Trí tuệ nhân tạo (Artificial Intelligence)

Bài toán thõa mãn ràng buộc

By Hoàng Hữu Việt
Email: viethh@vinhuni.edu.vn
Viện Kỹ thuật và Công nghệ, Đại học Vinh

Vinh, 3/2019

Tài liệu

- Tài liệu chính

  [1] Stuart Russell, Peter Norvig. Artificial Intelligence. A modern approach. 3rd ed. Prentice Hall, 2009.

- Tài liệu khác

  [2] Milos Hauskrecht. Artificial Intelligence, 2013.
  people.cs.pitt.edu/~milos/courses/cs1571-Fall2013/

Nội dung

- Bài toán thõa mãn ràng buộc
- Lan truyền ràng buộc (constraint propagation)
- Tìm kiếm quay lui (backtracking search)
- Tìm kiếm cục bộ (local search)

Bài toán thõa mãn ràng buộc

- Bài toán thõa mãn ràng buộc gồm bộ ba (X,D,C):
  - $X = \{X_1, X_2, \ldots, X_n\}$ là một tập hữu hạn các biến.
  - $D = \{D_1, D_2, \ldots, D_n\}$ là một tập hữu hạn miền giá trị của các biến với D_i là miền giá trị của biến V_i.
  - $C = \{C_1, C_2, \ldots, C_t\}$ là tập các ràng buộc của các biến.
- Một phép gán phù hợp (consistent) là một phép gán giá trị cho các biến mà không vi phạm các ràng buộc.
- Một phép gán hoàn chỉnh (complete assignment) là một phép gán mà mọi biến được gán giá trị.
- Một nghiệm của bài toán là một phép gán phù hợp và hoàn chỉnh, tức là gán các giá trị $v_i \in D_i$ cho mỗi biến X_i $(i = 1..n)$ để các ràng buộc được thõa mãn.

Bài toán thõa mãn ràng buộc

- Ví dụ 1. tìm các giá trị $x_1 \in \{1, 2, 3\}$, $x_2 \in \{1, 2, 3\}$ và $x_3 \in \{2,3\}$ để thõa mãn các điều kiện:

$$x_1 > x_2, x_2 \neq x_3, x_2 + x_3 > 4.$$

- Biểu diễn theo bài thoán thõa mãn ràng buộc:
  - $V = \{x_1, x_2, x_3\}$
  - $D = \{D_1, D_2, D_3\}$; $D_1 = \{1, 2, 3\}$, $D_2 = \{1, 2, 3\}$, $D_3 = \{2,3\}$
  - $C = \{x_1 > x_2, x_2 \neq x_3, x_2 + x_3 > 4\}$

- Nghiệm $x_1 = 3, x_2 = 2, x_3 = 3$.

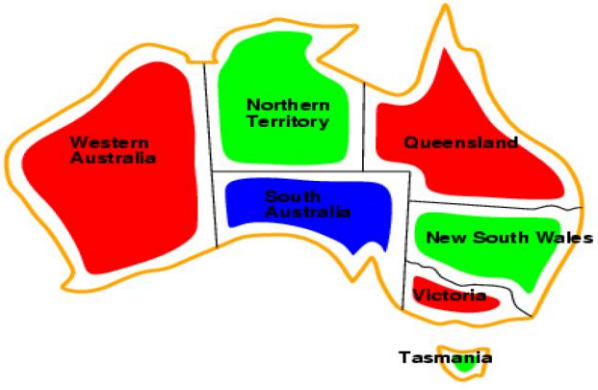
Bài toán thõa mãn ràng buộc

- Ví dụ 2. bài toán tô màu bản đồ
  - Định nghĩa tập biến $X = \{WA, NT, Q, NSW, V, SA, T\}$.
  - Miền giá trị của các biến: $D_i = \{\text{red, green, blue}\}$.
  - Các ràng buộc: các vùng cạch nhau phải có màu khác nhau:

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

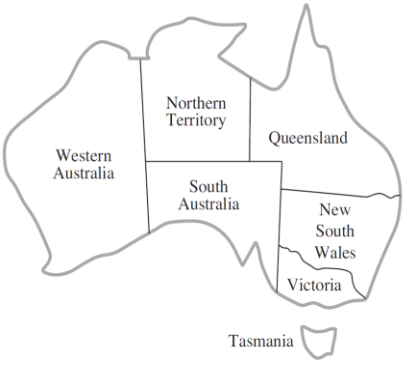
Bài toán thõa mãn ràng buộc

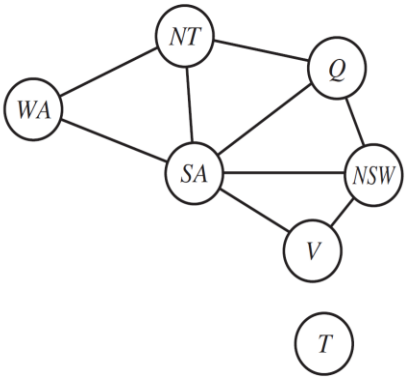
- Ví dụ 2. bài toán tô màu đồ thị
  - Bài toán có nhiều nghiệm.
  - Một nghiệm của bài toán: {WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green}



Bài toán thõa mãn ràng buộc

- Ví dụ 2. bài toán tô màu đồ thị
  - Bài toán có thể được biểu diễn dạng đồ thị:
    - Các đỉnh tương ứng với các biến.
    - Các cạnh tương ứng với các ràng buộc.





