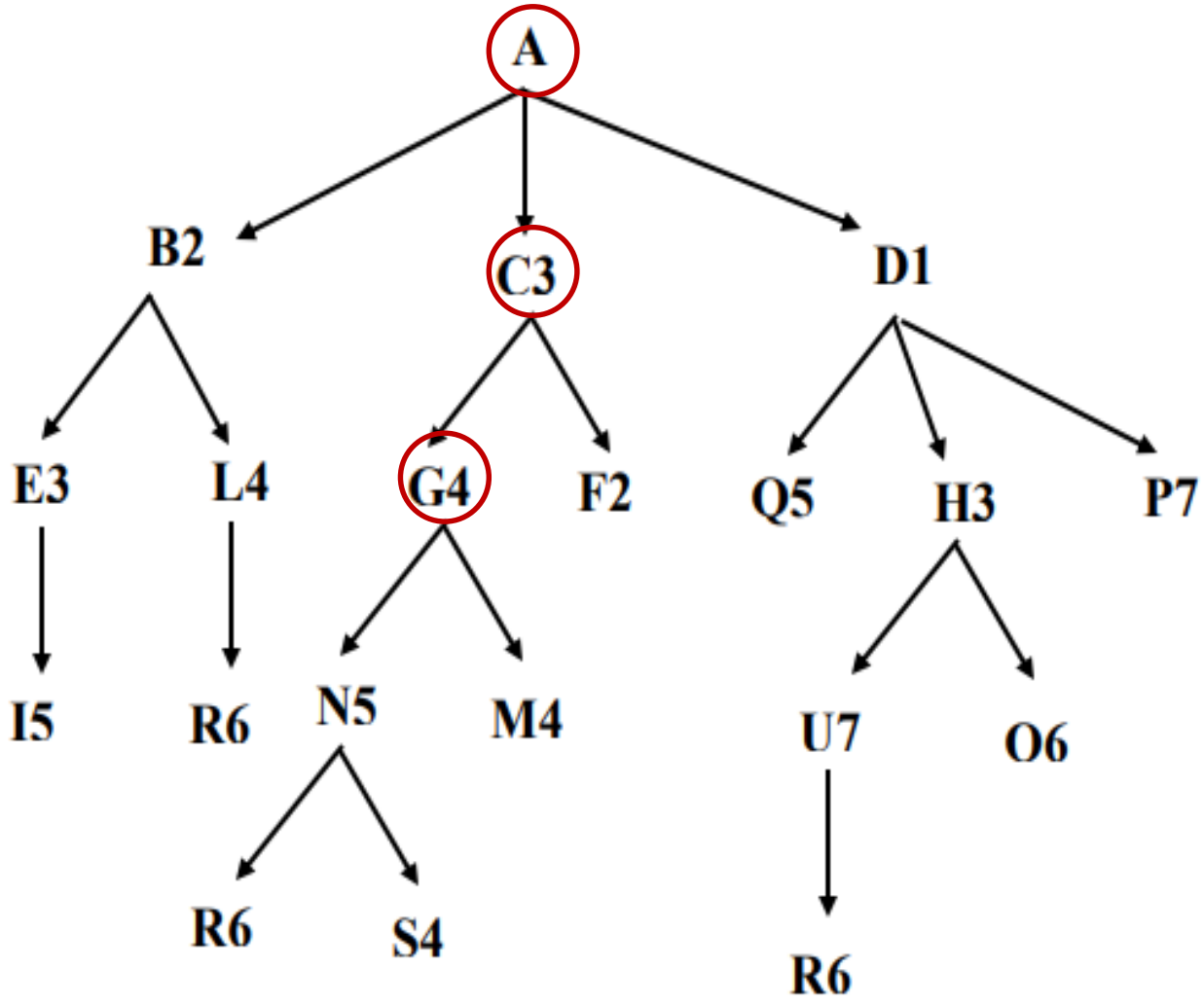
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



