ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Cơ sở trí tuệ nhân tạo

Bài toán thỏa mãn ràng buộc

Nguyễn Ngọc Đức

2021

Nội dung

Bài toán thỏa mãn ràng buộc

Bài toán thỏa mãn ràng buộc I

- Trong bài toán tìm kiếm, trạng thái là các "hộp đen"
  - Với mỗi bài toán ta cần có tri thức cụ thể để định nghĩa **mô hình di chuyển**
- Một **bài toán thỏa mãn ràng buộc (CSP)**:
  - Trạng thái: được biểu diễn cụ thể bằng tập các biến
  - Lời giải: các giá trị biến thỏa mãn tất cả ràng buộc
- Mô hình di chuyển có thể được suy diễn bằng định nghĩa bài toán
- Lời giải có tính tổng quát cao cho các bài toán phức tạp

Bài toán thỏa mãn ràng buộc II

Định nghĩa

Một bài toán thỏa mãn ràng buộc bao gồm 3 thành phần $\mathcal{X}, \mathcal{D}$ và $\mathcal{C}$.

- Tập các biến, $\mathcal{X} = \{X_1, ..., X_n\}$
- Tập miền giá trị $\mathcal{D} = \{D_1, ...D_n\}$
  - $D_i = \{v_1, ..., v_k\}$: miền giá trị của các biến X_i
- $\mathcal{C}$ tập ràng buộc chỉ định tổ hợp các giá trị cho phép.

Không gian trạng thái và lời giải I

- Mỗi trạng thái trong CSP được định nghĩa bằng một phép gán các giá trị biến

$$\{X_i = v_i, X_j = v_j, ...\}$$

- Một **phép gán** không vi phạm ràng buộc nào được gọi là một phép gán **nhất quán** hoặc **hợp lệ**
- Một phép gán **đầy đủ** gán giá trị cho tất cả các biến
- Lời giải của một bài toán CSP là một phép gán **đầy đủ** và **hợp lệ**

Không gian trạng thái và lời giải II

Các vấn đề giải quyết với một bài toán CSP:

- Xác định **tồn tại** lời giải hay không?
- **Tìm** lời giải
- **Tìm tất cả** lời giải
- **Đếm** số lượng lời giải
- Tìm lời giải **tốt nhất** $\Rightarrow$ bài toán tối ưu hóa
- Xác định **tính chất** của lời giải

Các loại ràng buộc

- Ràng buộc đơn (unary constraint): ràng buộc 1 biến
- Ràng buộc kép (binary constraint): ràng buộc 2 biến
- Ràng buộc bậc cao (high-order constraint): ràng buộc nhiều biến
- Ràng buộc mềm (soft constraint): ràng buộc có khả năng vi phạm

Đồ thị ràng buộc

- Một bài toán CSP có thể biểu diễn dưới dạng đồ thị

Đồ thị ràng buộc là một đồ thị

- Nút là các biến
- Cung là các ràng buộc

Mô hình hóa bài toán CSP

- Cho chúng ta một thể hiện **tự nhiên** cho nhiều loại bài toán
- Rất nhiều bài toán không có khả năng lưu vết trong không gian trạng thái có thể được giải dễ dàng bằng cách mô hình hóa CSP
- Giúp chúng ta hiểu rõ hơn về bài toán cũng như lời giải của nó

Ví dụ: Bài toán tô màu I

Tô màu cho các vùng lãnh thổ úc sao cho không vùng lân cận nào có màu giống nhau

Ví dụ: Bài toán tô màu II

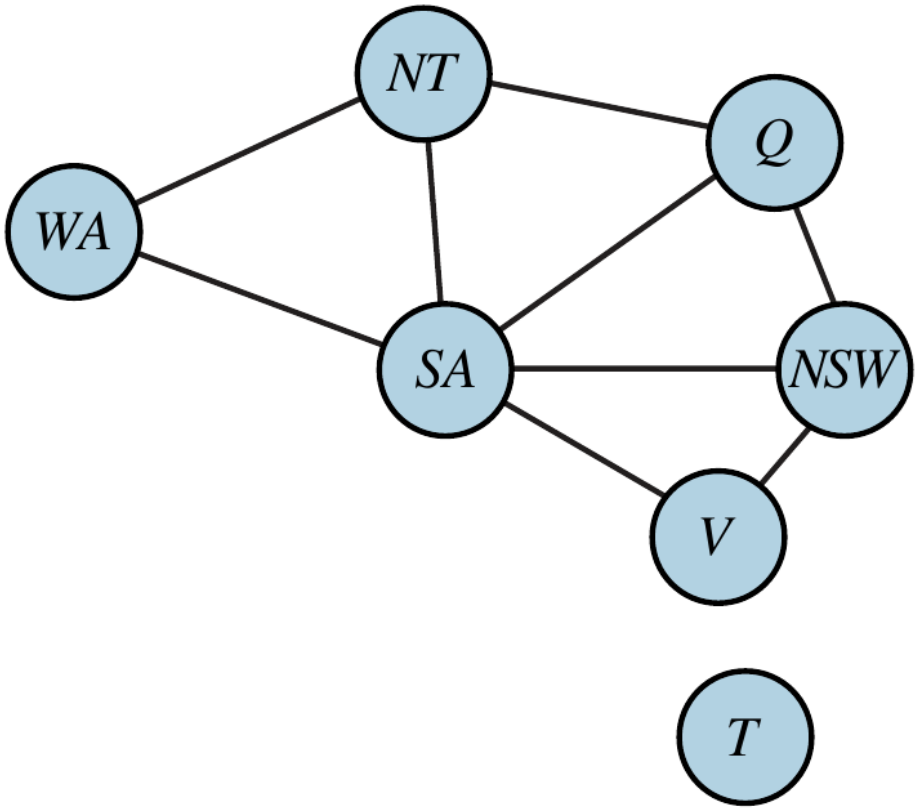
Ta có thể mô hình hóa CSP cho bài toán trên:

- **Biến:** $\mathcal{X} = \{WA, NT, Q, NSW, V, SA, T\}$
- **Miền giá trị:** $D_i = \{red, green, blue\}$
- **Ràng buộc:** các vùng lân cận có màu khác nhau

  $\mathcal{C} = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V, WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

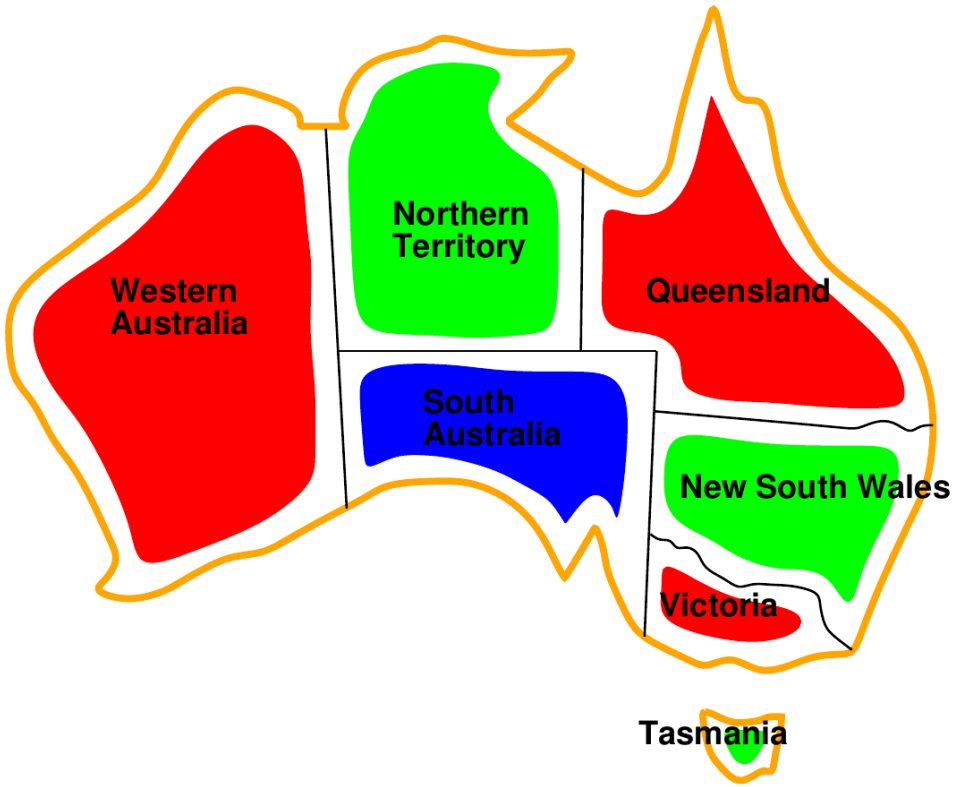
Ví dụ: Bài toán tô màu III

- Đồ thị ràng buộc:



Ví dụ: Bài toán tô màu IV

- Có rất nhiều lời giải, ví dụ: $\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$



Tìm kiếm quay lui

Mô tả CSP dưới dạng bài toán tìm kiếm

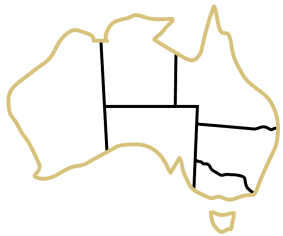
- **Trạng thái bắt đầu:** rỗng
- **Successor:**
  1. Gán giá trị cho từng biến
  2. Chỉ xem xét các giá trị không vi phạm ràng buộc
- **Kiểm tra đích:** phép gán hiện tại là **đầy đủ**
- Với CSP có miền giá trị kích thước d và n biến
  - Lời giải nằm ở nút lá chiều sâu $n \Rightarrow$ DFS
  - $b = (n - l) \cdot d$ ($l \in [1, n]$) tại chiều sâu $l \Rightarrow n! \cdot d^n$ nút lá !!!

