Constraint Satisfaction Problems
(ARTIFICIAL INTELLIGENCE)

(slides adapted and revised from Dan Klein, Pieter Abbeel, Anca Dragan, et al)

Cải thiện hiệu quả của thuật toán quay lui

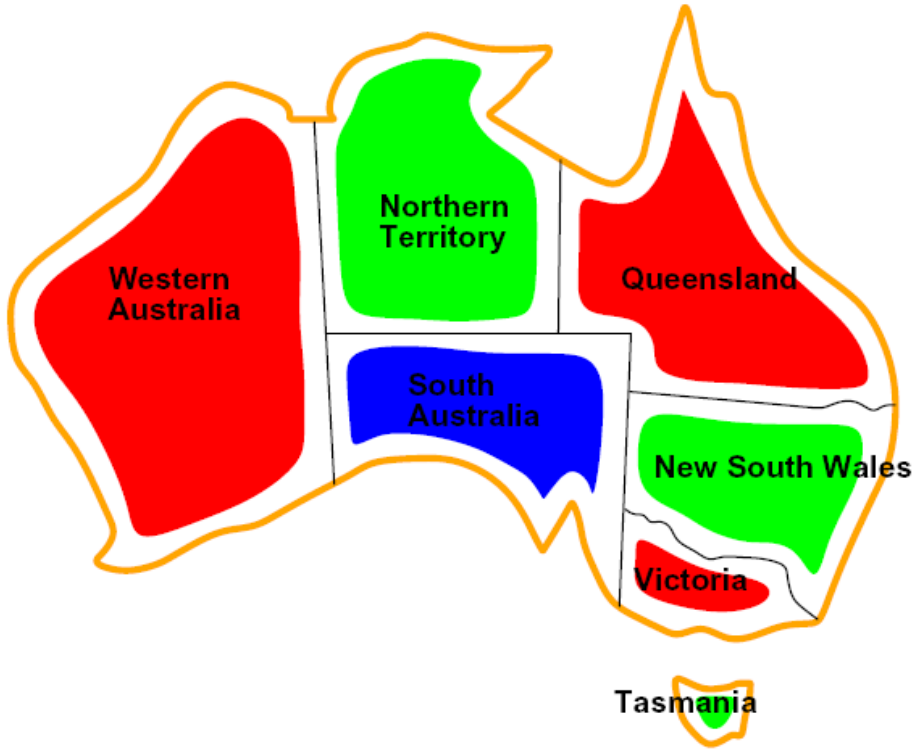
```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return BACKTRACK({ }, csp)

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
```

Cải thiện hiệu quả của thuật toán quay lui

- Select-Unassigned-Variable
  - Chọn biến chưa được gán giá trị
- Order-Domain-Value
  - Chọn giá trị để gán cho biến chưa được gán
- Inference
  - Kỹ thuật phát hiện lỗi (Những biến và giá trị khiến thuật toán phải quay lui)
- Backtrack
  - Các kỹ thuật quay lui