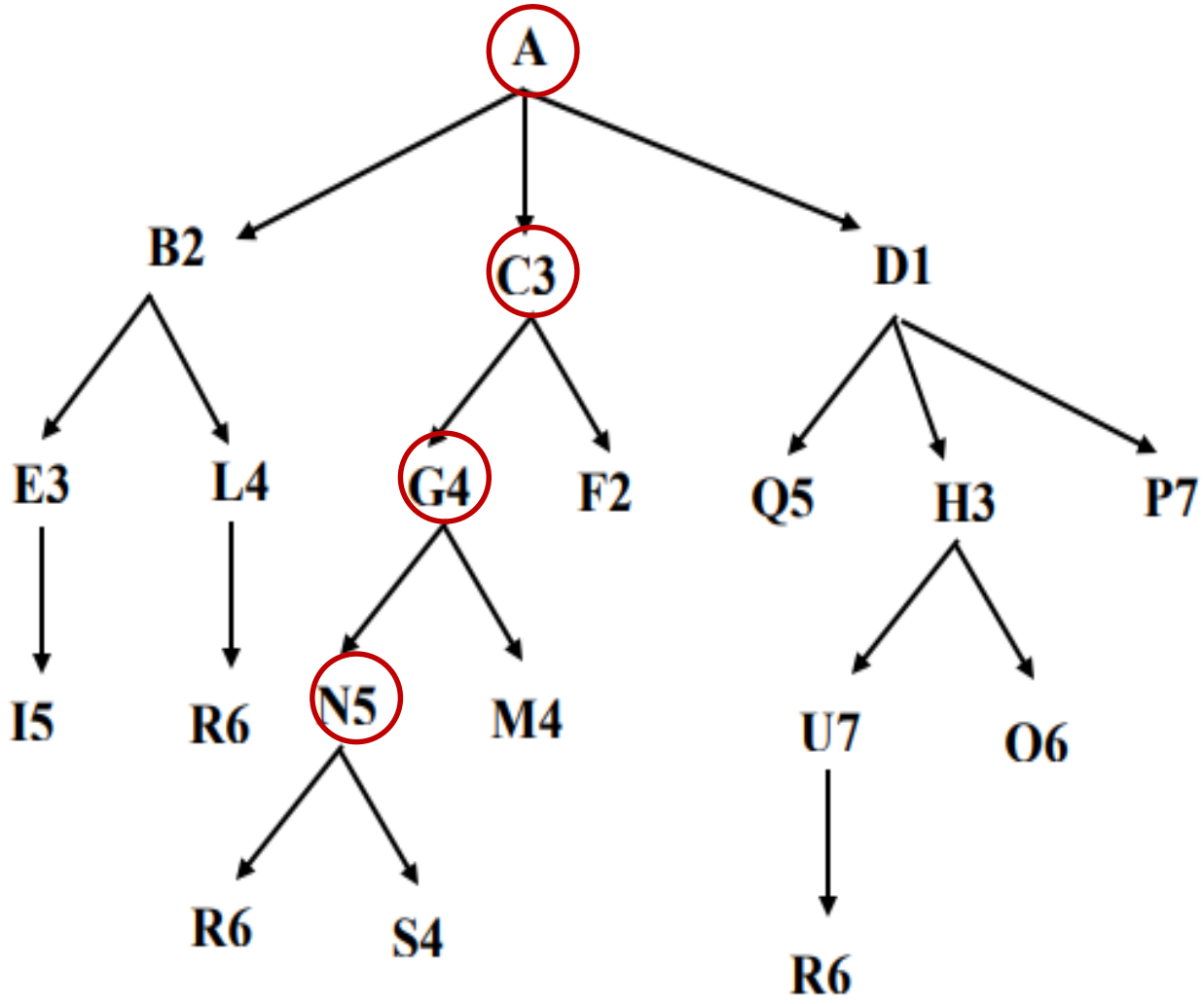
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