$\Rightarrow$ **Tìm kiếm quay lui**

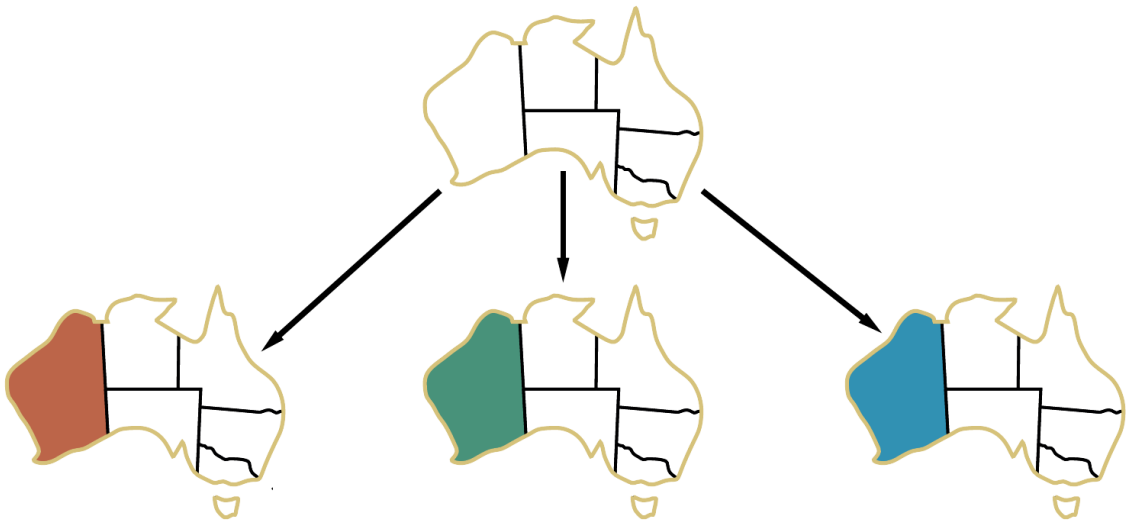
Tìm kiếm quay lui

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
 return BACKTRACK(∅, csp)
function BACKTRACK(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, value)
      if inferences ≠ failure then
        add inferences to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ failure then
          return result
      remove {var = value} and inferences from assignment
  return failure
```

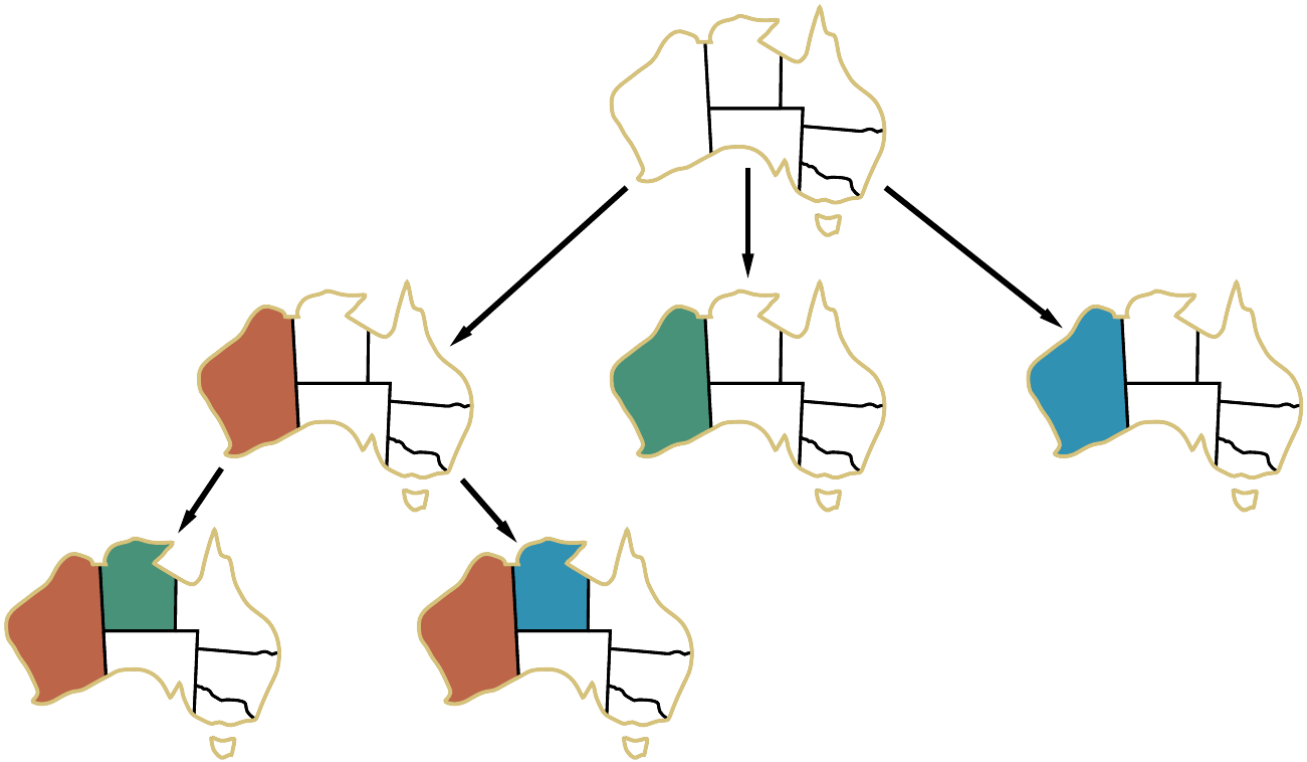
Ví dụ



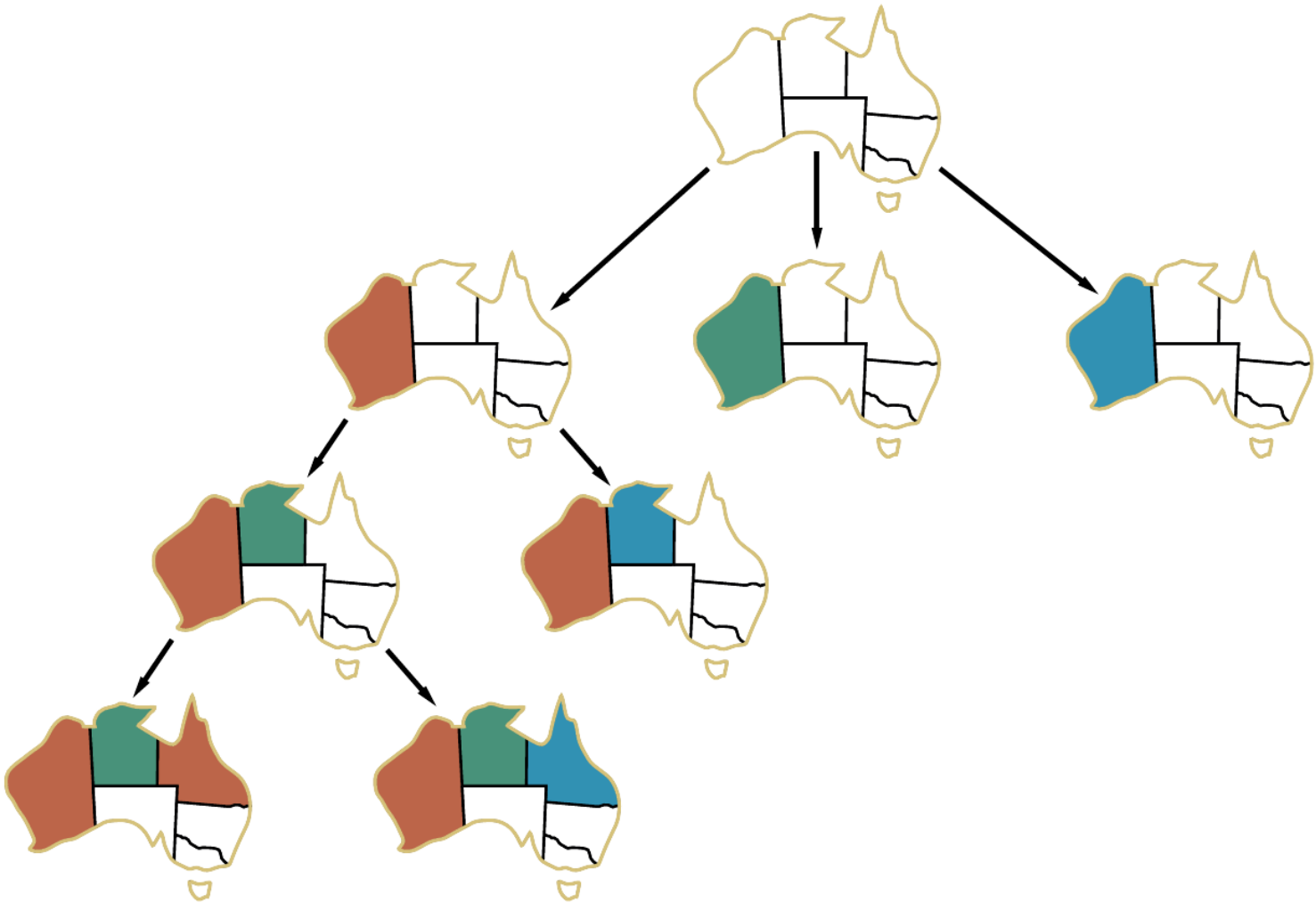
Ví dụ



Ví dụ



Ví dụ



Tăng hiệu suất

1. Thứ tự gán các **biến**?
2. Thứ tự **giá trị** gán thử?
3. Phát hiện trường hợp vô nghiệm sớm?
4. Khai thác kiến trúc bài toán?

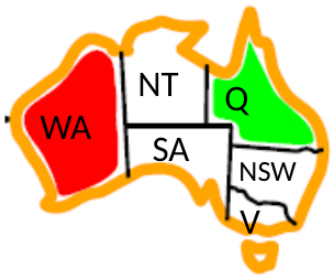
Lan truyền ràng buộc

Lan truyền ràng buộc

- Giảm số lượng giá trị hợp lệ của một biến bằng các ràng buộc $\rightarrow$ giá trị hợp lệ của các biến khác cũng giảm
- Tiền xử lý (presolve) bài toán (vd: sử dụng phương pháp tìm kiếm) $\rightarrow$ đôi khi giải quyết toàn bộ bài toán!!!
- Tăng tính nhất quán cục bộ $\rightarrow$ loại bỏ sớm các giá trị không hợp lệ