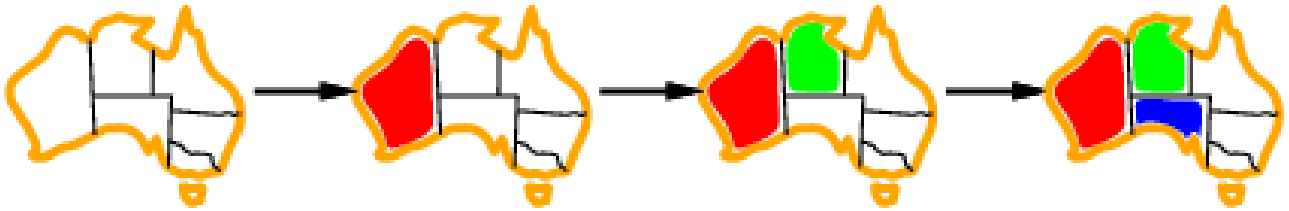
Chọn biến chưa được gán giá trị

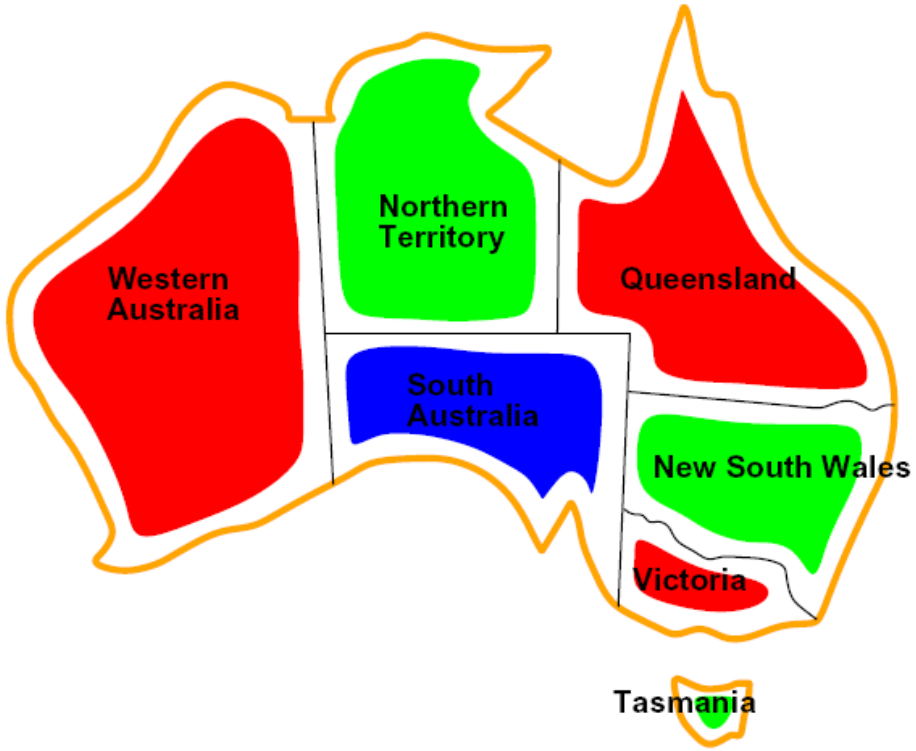
- Chọn biến ngẫu nhiên
- Chọn biến bị ràng buộc nhiều nhất
- Chọn biến có nhiều ràng buộc nhất đối với các biến còn lại