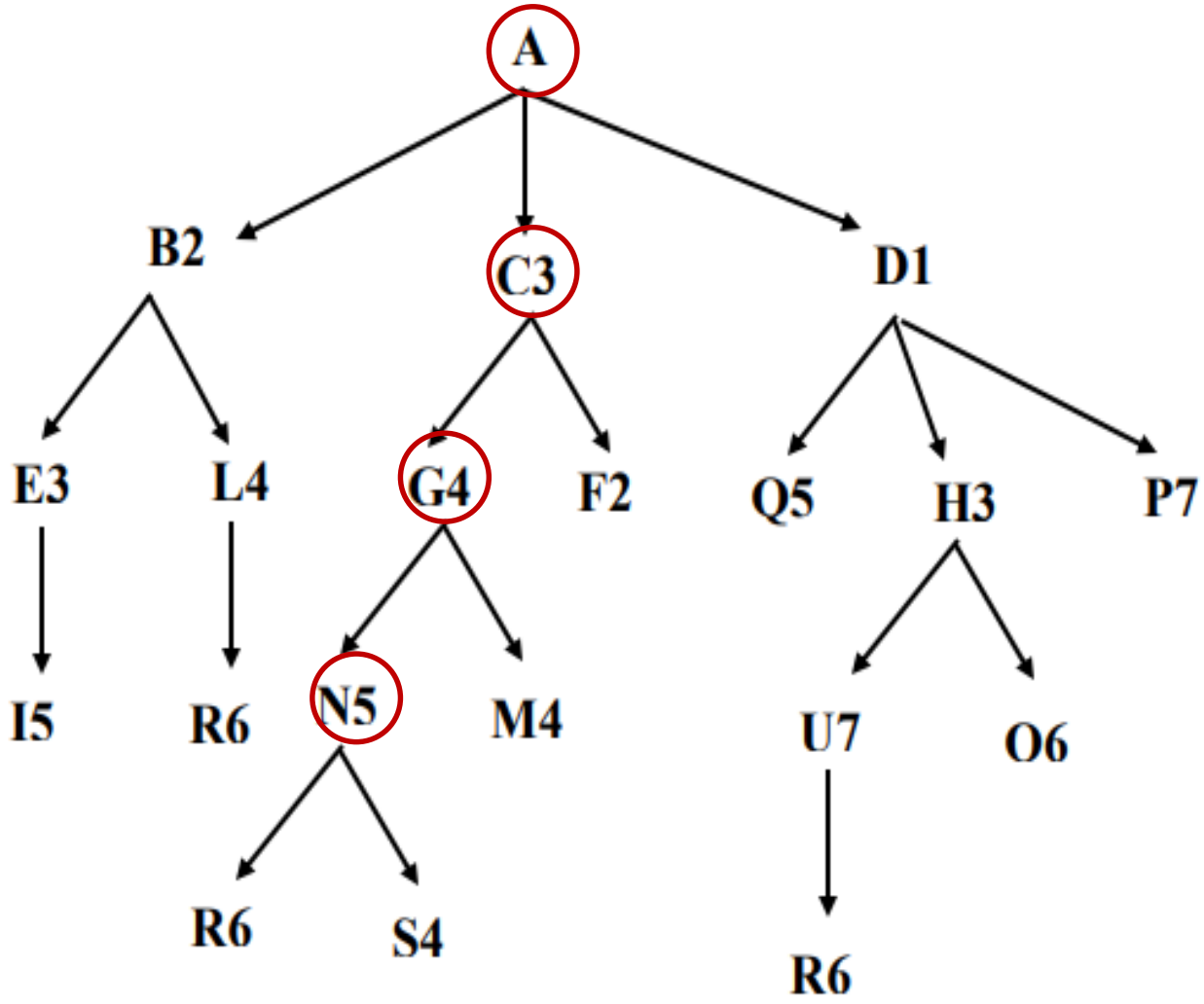
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	(5, NGCA) (4, MGCA)