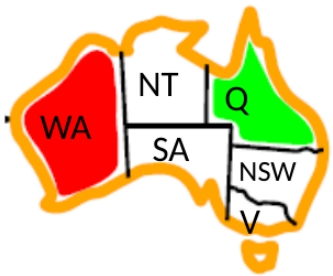
Kiểm tra tiến

- **Ý tưởng:** Kiểm soát các giá trị hợp lệ cho các biến chưa được gán giá trị $\rightarrow$ Ngừng tìm kiếm nếu không còn giá trị hợp lệ
- **Chú ý:** Kiểm tra tiến không phát hiện lỗi sớm
- Chỉ đảm bảo tính **nhất quán cục bộ (local consistency)**



Kiểm tra tiến

- **Ý tưởng:** Kiểm soát các giá trị hợp lệ cho các biến chưa được gán giá trị $\rightarrow$ Ngừng tìm kiếm nếu không còn giá trị hợp lệ
- **Chú ý:** Kiểm tra tiến không phát hiện lỗi sớm
- Chỉ đảm bảo tính **nhất quán cục bộ (local consistency)**

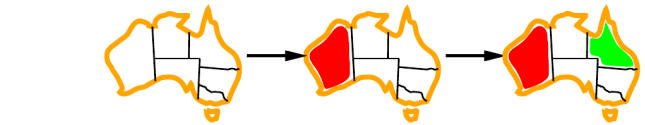


NT và SA không thể cùng là xanh dương!!!

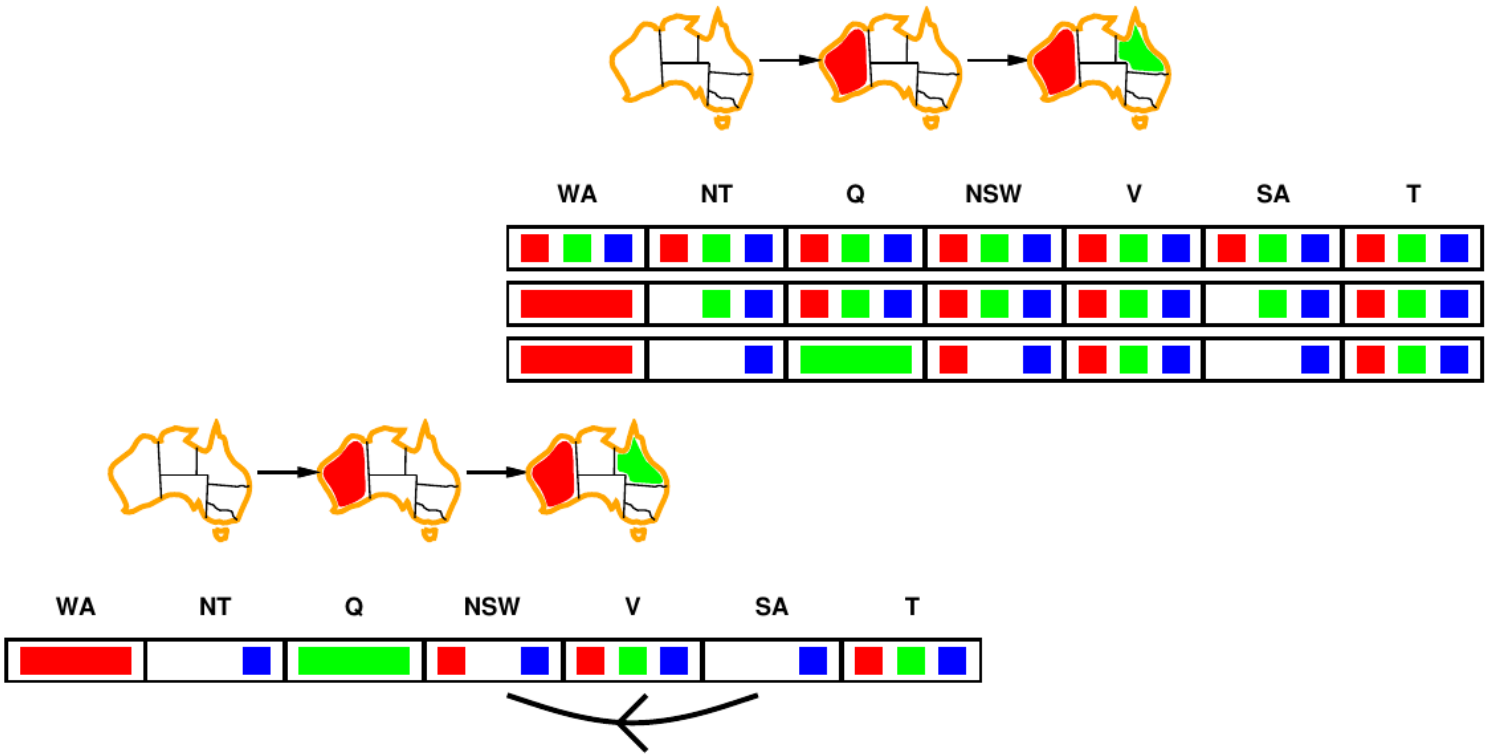
Cạnh hợp lệ

- Một cạnh $X \rightarrow Y$ là **hợp lệ (arc-consistency)** khi $\forall x \in D_X, \exists y \in D_Y$ không vi phạm ràng buộc
- Kiểm tra trước mỗi phép gán
  - Vẫn dựa trên tìm kiếm quay lui!!!
  - **Time:** **Có khả năng** giảm từ $O(n^2d^3)$ xuống còn $O(n^2d^2)$
- **Lưu ý:** Nếu D_X giảm 1 giá trị, lân cận của X cần được kiểm tra lại

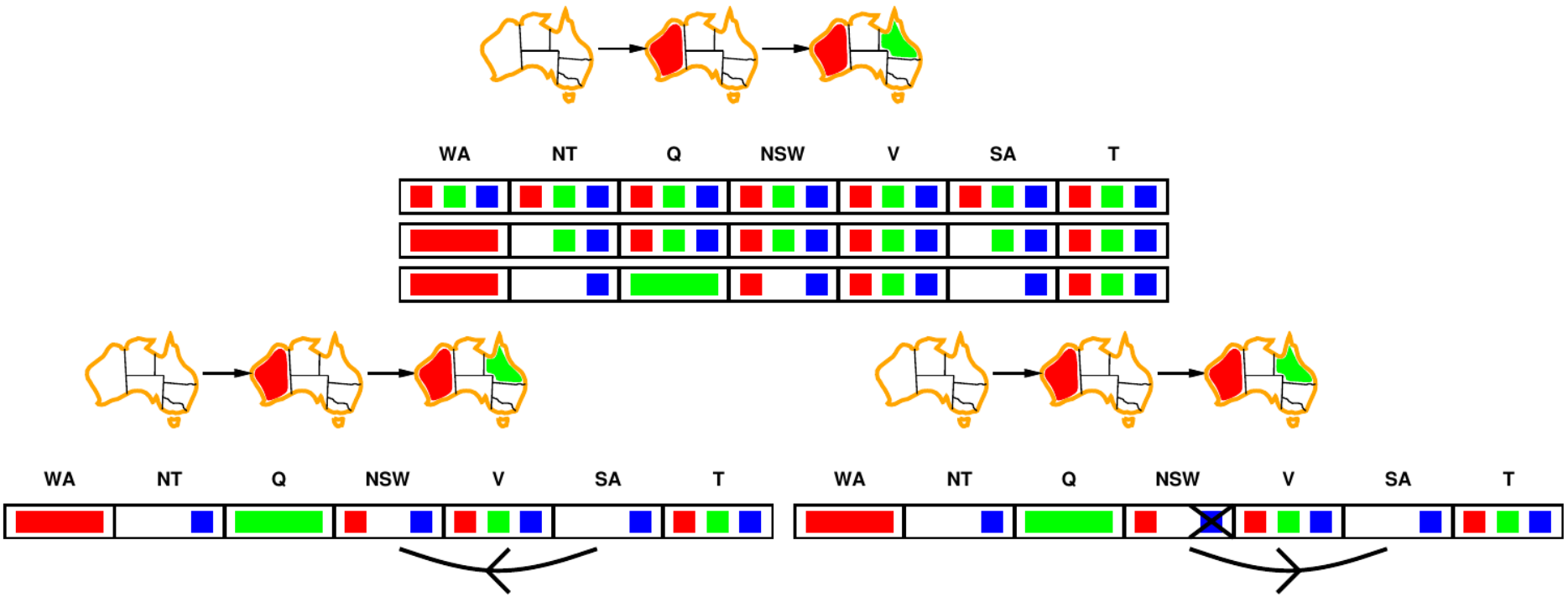
Ví dụ



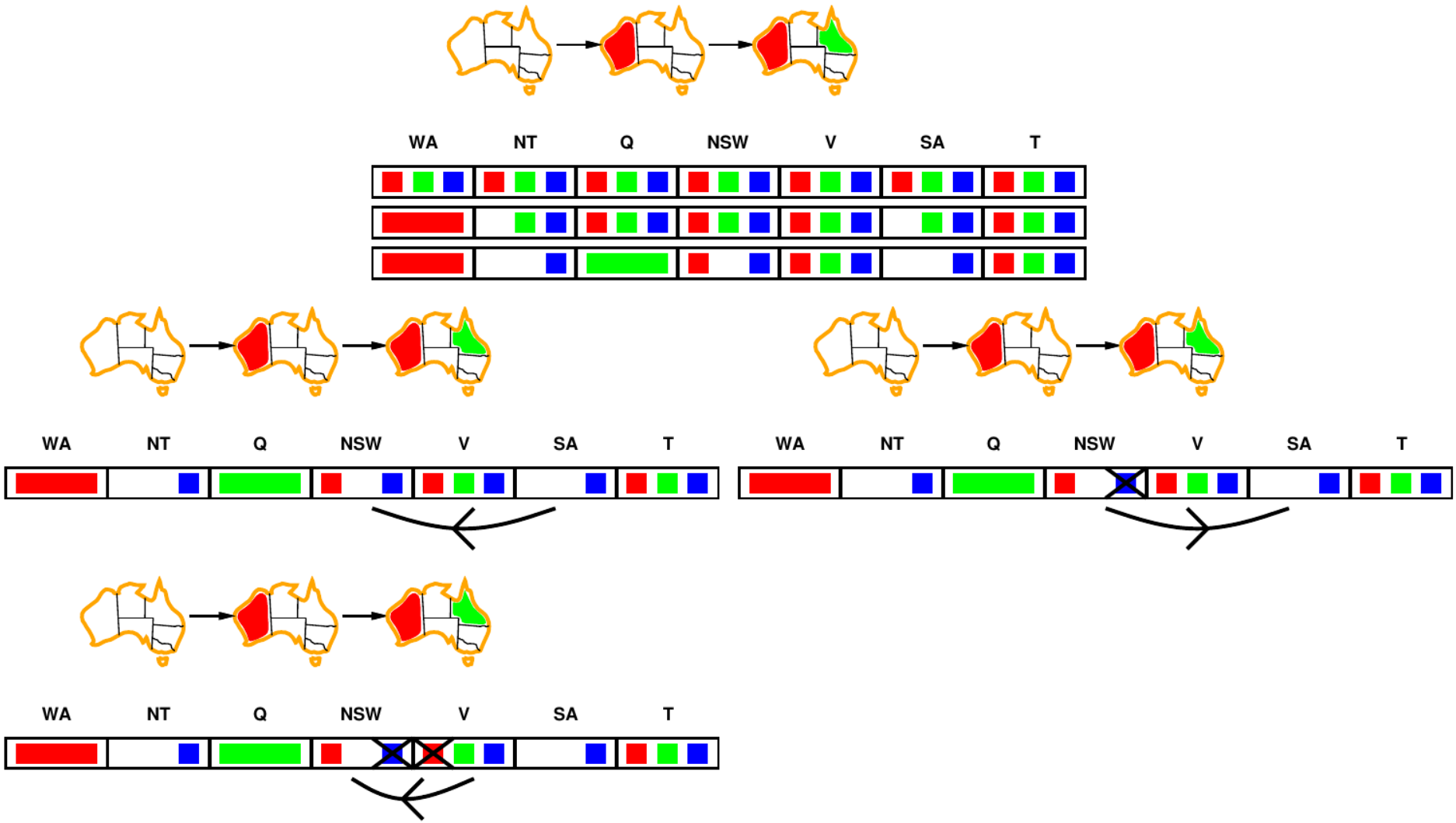