Bài toán thõa mãn ràng buộc

- Ví dụ 3. bài toán mật mã số học
  - Tìm các chữ số thay thế cho các chữ cái với điều kiện mỗi chữ cái ứng với một chữ số khác nhau và thõa mãn:

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

  - Định nghĩa tập biến $X = \{T, W, O, F, U, R\}$.
  - Miền giá trị của các biến: $D_T = D_F = \{1, 2, \ldots, 9\}$, $D_W = D_O = D_U = D_R = \{0, 1, \ldots, 9\}$.
  - Các ràng buộc: $Allfdiff(T, W, O, F, U, R\}$, nghĩa là các biến nhận các giá trị khác nhau.
  - Nghiệm ???

Dạng biến và kiểu ràng buộc

- Các dạng biến
  - Biến rời rạc
    - Miền giá trị hữu hạn: ví dụ $x_1 \in \{1,2,3\}$, $x_2 \in \{1,2,3\}, \ldots$
    - Miền giá trị vô hạn: tập số nguyên.
  - Biến liên tục: ví dụ biến nhận giá trị là số thực.
- Các kiểu ràng buộc
  - Ràng buộc đơn (unary constraint): chỉ ràng buộc bởi 1 biến. Ví dụ SA ≠ green.
  - Ràng buộc nhị phân (binary constraint): ràng buộc bởi 2 biến. Ví dụ SA ≠ NSW.
  - Ràng buộc bậc cao (higher-order constraint): ràng buộc bởi nhiều hơn 2 biến.

Giải bài toán thõa mãn ràng buộc

- Lan truyền ràng buộc (constraint propagation)
- Tìm kiếm quay lui (backtracking search)
- Tìm kiếm cục bộ (local search)

Lan truyền ràng buộc

- Lan truyền ràng buộc
  - Sử dụng các ràng buộc để giảm số lượng các giá trị hợp lệ của một biến.
  - Từ đó, có thể giảm các giá trị hợp lệ cho các biến khác.
- Các loại ràng buộc
  - Phù hợp đỉnh (node consistency)
  - Phù hợp cạnh (arc consistency)
  - Phù hợp đường (path consistency)
  - K – phù hợp (k – consistency)

Phù hợp đỉnh (node consistency)

- Một biến (tương ứng với một đỉnh trong đồ thị) là đỉnh phù hợp nếu tất cả các giá trị trong miền giá trị của biến thõa mãn các ràng buộc đơn biến (unary constraints).
  - Ví dụ biến SA ban đầu có miền giá trị {red, green, blue} và biến SA là nút phù hợp bằng cách loại giá trị green để SA có miền giá trị {red,blue}.
- Một đồ thị là phù hợp đỉnh nếu mọi biến trong đồ thị là đỉnh phù hợp.

Phù hợp cạnh (arc consistency)

- Một biến là phù hợp cạnh nếu mọi giá trị trong miền giá trị của nó thõa mãn các ràng buộc nhị phân của biến.
  - X_i là phù hợp cạnh với X_j nếu $\forall v_i \in D_i$, $\exists v_j \in D_j$ để (v_i, v_j) thõa mãn ràng buộc trên cạnh (X_i, X_j).
- Một đồ thị là phù hợp cạnh nếu mọi biến là phù hợp cạnh với các biến khác.
  - Ví dụ: $V = \{X, Y\}$, $D_X = \{0,1,2,3\}$, $D_Y = \{0,1,4,9\}$, C = $\{Y = X^2\}$.
- Có thể đạt được thõa mãn ràng buộc trên cạnh (V_i, V_j) bằng cách xóa các giá trị trong miền D_i mà không thõa mãn ràng buộc trên cạnh.

Phù hợp cạnh (arc consistency)