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



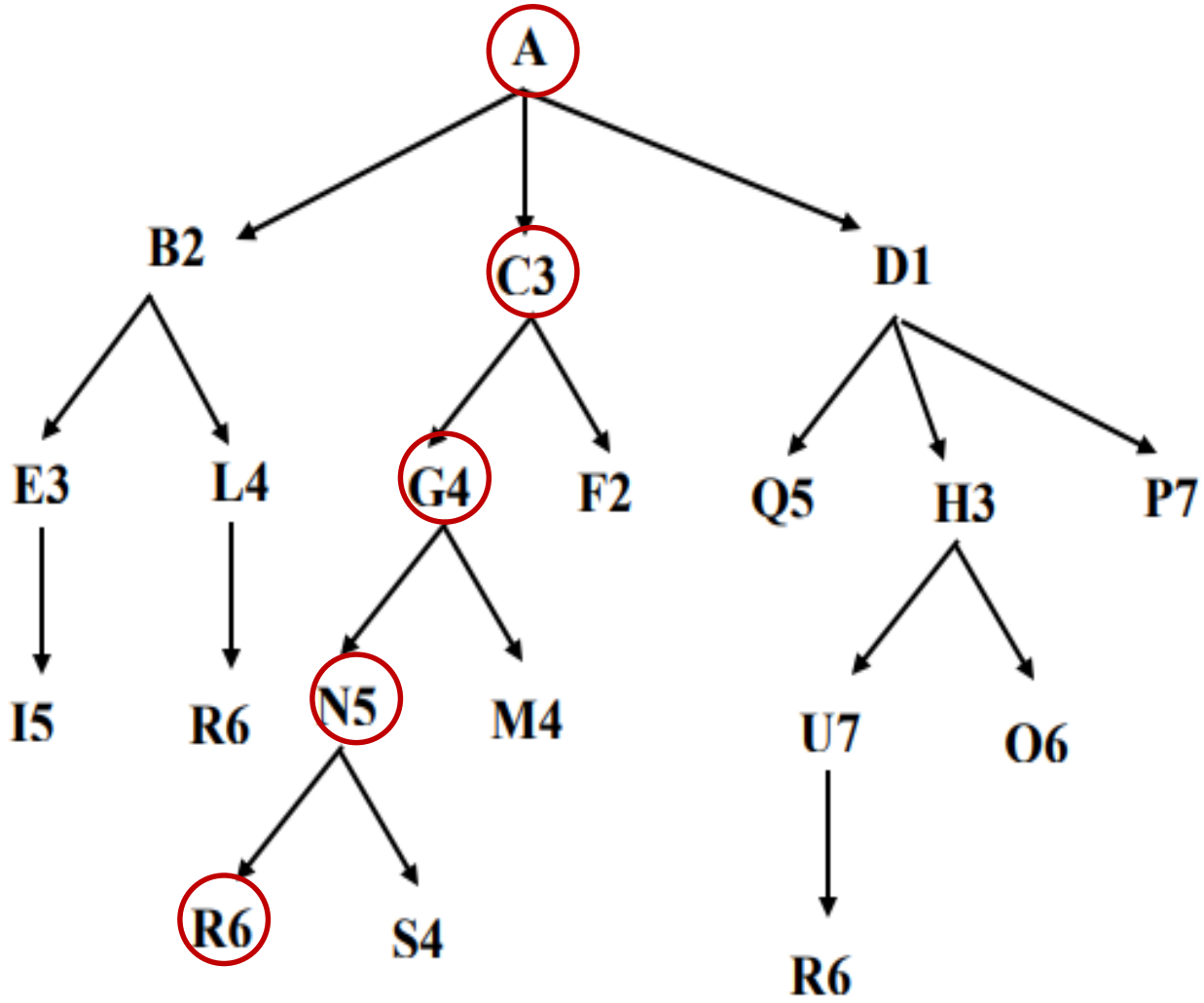
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	(5, NGCA) (4, MGCA)
(5, NGCA)	

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



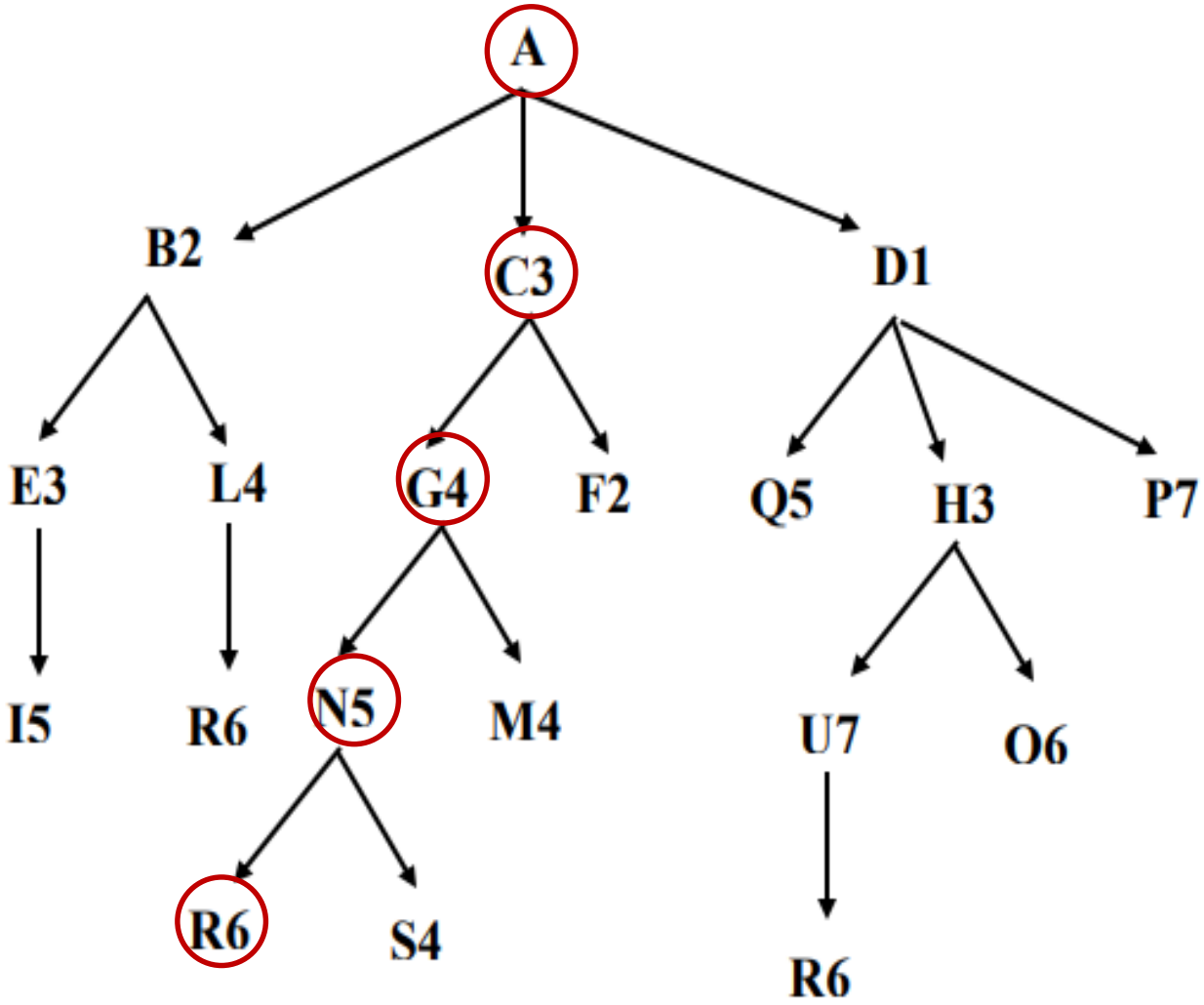
Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	(5, NGCA) (4, MGCA)
(5, NGCA)	(6, RNGCA) (4, SNGCA)