Ví dụ



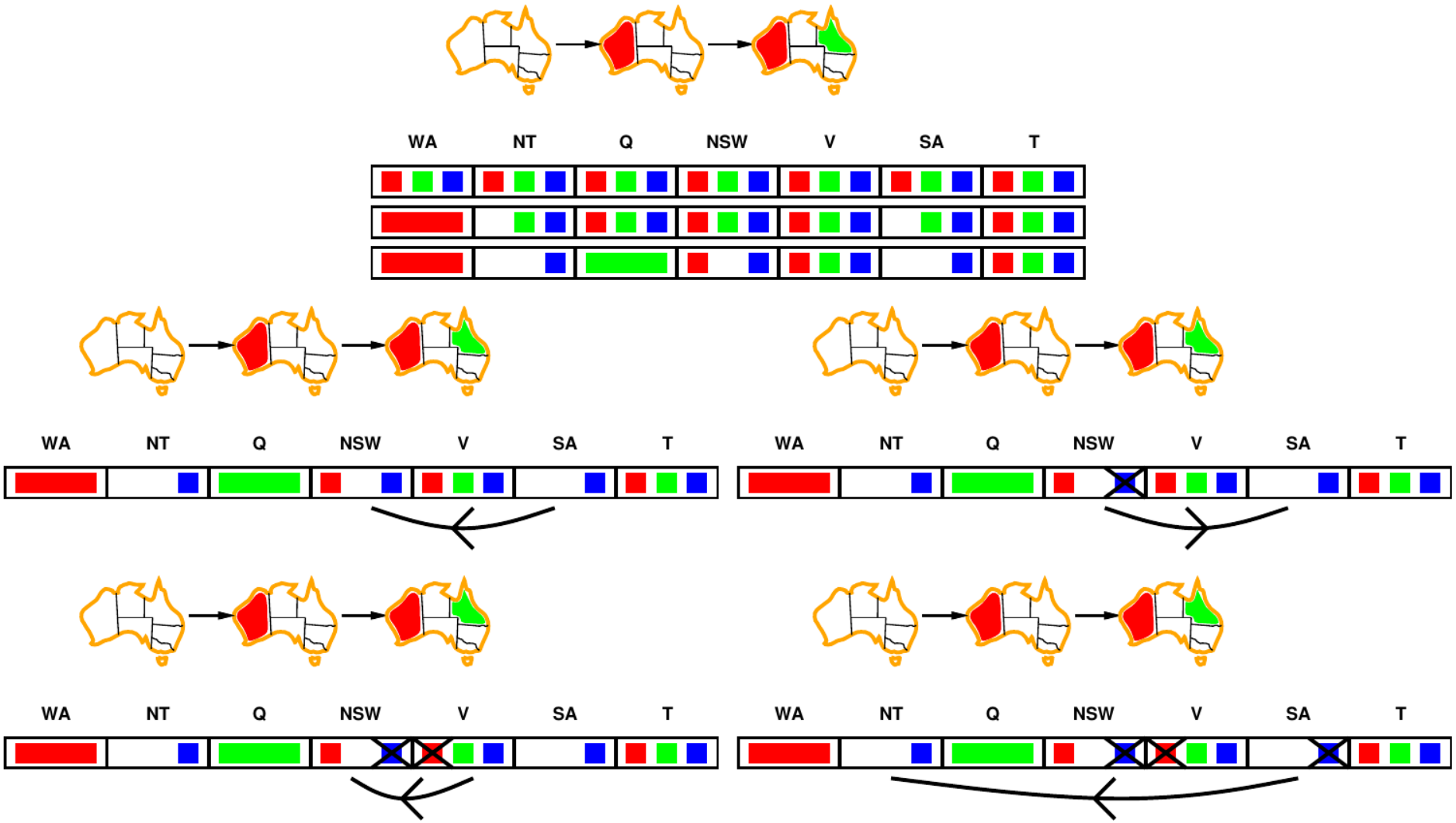
Ví dụ



Ví dụ



Ví dụ


WA NT Q NSW V SA T
WA NT Q NSW V SA T
WA NT Q NSW V SA T
WA NT Q NSW V SA T
WA NT Q NSW V SA T
WA NT Q NSW V SA T

k-hợp lệ

- 1-consistency (nút hợp lệ): miền giá trị một nút thỏa mãn ràng buộc
- 2-consistency (cạnh hợp lệ): với mỗi cặp nút, kiểm tra phép gán hợp lệ từ một nút đến nút còn lại
- k-consistency (k-hợp lệ): với mỗi k nút, kiểm tra các phép gán từ k-1 nút đến nút thứ k
- n-hợp lệ $\Rightarrow$ tìm kiếm không cần quay lại

k càng lớn, chi phí tính toán càng cao

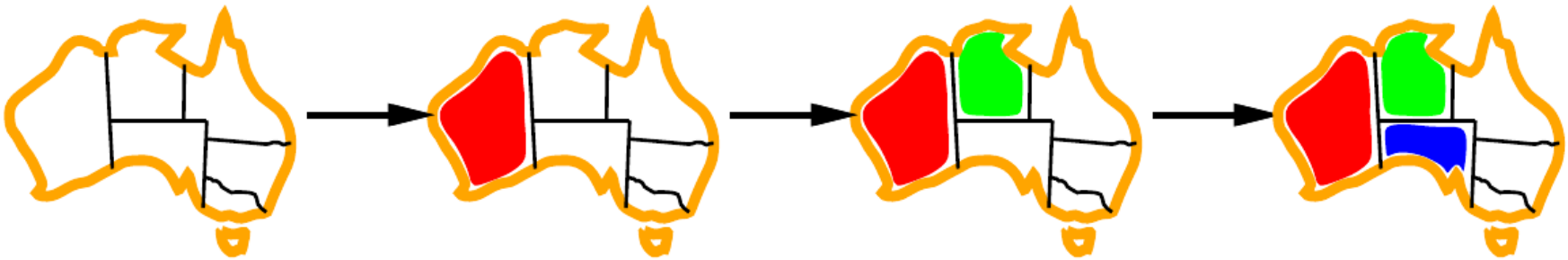
Thứ tự

Minimum Remaining Values

Minimum Remaining Values - MRV

Lựa chọn các biến có tập giá trị hợp lệ nhỏ nhất

- MRV còn được gọi là "fail-first" heuristic

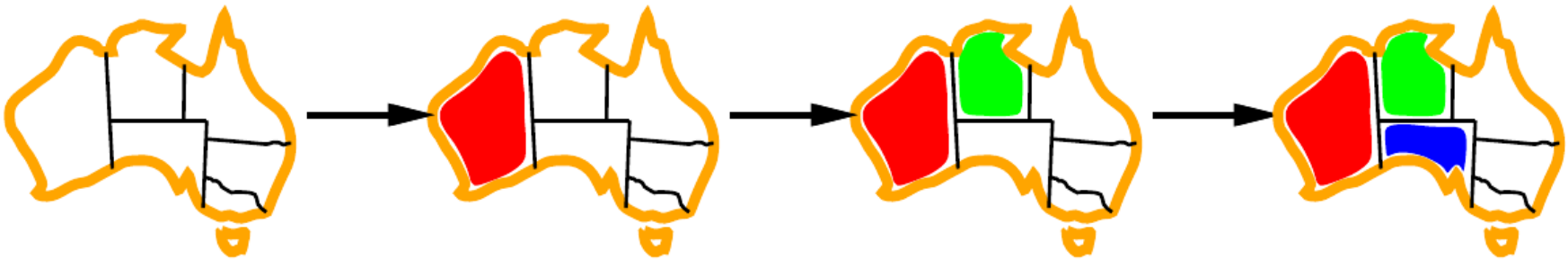


Minimum Remaining Values

Minimum Remaining Values - MRV

Lựa chọn các biến có tập giá trị hợp lệ nhỏ nhất

- MRV còn được gọi là "fail-first" heuristic

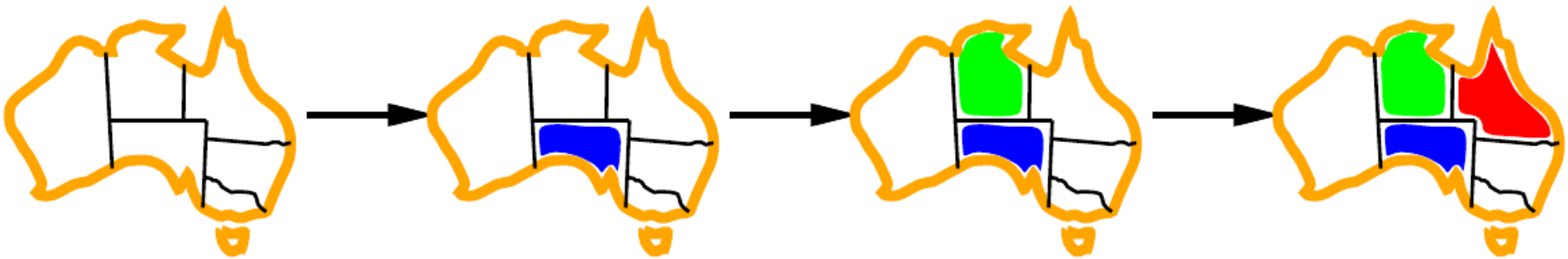


- **Lựa chọn các biến có khả năng tạo lỗi sớm để tỉa nhánh cây tìm kiếm**

Degree Heuristic

Degree Heuristic

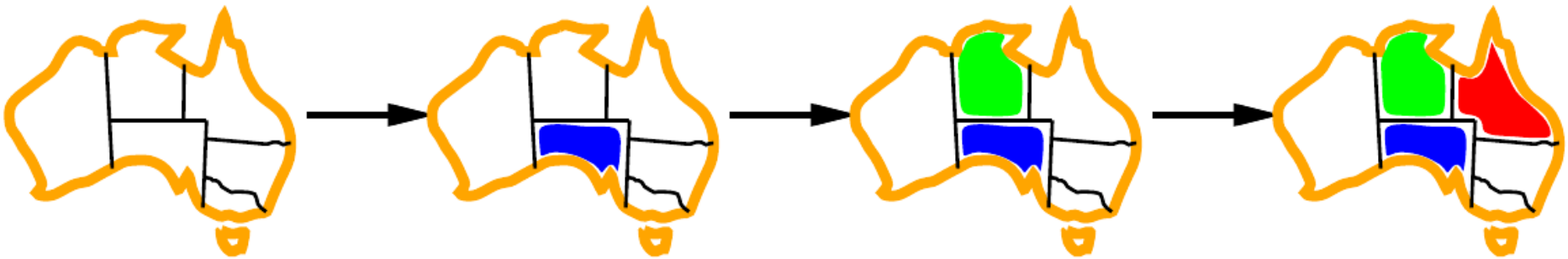