- Thuật toán AC3

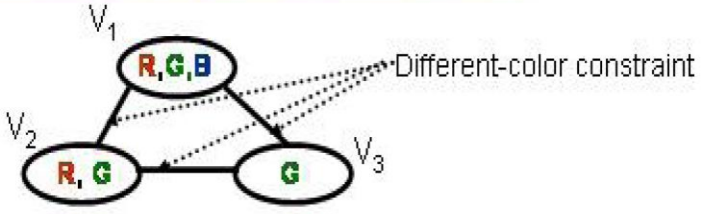
```
function AC-3( csp) returns false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components (X, D, C)
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true
```

```
function REVISE( csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

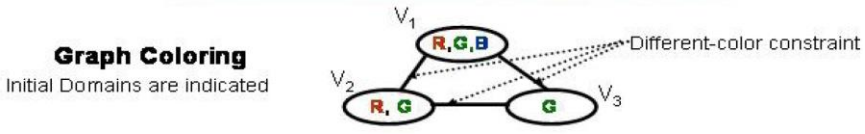
$D_1 = \{R, G, B\}$, $D_2 = \{R, G\}$, $D_3 = \{G\}$.



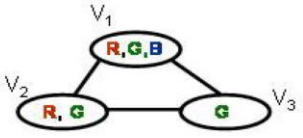
Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$



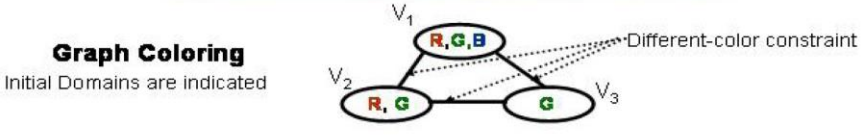
Arc examined	Value deleted



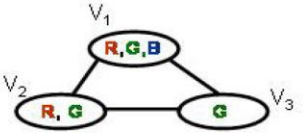
Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$



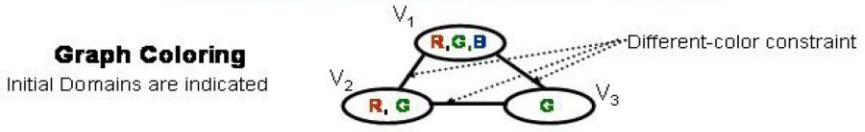
Arc examined	Value deleted
$V_1 - V_2$	none



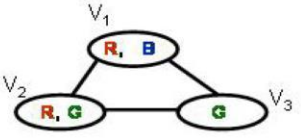
Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$



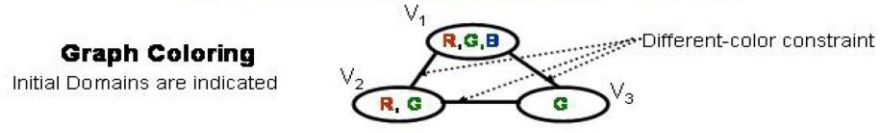
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)



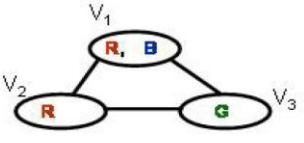
Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$



Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)

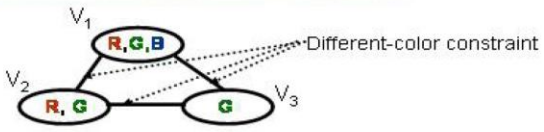


Phù hợp cạnh (arc consistency)

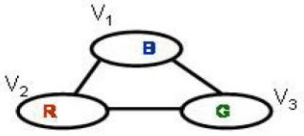
- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$

Graph Coloring
Initial Domains are indicated



Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)

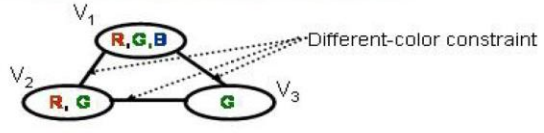


Phù hợp cạnh (arc consistency)

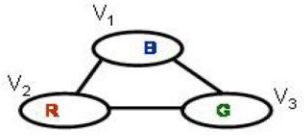
- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$

Graph Coloring
Initial Domains are indicated



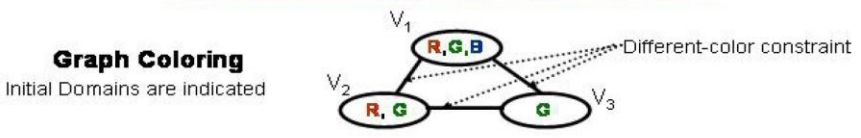
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)
$V_1 - V_3$	none



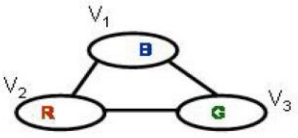
Phù hợp cạnh (arc consistency)

- Ví dụ bài toán tô màu đồ thị:

$$D_1 = \{R, G, B\}, D_2 = \{R, G\}, D_3 = \{G\}.$$



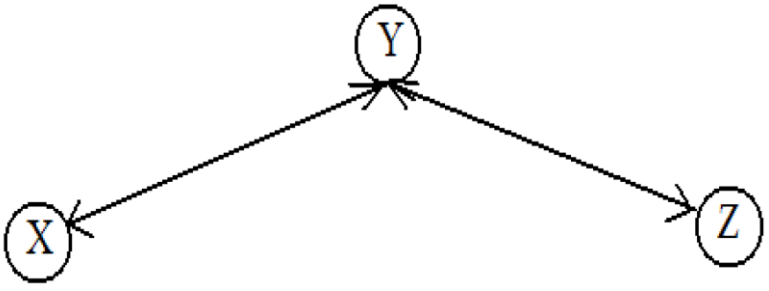
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)
$V_1 - V_3$	none
$V_2 - V_3$	none



Phù hợp cạnh (arc consistency)

- Ví dụ cho bài toán thõa mãn ràng buộc:
  - Các biến: X, Y, Z
  - Các miền giá trị:

    $D_x = (4, 5, 6, 7), D_y = (4, 5, 6, 8, 9), D_z = (3, 5, 6, 7, 9)$
  - Các ràng buộc: $X = Y$, $Y = Z$

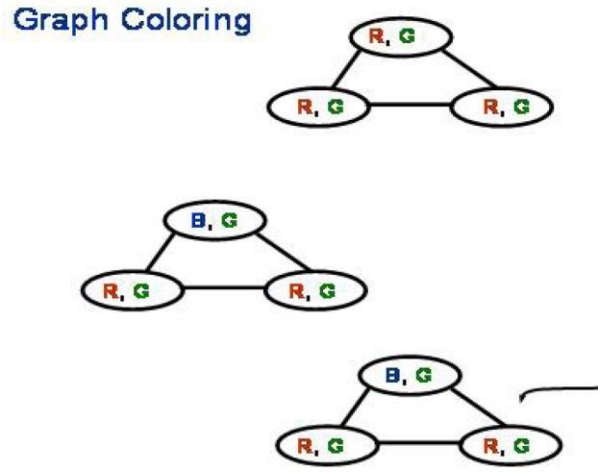


Phù hợp cạnh (arc consistency)

- Các bước thực hiện:
  - Khởi tạo: $queue$ = {(X, Y), (Y, X), (Y, Z), (Z, Y)}
  - Lặp 1: REVISE(X,Y): D_x = (4, 5, 6), D_y = (4, 5, 6, 8, 9), D_z giữ nguyên, $queue$ = {(Y,X), (Y,Z), (Z,Y)}.
  - Lặp 2: REVISE(Y,X): D_y = (4, 5, 6), D_x = (4, 5, 6), D_z giữ nguyên. Do (Z,Y) đã có trong $queue$ nên không được chèn vào $queue$ => $queue$ = {(Y,Z), (Z,Y)}.