Chọn biến chưa được gán giá trị

- Chọn biến ngẫu nhiên

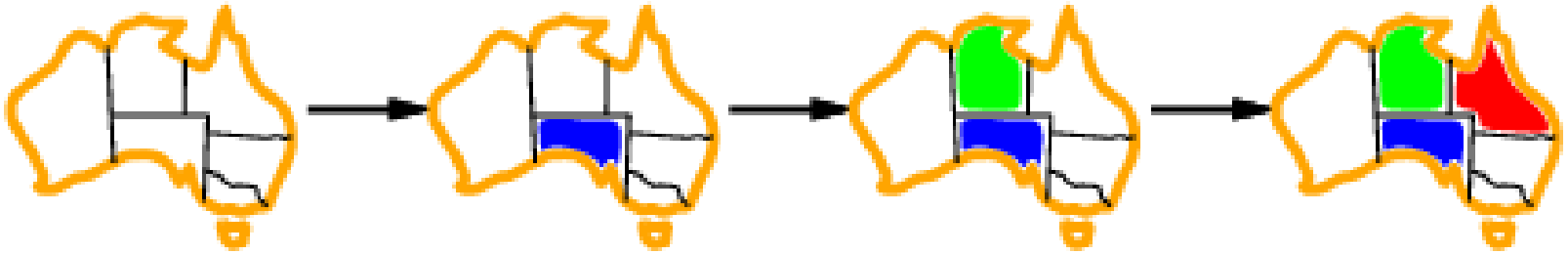


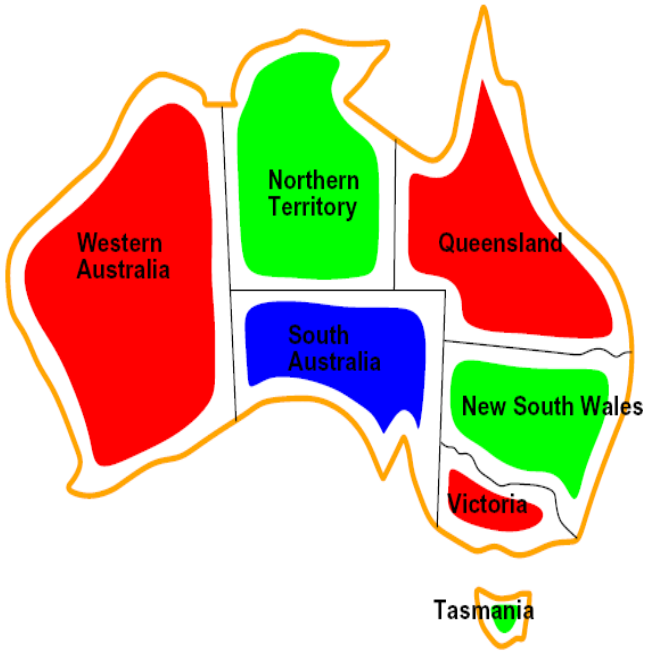


- Ví dụ: Chọn WA->NT->SA->….

Chọn biến chưa được gán giá trị

- Chọn biến bị ràng buộc nhiều nhất:
  - Nghĩa là chọn biến có ít giá trị hợp lệ nhất





  - Ví dụ: Chọn SA->NT->Q->….
  - Nếu có 2 biến có cùng số lượng ràng buộc thì chọn theo chiến lược nào?

Chọn biến chưa được gán giá trị

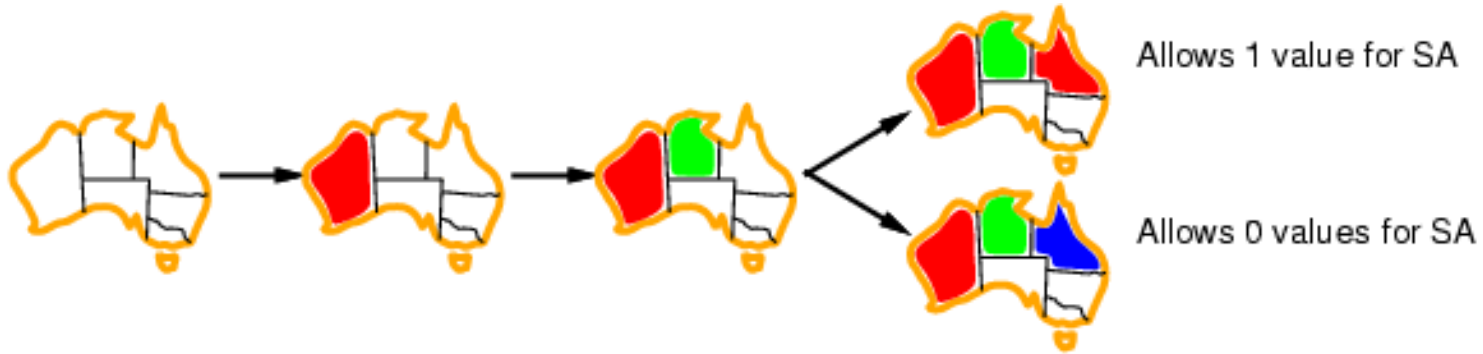
- Chọn biến có nhiều ảnh hưởng nhất đến domain của các biến còn lại

Biến được gán giá trị	Các biến bị ảnh hưởng đến domain							
	PT	VP	BG	BN	HD	HY	HP	HN
HN=r	Có	Có	Có	Có	Không	Có	Không	X
VP=r	Có	X	Không	Không	Không	Không	Không	Có
BG=r	Không	Có	X	Có	Có	Không	Không	Có
BN=r	Không	Không	Có	X	Có	Có	Không	Có
HD								
HY								
HP								
PT								

- Chọn biến HN để gán giá trị

Chọn giá trị để gán cho biến

- Với một biến để gán, hãy chọn giá trị ít ràng buộc nhất.
- Ví dụ:



- Kết hợp các phương pháp này sẽ làm cho 1000 quân hậu trở nên khả thi

Cài đặt Inference

- Forward checking
- Maintaining arc consistency

Kiểm tra chuyển tiếp

- Ý tưởng:
  - Theo dõi các giá trị hợp lệ còn lại cho các biến chưa được gán
  - Kết thúc tìm kiếm khi bất kỳ biến nào không có giá trị hợp lệ