Lựa chọn biến bị ràng buộc nhiều nhất với các biến còn lại



Degree Heuristic

Degree Heuristic

Lựa chọn biến bị ràng buộc nhiều nhất với các biến còn lại

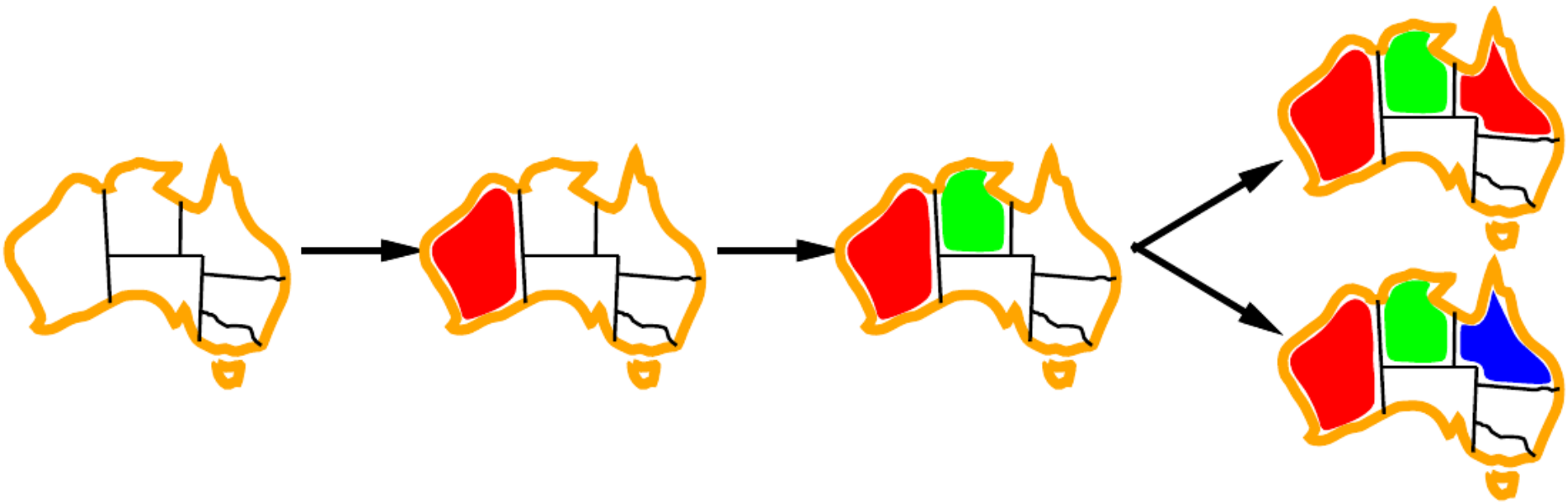


- **Giảm số lượng nhánh con**

Least-constraining value

Least-constraining value

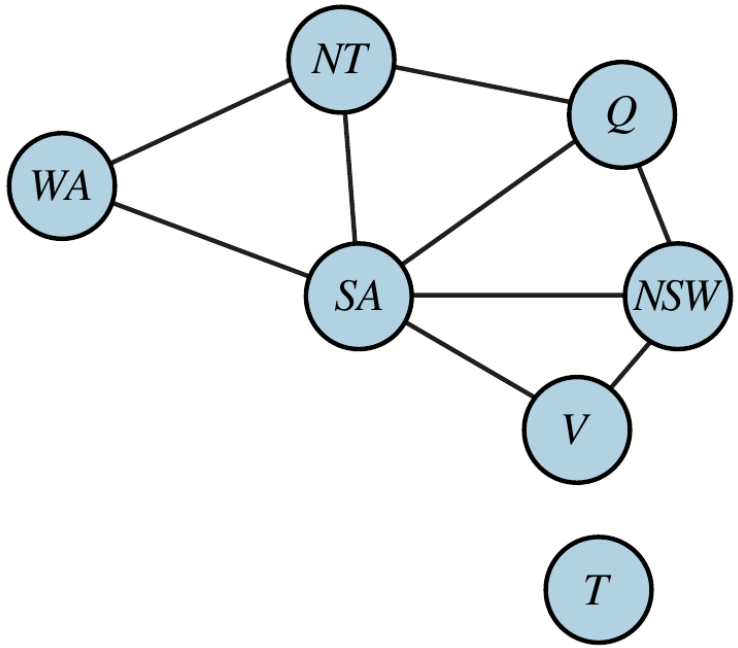
Lựa chọn giá trị có ảnh hưởng tối thiểu đến các giá trị khác



Kiến trúc bài toán

Bài toán con độc lập I

- Một bài toán CSP có thể phân rã thành các **bài toán con độc lập**



Hình 1: Tasmania và vùng đất liền là 2 **bài toán con độc lập**

Bài toán con độc lập II

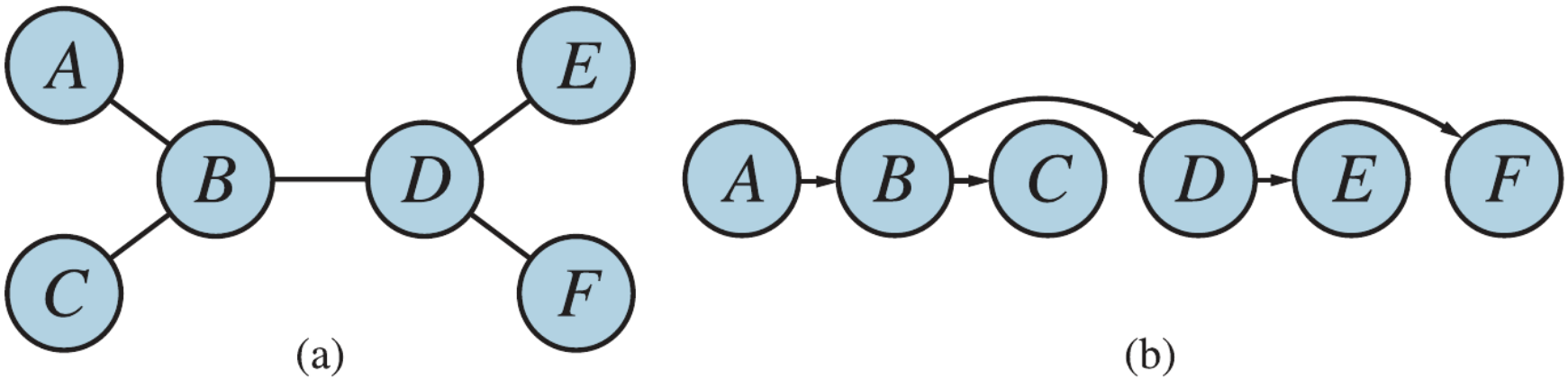
- Nếu S_i là lời giải của bài toán thứ i
- Lời giải của bài toán CSP: $S = \bigcup_{i=1} S_i$
- Giả sử một bài toán CSP với n biến được tách ra thành các bài toán con với trung bình c biến, d là kích thước miền giá trị
  - Chi phí tính toán $O\left(\frac{n}{c} d^c\right)$
  - Chi phí bài toán gốc $O\left(d^n\right)$

CSP có kiến trúc cây

Theorem

Một CSP có kiến trúc cây có thể được giải với độ phức tạp tính toán $O(n \times d^2)$

- Sử dụng sắp xếp topo để sắp thứ tự các biến theo nút gốc

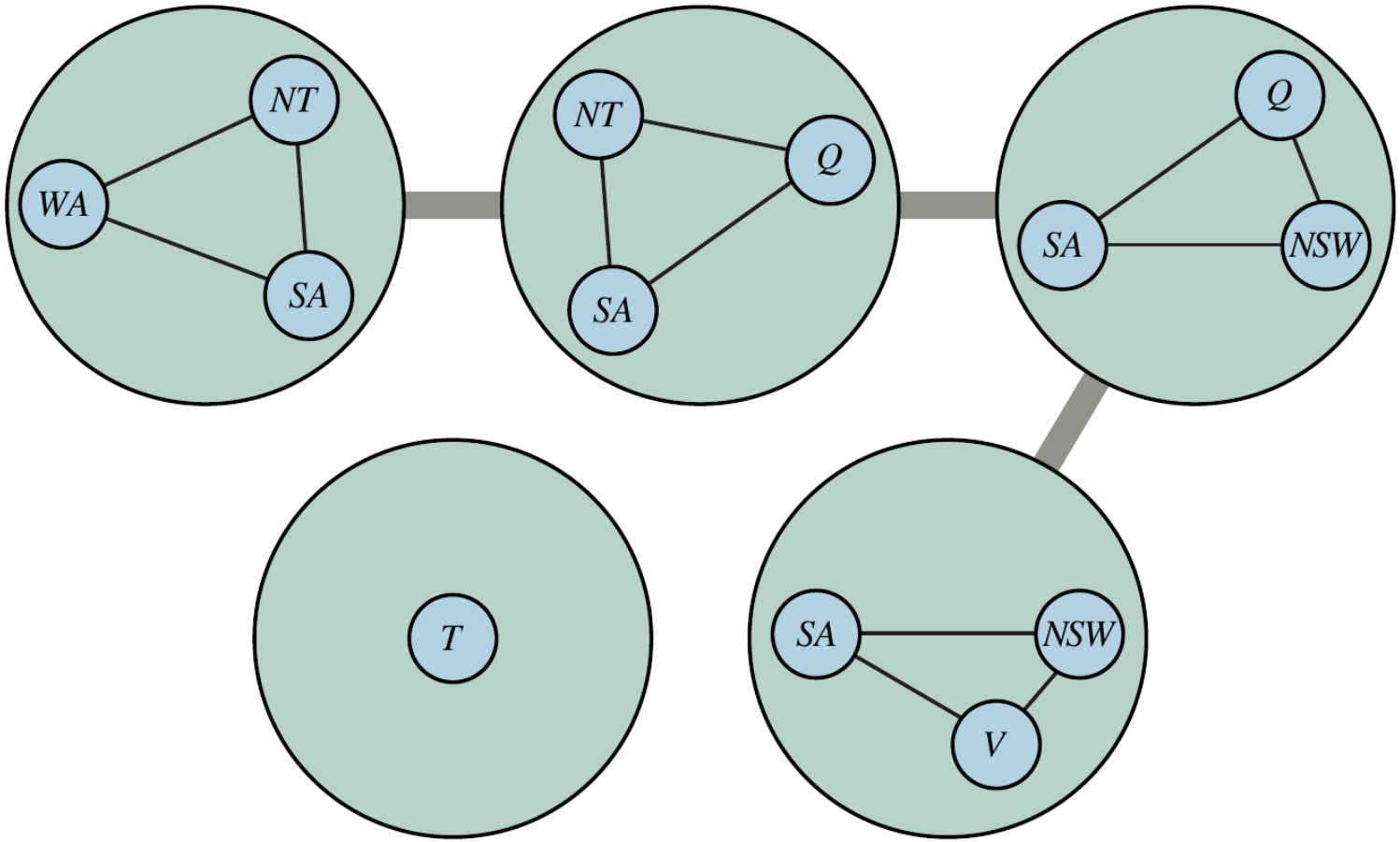