  - Lặp 3: REVISE(Y,Z): D_y = (5, 6), D_z giữ nguyên, D_x = (4, 5, 6). (X,Y) được chèn vào $queue$ => $queue$ = {(X,Y), (Z,Y)}.
  - Lặp 4: ...

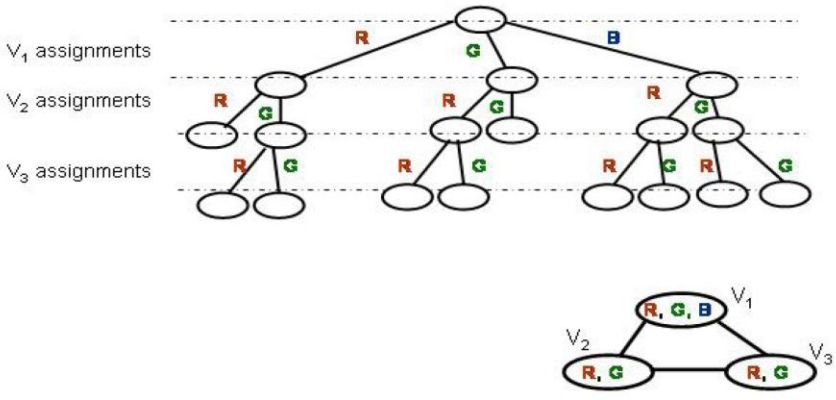
Bài tập

- Sử dụng thuật toán AC-3 tìm nghiệm của các bài toán tô màu bản đồ tương ứng với các đồ thị sau:



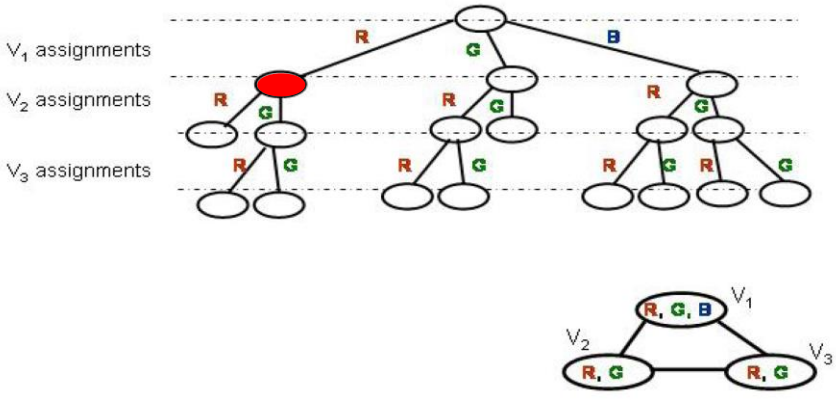
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = {}



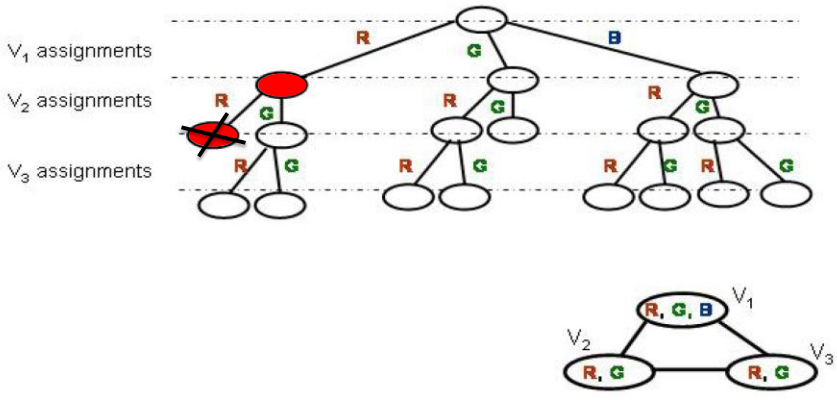
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = $\{V_1 = R\}$



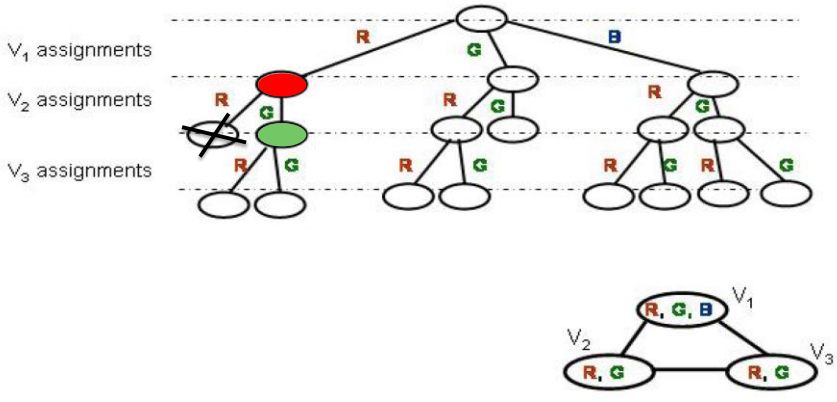
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = {V_1= R, V_2 = R}



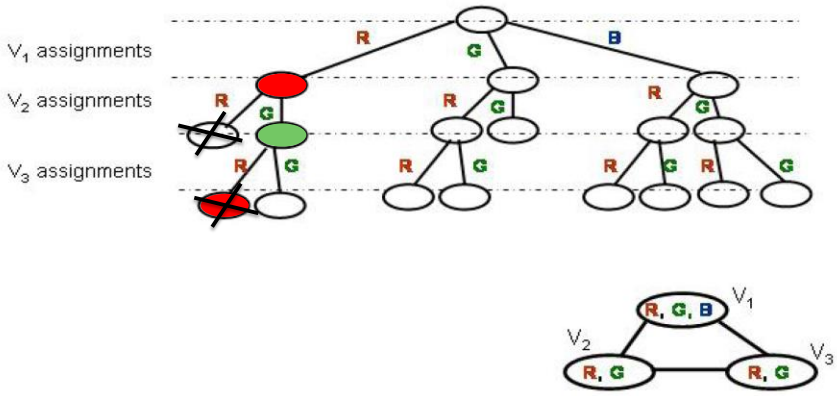
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = {V_1= R, V_2 = G}



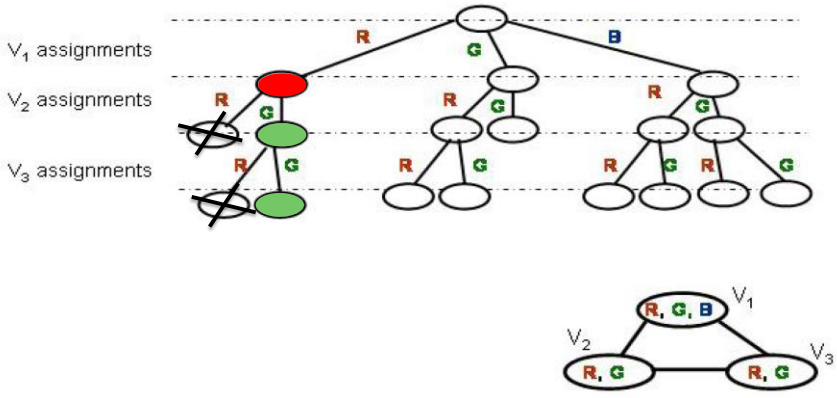
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = {V_1= R, V_2 = G, V_3 = R}



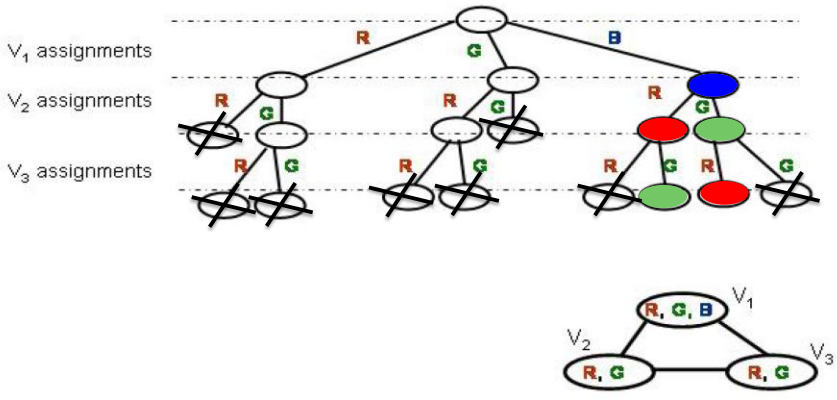
Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Gán giá trị *assignment* = {V_1= R, V_2 = G, V_3 = G}



Tìm kiếm quay lui (backtracking)

- Duyệt từ trái sang phải và quay lui khi phép gán không phù hợp.
- Tiếp tục gán đến khi duyệt hết các giá trị → nghiêm.



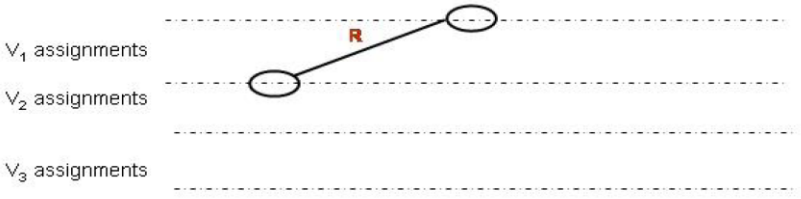
Tìm kiếm quay lui (backtracking)

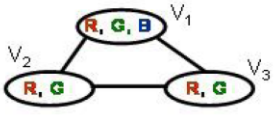
```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return BACKTRACK({ }, csp)