- Cách làm:
  - Khi một biến được gán giá trị var = value:
    - Thì kỹ thuật này được thực hiện để kiểm tra xem phép gán đó có thỏa mãn không
    - Chỉ kiểm tra những biến neighbors của nó có giá trị nào và có biến nào có giá trị bằng rỗng hay không
    - Kết quả trả về là True hoặc False

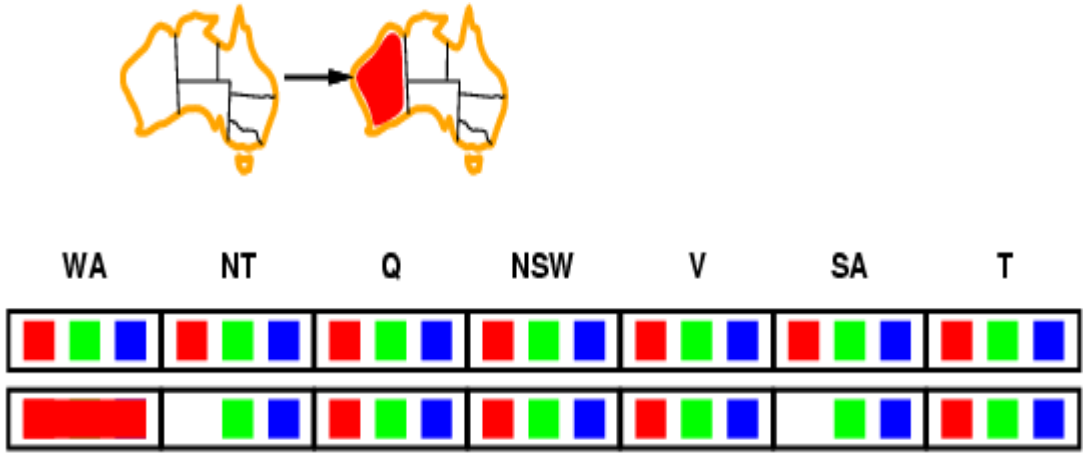
Kiểm tra chuyển tiếp

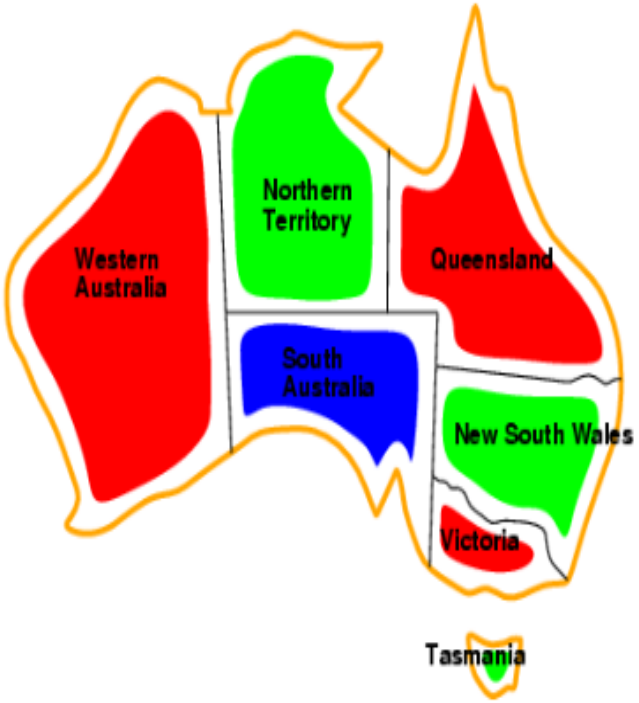
	PhuTho	VinhPhuc	BacNinh	BacGiang	HungYen	Hải phòng	
HaNoi=blue	R,g	R,g	R,g	R,g	R,g	x	
HaiDuong=g	x	x	R	R	R	R,b	
BacNinh=r	x	x	Đang xét	Rỗng	Rỗng	x	Dừng

- Ví dụ:
  - Khi gán HN bằng xanh thì những neighbors của nó là PT, VP, BN, BG, và HY sẽ được kiểm tra để xem có biến nào không có giá trị để gán hay không? Không =>Gán được
  - Khi gán BN=r thì có BG, HY không có giá trị để gán =>Không gán được