Phân rã cây

- **Ý tưởng:** Biến đổi đồ thị gốc thành cây với các nút là tập biến

1. Mỗi biến cần phải xuất hiện ít nhất một lần
2. Nếu giữa 2 biến có ràng buộc, chúng cần phải xuất hiện cùng nhau ít nhất một lần
3. Nếu một biến xuất hiện trong 2 nút u và v, thì biến này cần phải xuất hiện trên toàn bộ các nút thuộc đường đi từ u đến v

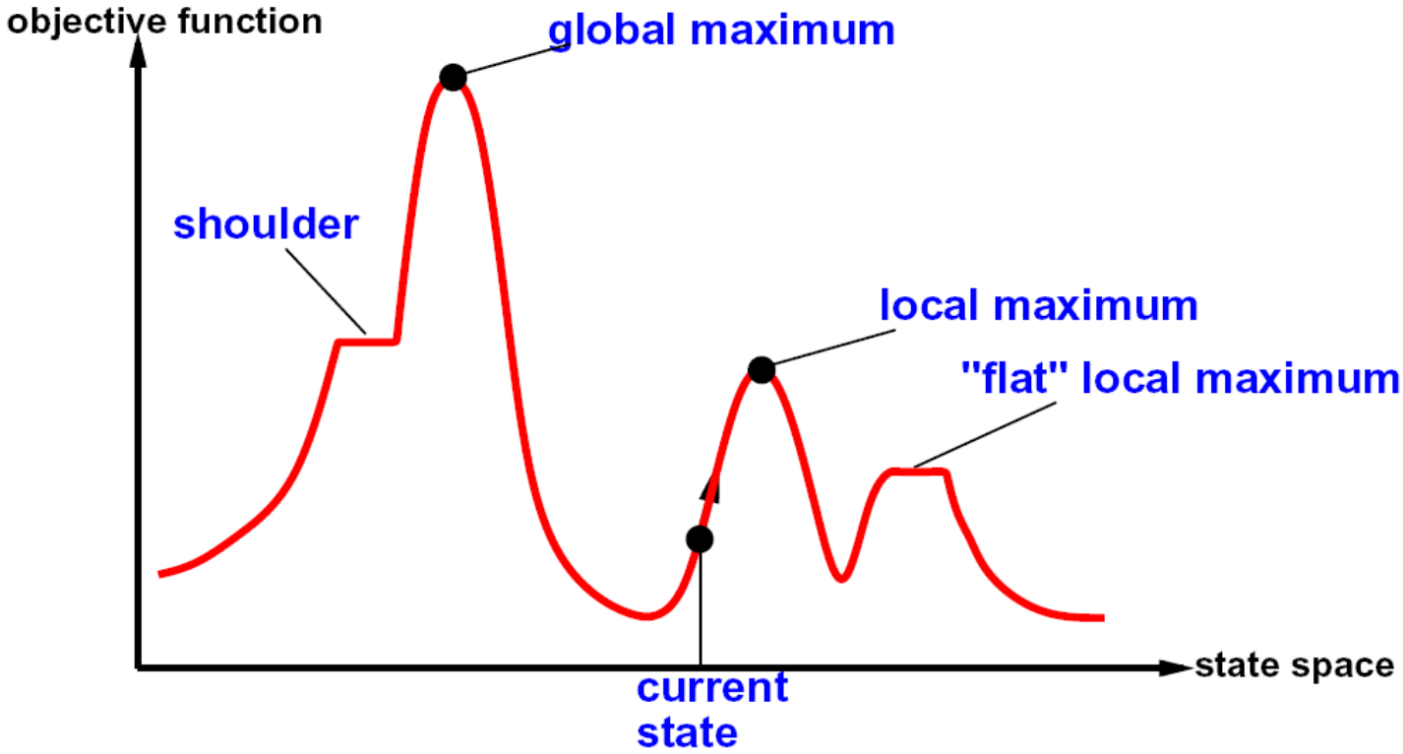
Ví dụ



Tìm kiếm cục bộ

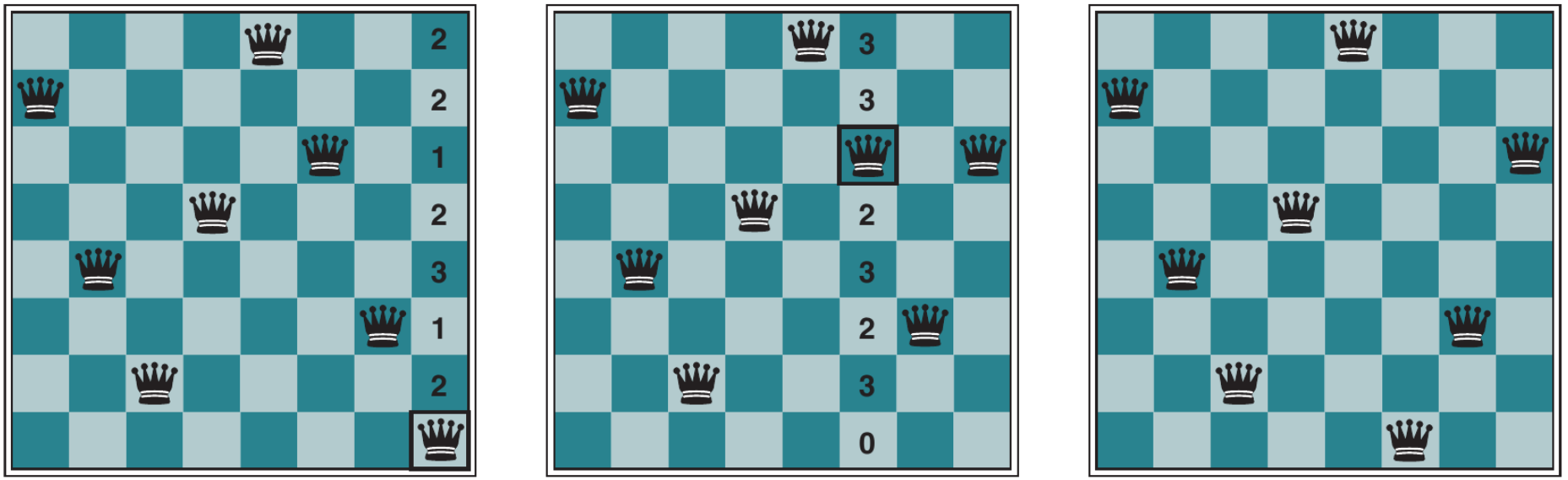
Tìm kiếm cục bộ

- **Ý tưởng:** di chuyển đến lân cận tốt nhất
- Di chuyển đến nút ít bị mâu thuẫn ràng buộc nhất $\rightarrow$ Thuật toán tìm kiếm leo đồi



Bài toán 8-hậu

Bài toán 8-hậu



Hình 2: Lời giải 2 bước cho bài toán 8-hậu

Hiệu quả bất ngờ với nhiều bài toán CSP!!!

Tìm kiếm cục bộ

- Dễ cài đặt
- Hiệu quả tính toán
- Không phải luôn cho kết quả tốt
  - **Thậm chí không đánh giá được lời giải**

Bài toán xếp giỏ ít được ưa thích







Mô hình bài toán

- Mỗi món ăn được biểu diễn theo cặp <giá trị, cân nặng>
- Chúng ta chỉ có thể ăn một lượng thức ăn thấp hơn w
- Vector $\mathbf{v}$ có chiều dài n biểu diễn tập giá trị các món ăn
- Vector $\mathbf{w}$ có chiều dài n biểu diễn tập cân nặng các món ăn