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
```

- Hàm INFERENCE có thể là cung (arc-), đường (path-) hoặc k-phù hợp (k-consistency).

Tìm kiếm quay lui (backtracking)

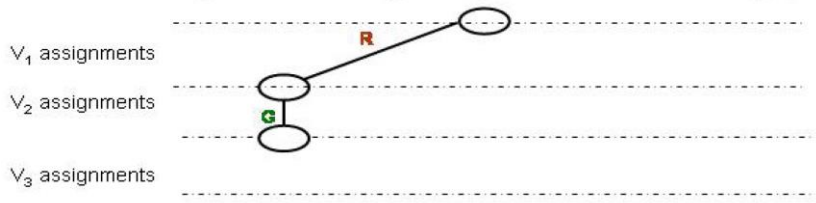
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

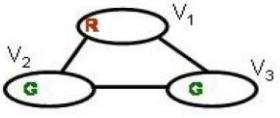




Tìm kiếm quay lui (backtracking)

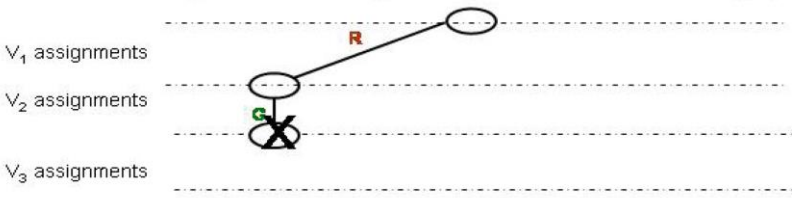
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.



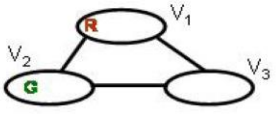


Tìm kiếm quay lui (backtracking)

- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

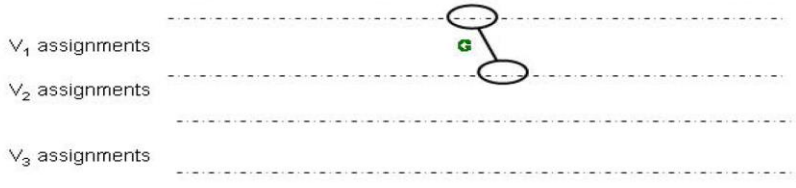


We have a conflict whenever a domain becomes empty.

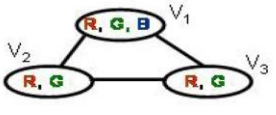


Tìm kiếm quay lui (backtracking)

- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

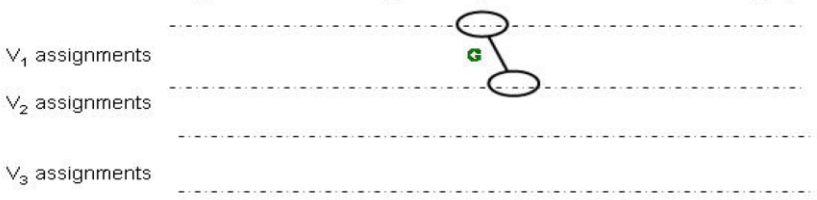


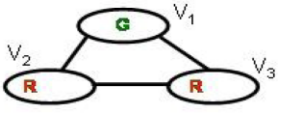
When backing up, need to restore domain values, since deletions were done to reach consistency with tentative assignments considered during search.



Tìm kiếm quay lui (backtracking)

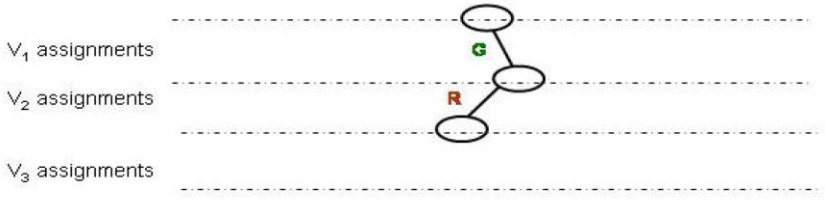
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

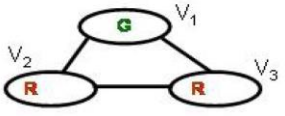




Tìm kiếm quay lui (backtracking)

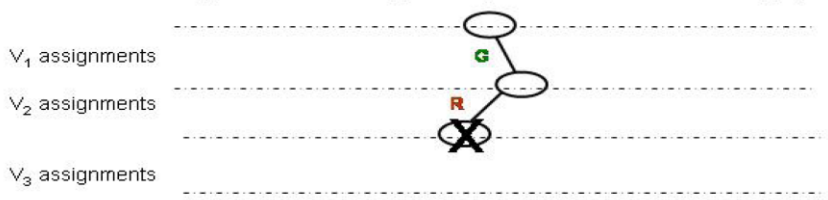
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

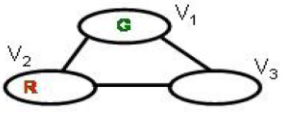




Tìm kiếm quay lui (backtracking)

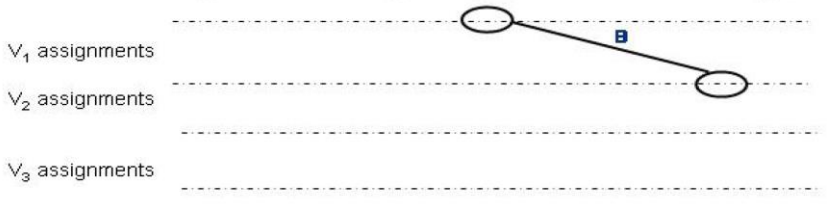
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

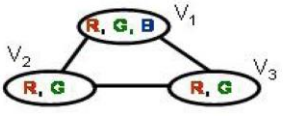




Tìm kiếm quay lui (backtracking)

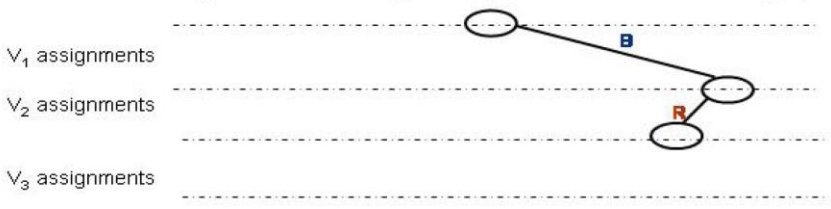