Kiểm tra chuyển tiếp

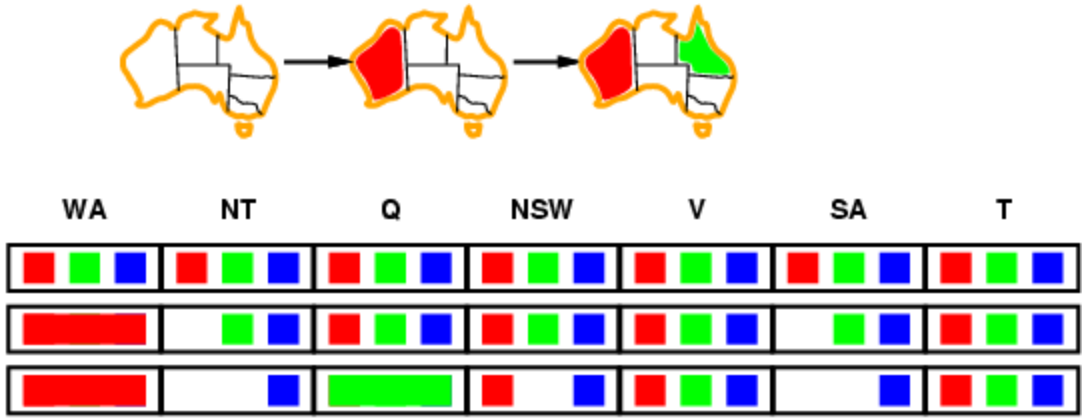
- Ví dụ:

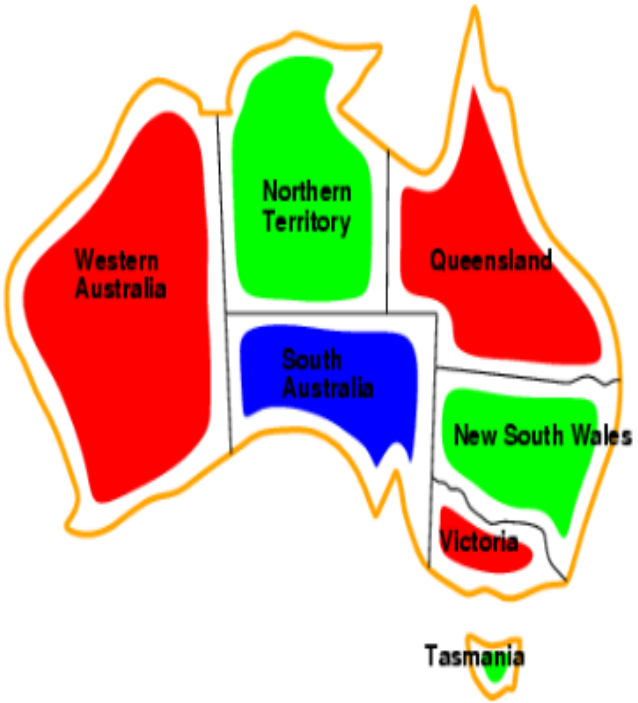




Kiểm tra chuyển tiếp

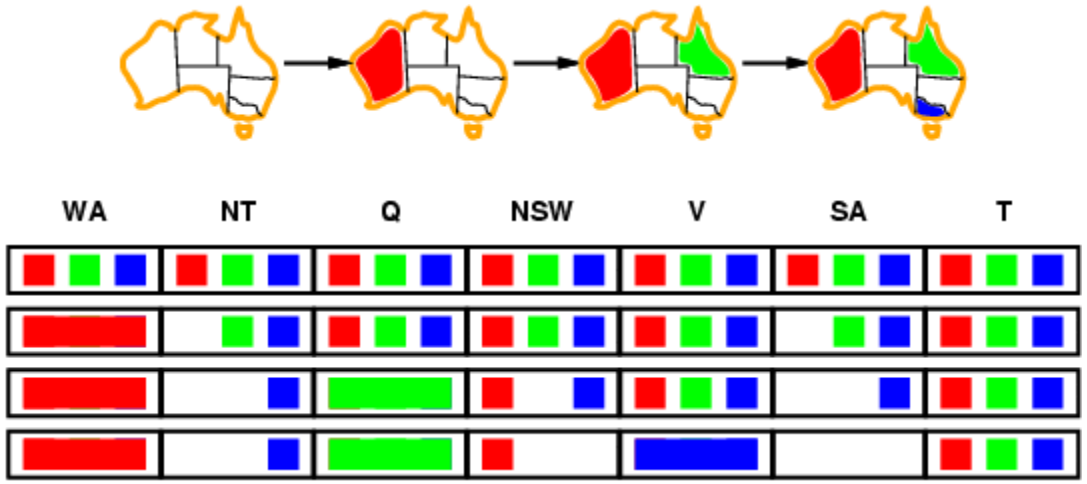
Ví dụ:

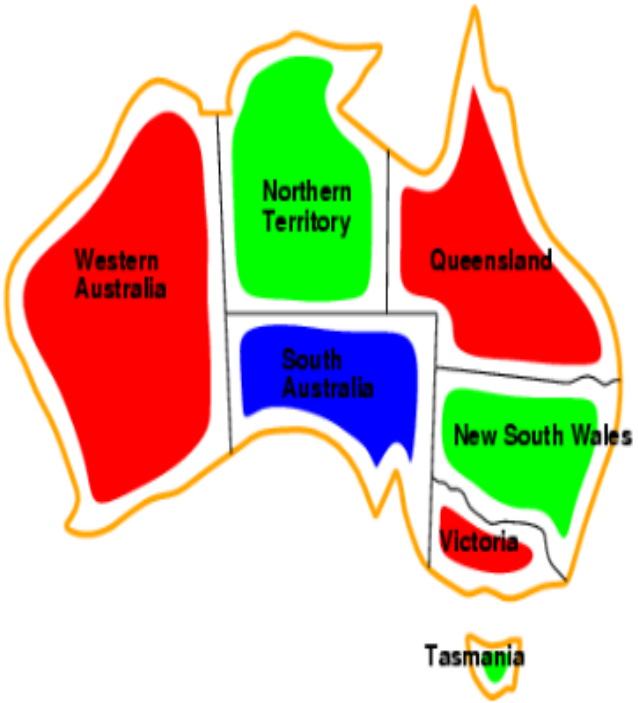




Kiểm tra chuyển tiếp

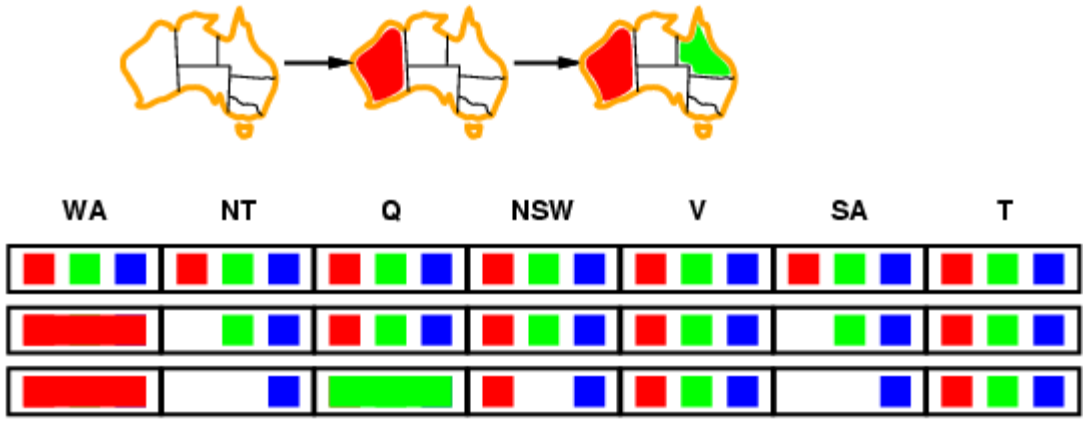
- Ví dụ:

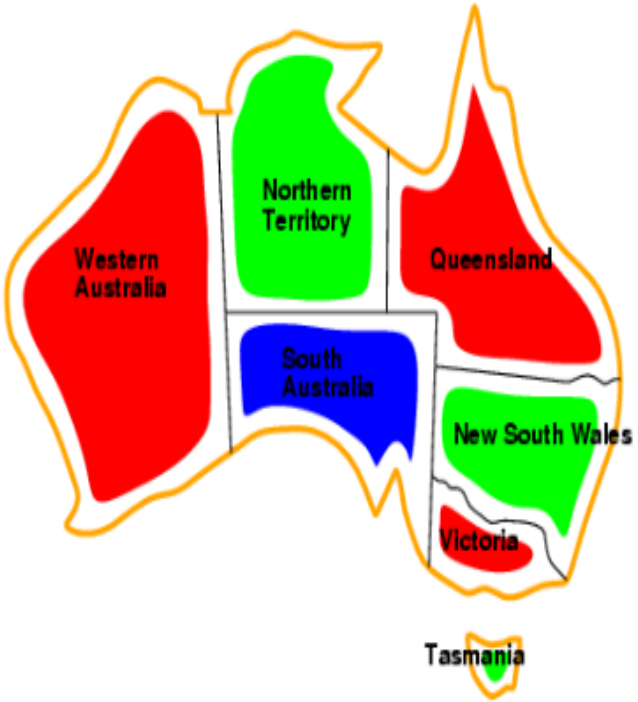




Kiểm tra chuyển tiếp

- Kiểm tra chuyển tiếp truyền thông tin từ các biến đã gán đến các biến chưa gán, nhưng không cung cấp khả năng phát hiện sớm cho tất cả các lỗi.
- Ví dụ:





  - NT và SA không thể cùng có màu xanh! => lỗi
  - Các thuật toán truyền ràng buộc liên tục thực thi các ràng buộc cục bộ…

Maintaining arc consistency

- Ý tưởng: đảm bảo mội ràng buộc 2 biến đều thỏa mãn
- Không chỉ kiểm tra từng neighbors của biến đang được gán mà kỹ thuật này sẽ kiểm tra tất cả binary contraints

	PhuTho	VinhPhuc	BacNinh	BacGiang	HungYen	Hải phòng	
HaNoi=blue	R,g	R,g	R,g	R,g	R,g	x	
HaiDuong=g	x	x	R	R	R	R,b	Dừng
BacNinh=r	x	x	Đang xét	Rỗng	Rỗng	x	

– Kỹ thuật này sẽ chạy nhanh hơn kỹ thuật Kiểm tra chuyển tiếp
– Kỹ thuật này phát hiện lỗi sớm hơn kiểm tra chuyển tiếp
– Có thể chạy như một bộ xử lý trước hoặc sau mỗi lần gán

Chiến lược Backtrack

- Giả sử ta có lần lượt các phép gán: HN=b, BG=r, HY=g, VP=r, PT=g, HP=b
- Và biến được gán tiếp theo giả xử ta chọn BN thì lúc này BN=rỗng, lúc này backtrack được thực hiện để chọn lại biến, và biến quay lại là HP, và HP cũng ko thay đổi được vấn đề gì. Và phải đến đổi HY=r thì mới thay đổi được vấn đề. => Đây là cách backtrack tự nhiên.

Chiến lược Backtrack

- Mỗi lần backtrack tự nhiên sẽ rất lãng phí với các trường hợp không thay đổi được kết quả. Vì vậy có cách thứ 2 là Backjumping bằng cách cài đặt 1 tập gọi là conflict set (là những biến đã được gán và là hàng xóm của biến hiện tại).
- Ví dụ: quay lại ví dụ trên với BN thì conflict set sẽ là [HN, BG, HY]. Mục tiêu là để khi gặp vấn đề phải back thì chúng ta nhảy thẳng vào xét các giá trị conflict của BN luôn.

Thuật toán nhất quán cung AC-3

```
function AC-3( csp) returns the CSP, possibly with reduced domains
    inputs: csp, a binary CSP with variables {X_1, X_2, ..., X_n}
    local variables: queue, a queue of arcs, initially all the arcs in csp

    while queue is not empty do
        (X_i, X_j) ← REMOVE-FIRST(queue)
        if RM-INCONSISTENT-VALUES(X_i, X_j) then
            for each X_k in NEIGHBORS[X_i] do
                add (X_k, X_i) to queue

function RM-INCONSISTENT-VALUES( X_i, X_j) returns true iff remove a value
    removed ← false
    for each x in DOMAIN[X_i] do
        if no value y in DOMAIN[X_j] allows (x,y) to satisfy constraint(X_i, X_j)
            then delete x from DOMAIN[X_i]; removed ← true
    return removed
```

- Độ phức tạp thời gian: O(#constraints |domain|3)

Checking consistency of an arc is O(|domain|^2)