- Vector $\mathbf{p}$ chiều dài n chỉ định món ăn được chọn hay không. $p_i = 1$, chọn món i và ngược lại.

$$maximize \quad \mathbf{p} \cdot \mathbf{v} \tag{1}$$

$$\mathbf{p} \cdot \mathbf{w} \leq w \tag{2}$$

Bài toán tìm kiếm

1. Duyệt tất cả tổ hợp món ăn
2. Xóa các tổ hợp vi phạm ràng buộc
3. Trong các tổ hợp còn lại chọn ra tổ hợp có giá trị cao nhất

Bài toán tìm kiếm

1. Duyệt tất cả tổ hợp món ăn
2. Xóa các tổ hợp vi phạm ràng buộc
3. Trong các tổ hợp còn lại chọn ra tổ hợp có giá trị cao nhất

Thực đơn 100 món?

Bài toán tìm kiếm

1. Duyệt tất cả tổ hợp món ăn
2. Xóa các tổ hợp vi phạm ràng buộc
3. Trong các tổ hợp còn lại chọn ra tổ hợp có giá trị cao nhất

Thực đơn 100 món?

- Số lượng tổ hợp:

  1,267,650,600,228,229,401,496,703,205,376

Giải thuật tham lam?

- Khi "giỏ" chưa đầy, bỏ vào món ăn tốt nhất
- Món ăn tốt nhất?
  - Giá trị nhất
  - Rẻ nhất
  - Dinh dưỡng nhất

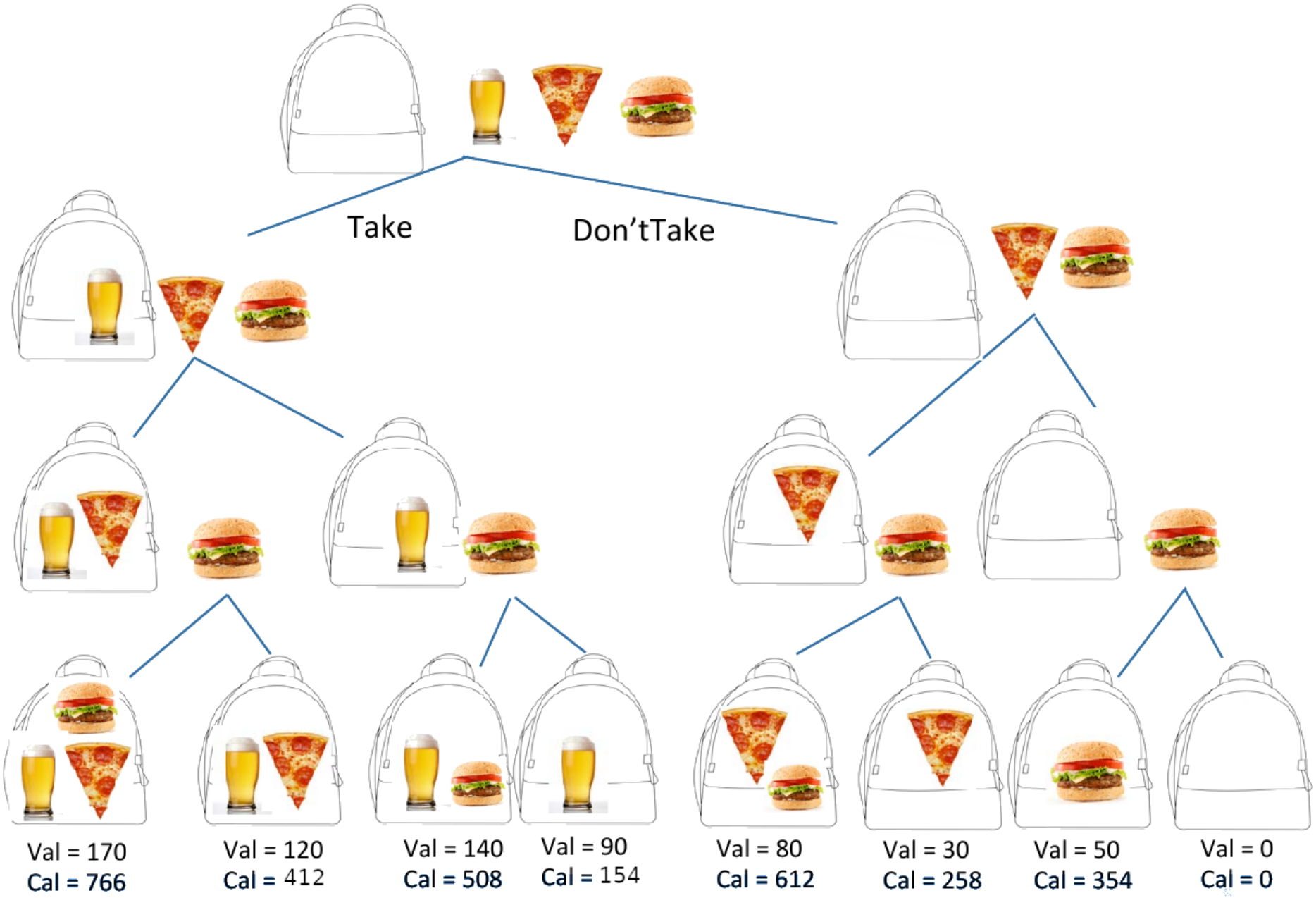
Ví dụ

- Bạn được mời đi ăn cơm
- Bạn biết cách đánh giá các món ăn khác nhau, vd: bạn thích ăn bánh rán nhiều hơn ăn táo
- **Nhưng bạn đang ăn kiêng, bạn không thể tiêu thụ hơn 777 calo mỗi ngày!!!**