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

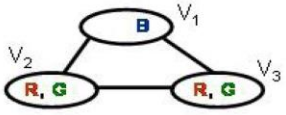




Tìm kiếm quay lui (backtracking)

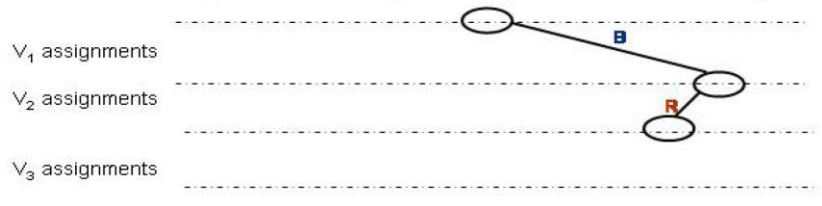
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

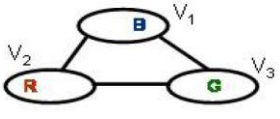




Tìm kiếm quay lui (backtracking)

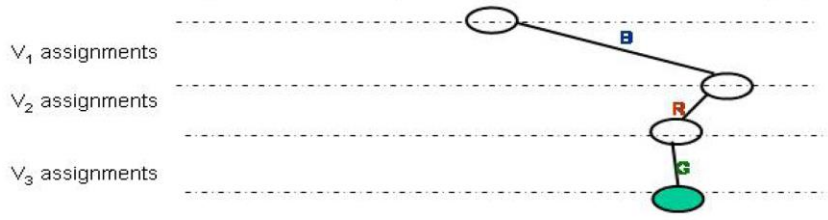
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

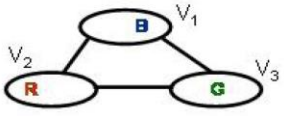




Tìm kiếm quay lui (backtracking)

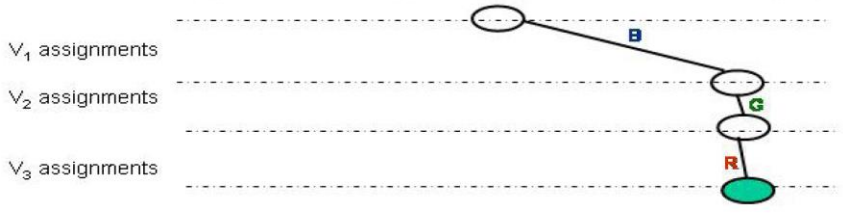
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.

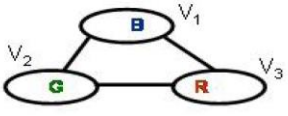




Tìm kiếm quay lui (backtracking)

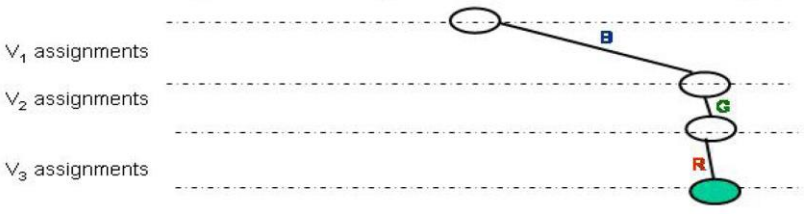
- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.



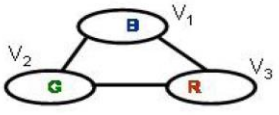


Tìm kiếm quay lui (backtracking)

- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking)
  - Khi gán $V_i = v_k \in D_i$, xóa các giá trị không phù hợp của các biến trong các miền lân cận.



No need to check previous assignments



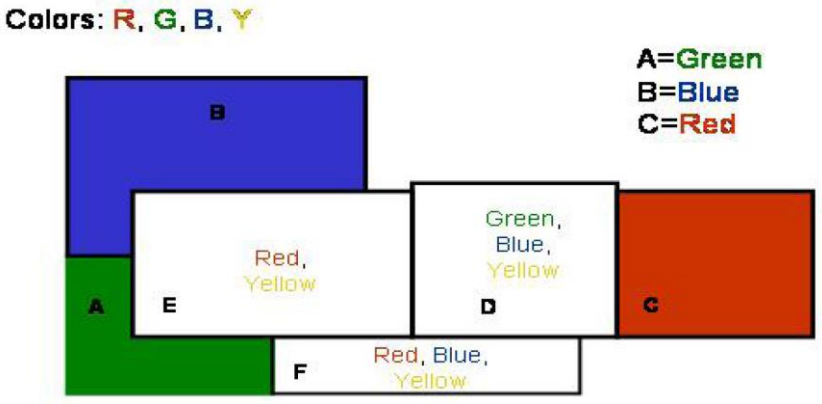
Generally preferable to pure BT

Tìm kiếm quay lui (backtracking)

- Tìm kiếm quay lui (backtracking) kết hợp kiểm tra tiến (forward checking) với thứ tự động (dynamic ordering)
  - Tìm kiếm quay lui sử dụng thứ tự cố định của các biến và các giá trị.
  - Có thể cải tiến hiệu quả tìm kiếm bằng cách chọn thứ tự động trong khi tìm kiếm.
- Khi kiểm tra tiến, chọn biến ràng buộc nhiều nhất.
- Chọn giá trị ràng buộc ít nhất.

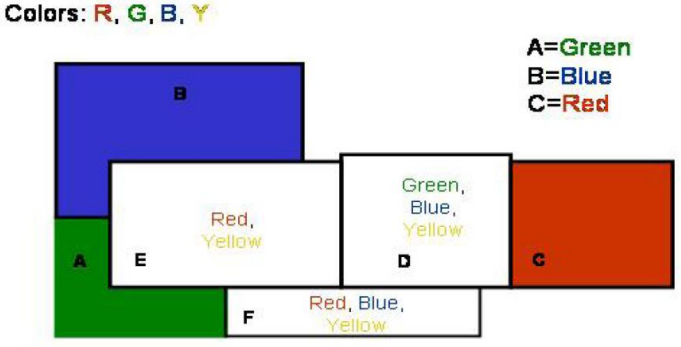
Tìm kiếm quay lui (backtracking)

- Bài toán tô màu bản đồ
  - Tiếp theo nên chọn vùng nào?