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	(5, NGCA) (4, MGCA)
(5, NGCA)	(6, RNGCA) (4, SNGCA)
(6, RNGCA)	

Steepest-Ascent hill climbing (Leo đồi dốc nhất)



Nút được xét	Tập N
(0,A)	(2,BA) (3,CA) (1,DA)
(3,CA)	(4, GCA) (2, FCA)
(4, GCA)	(5, NGCA) (4, MGCA)
(5, NGCA)	(6, RNGCA) (4, SNGCA)
(6, RNGCA)	ĐÍCH

Stochastic hill Climbing (Leo đồi ngẫu nhiên)

- Là một biến thể của thuật toán leo đồi đơn giản.
- Thay vì tìm ra hàng xóm tốt nhất, phiên bản này lựa chọn ngẫu nhiên một hàng xóm. Nếu hàng xóm đó tốt hơn trạng thái hiện thời, hàng xóm đó sẽ được chọn làm trạng thái hiện thời và thuật toán lặp lại. Ngược lại, nếu hàng xóm được chọn không tốt hơn, thuật toán sẽ chọn ngẫu nhiên một hàng xóm khác và so sánh. Thuật toán kết thúc và trả lại trạng thái hiện thời khi đã hết "kiên nhẫn".

Stochastic hill Climbing (Leo đồi ngẫu nhiên)

- BƯỚC 1: Chọn trạng thái hiện tại
- BƯỚC 2: Tạo tất cả các trạng thái lân cận của trạng thái hiện tại
- BƯỚC 3: Đánh giá hàm mục tiêu tại tất cả các lân cận
- BƯỚC 4: Nếu hàm mục tiêu tại trạng thái hiện tại có giá trị cao hơn tất cả các trạng thái lân cận, thì trạng thái hiện tại đã là giá trị tối đa: KẾT THÚC TÌM KIẾM và nhảy đến BƯỚC 6. Nếu không, lân cận có cải thiện cao nhất của hàm mục tiêu sẽ trở thành trạng thái hiện tại
- BƯỚC 5: Lặp lại BƯỚC 2-BƯỚC 4 trong n lần lặp
- BƯỚC 6: Trả về trạng thái hiện tại và giá trị hàm mục tiêu của nó

Ví dụ

- N-queen: f(s) = số lượng quân hậu xung đột trong trạng thái s
  - Lưu ý chúng ta muốn s có điểm thấp nhất f(s)=0
- Bài toán người du lịch (TSP)
  - Đến mỗi thành phố một lần, quay lại thành phố đầu tiên
  - Trạng thái = thứ tự các thành phố, f(s) = tổng số dặm