- Lựa chọn nên ăn gì là một bài toán xếp giỏ

Một thực đơn

Món ăn	Rượu	Bia	Pizza	Burger	Khoai	Coca	Táo	Bánh rán
Giá trị	89	90	30	50	90	79	90	10
Năng lượng	123	154	258	354	365	150	95	195

- Tìm kiếm quay lui
- Tìm kiếm tham lam


Take
Don'tTake
Val = 170
Cal = 766
Val = 120
Cal = 412
Val = 140
Cal = 508
Val = 90
Cal = 154
Val = 80
Cal = 612
Val = 30
Cal = 258
Val = 50
Cal = 354
Val = 0
Cal = 0

Tổng kết

Tổng kết

- Rất nhiều các vấn đề thực tế có thể được mô tả bằng bài toán thỏa mãn ràng buộc
- Bài toán thỏa mãn ràng buộc (CSP) được thể hiện qua 3 thành phần $\mathcal{X}, \mathcal{D}, \mathcal{C}$
- Tìm kiếm quay lui, một loại tìm kiếm theo chiều sâu được sử dụng để giải bài toán CSP
- Các kỹ thuật nhằm tăng hiệu suất tìm kiếm: suy diễn tiến, k-hợp lệ, thứ tự biến, thứ tự giá trị, phân rã cây
- Tìm kiếm cục bộ

Tài liệu tham khảo

[1] Bùi Tiến Lên, Bộ môn Khoa học máy tính

"Bài giảng môn Cơ sở trí tuệ nhân tạo"

[2] Michael Negnevitsky Russell, S. and Norvig, P. (2021).

"Artificial intelligence: a modern approach."

[3] Berkelev University

"CS188"

[4] Prof. Eric Grimson, Prof. John Guttag

"MIT 6.0002 Course".