  - Màu nào nên chọn cho vùng đã chọn?



Tìm kiếm quay lui (backtracking)

- Bài toán tô màu bản đồ
  - Tiếp theo nên chọn vùng nào? chọn vùng E vì E bị ràng buộc nhiều nhất.
  - Màu nào nên chọn cho vùng đã chọn? chọn màu RED của vùng E vì có ràng buộc ít nhất.



Tìm kiếm cục bộ

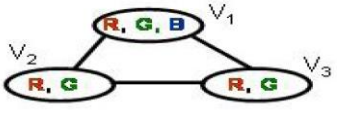
- Chọn giá trị min-conflicts heuristic, tức là chọn giá trị của biến mà có xung đột tối thiểu với các biến khác.

function MIN-CONFLICTS(*csp*, *max_steps*) **returns** a 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 *csp*.VARIABLES
    value ← the value *v* for *var* that minimizes CONFLICTS(*var*, *v*, *current*, *csp*)
    set *var* = *value* in *current*
  return *failure*

Tìm kiếm cục bộ

- Ví dụ bài toán tô màu đồ thị:
  - Khởi tạo: current = {G, R, R}

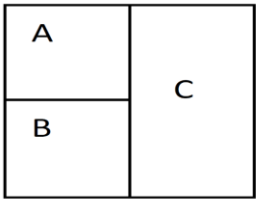


  - Lặp 1:
    - Các biến xung đột là $\{V_2,V_3\}$, chọn ngẫu nhiên *var* = V_3.
    - Giá trị xung đột của các giá trị của V_3: (R,2), (G,2).
    - Chọn *value* = {G}, thay *var* = {G} → *current* = {G,R,G}.
  - Lặp 2:
    - Các biến xung đột là $\{V_1,V_3\}$, chọn ngẫu nhiên *var* = V_1.
    - Giá trị xung đột của các giá trị của V_1: (R,2),(G,2),(B,1).
    - Chọn *value* = {B}, thay *var* = {B} → *current* = {B,R,G}.
  - Lặp 3: *current* = {B,R,G}, thuật toán dừng.

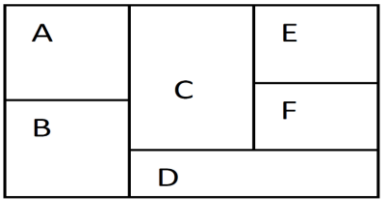
Bài tập

1. Tô màu bản đồ hình (a) với điệu kiện các vùng lân cận nhau phải cómàu khác nhau: D_A = {R,G}, D_B = {R,G}, D_C = {R,G}.

2. Tô màu bản đồ hình (b) với điệu kiện các vùng lân cận nhau phải cómàu khác nhau: D_A = {R,G,B}, D_B = {R,G}, D_C = {G,B}, D_D = {R,B}, D_E = {R,G,B}, D_F = {R,G}.



(a)



(b)