Source: (Russell, 2016)

Tìm kiếm cục bộ cho CSPs

- Sử dụng biểu diễn trạng thái hoàn chỉnh
  - Trạng thái ban đầu = tất cả các biến được gán giá trị
  - Trạng thái kế thừa = thay đổi giá trị
- Đối với CSP
  - cho phép các trạng thái có ràng buộc chưa được thỏa mãn (không giống như quay lui)
  - toán tử gán lại giá trị biến
  - leo đồi với n-queens là một ví dụ (chúng ta đã thấy trong các ghi chú trước đó)
- Lựa chọn biến: chọn ngẫu nhiên bất kỳ biến nào có xung đột
- Lựa chọn giá trị: min-conflicts heuristic
  - Chọn giá trị mới tạo ra số lượng xung đột tối thiểu với các biến khác

Tìm kiếm cục bộ cho CSPs

```
function MIN-CONFLICTS(csp, max_steps) return solution or failure
    inputs: csp, a constraint satisfaction problem
            max_steps, the number of steps allowed before giving up

    current ← an initial complete assignment for csp
    for i = 1 to max_steps do
            if current is a solution for csp then return current
            var ← a randomly chosen, conflicted variable from VARIABLES[csp]
            value ← the value v for var that minimize CONFLICTS(var,v,current,csp)
            set var = value in current
    return failure
```

Source: (Russell, 2016)

Demo

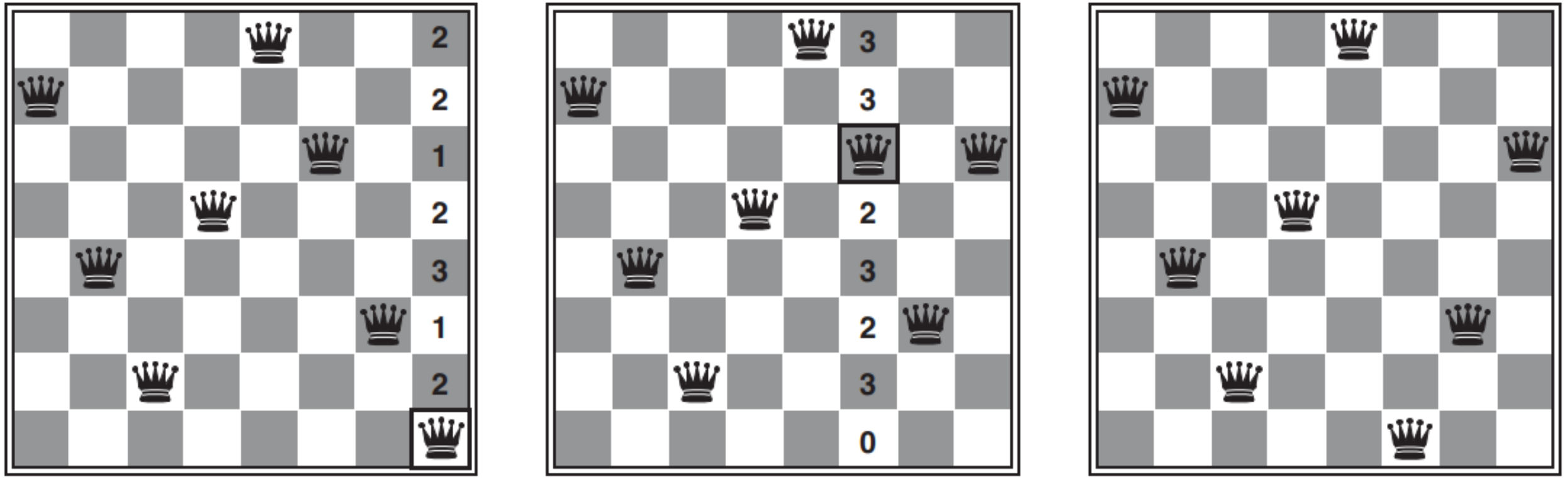


Figure 6.9 A two-step solution using min-conflicts for an 8-queens problem. At each stage, a queen is chosen for reassignment in its column. The number of conflicts (in this case, the number of attacking queens) is shown in each square. The algorithm moves the queen to the min-conflicts square, breaking ties randomly.

Source: (Russell, 2016)

Thanks for your attention!

Q&A