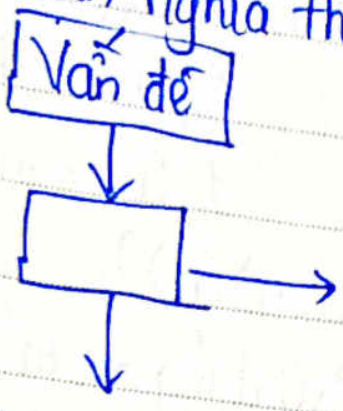
Date: ...../...../.....

CONSTRAINT SATISFACTION PROBLEMS
(Thoả mãn ràng buộc)

1. Định nghĩa thoả mãn ràng buộc



Biến X_1 – Giá trị + ràng buộc
Biến X_2 – Giá trị + ràng buộc
...
Biến X_n – Giá trị + ràng buộc

Phân rã các trạng thái ra thành các thành tố

- State (Trạng thái) là 1 tập các biến và mỗi biến sẽ có cách gán giá trị cho nó.
- Solution là 1 cách nào đó ta gán giá trị cho biến sao cho thoả mọi ràng buộc trên biến đó
- Mỗi CSP sẽ gồm ba thành phần:
  + $X = \{X_1, \ldots, X_n\}$: Tập biến
  + $D = \{D_1, \ldots, D_n\}$: Tập domain tương ứng với mỗi X
    $D_i = \{v_1, \ldots, v_k\}$: Tập các giá trị cho gắp cho biến X_i
  + C là tập hợp các ràng buộc, chỉ định tổ hợp giá trị nào hợp lệ, nào không.

2. Ràng buộc trong CSPs

- Mỗi ràng buộc C_i bao gồm giá trị <scope, rel>
  + scope: Là các biến tham gia vào trong ràng buộc.
  + relation (rel): Sẽ định nghĩa các giá trị mà các biến tham

gia trong scope có thể nhận.

3. Lời giải cho 1 bài toán CSPs

- Là 1 cách gán giá trị vào các biến sao cho các phép gán đều hợp lệ, k° vi phạm ràng buộc nào của bài toán.

+ Phép gán đầy đủ (complete assignment) (các biến chưa được gán đầy đủ gọi là partial assignment)

+ Consistent assignment (k° vi phạm 1 ràng buộc nào)

VD: Map coloring

Biến: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D_i = \{red, green, blue\}$

Ràng buộc: Các vùng liền nhau có màu khác nhau

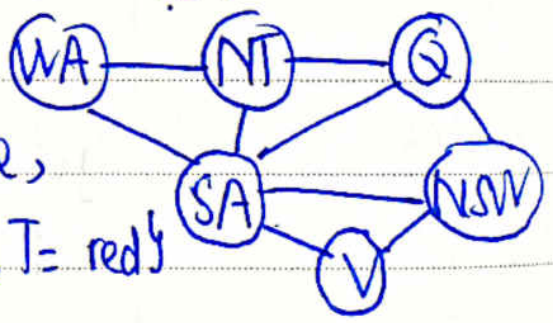
$C = \{SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, SA \neq V,$
$WA \neq NT, NT \neq Q, Q \neq NSW, NSW \neq V\}$

$SA \neq WA$ biểu diễn cho $\langle (SA, WA), SA \neq WA \rangle$

(Có thể dùng liệt kê hoặc biểu diễn toán học)

Có rất nhiều lời giải

$\{WA = blue, NT = green, Q = blue,$
$NSW = green, SA = red, V = blue, T = red\}$



4. Tại sao lại sử dụng CSPs?

- 1 số bài toán khi giải bằng k°gian trạng thái sẽ (T) lâu còn giải bằng CSPs sẽ nhanh hơn nhiều

- Hiểu và phân tích yếu tố trong bài toán tốt hơn, đẩy nhanh quá trình tìm lời giải, tiếp cận tổng quát hơn heuristic.

Date: ...../...../.....

5. Các cấp độ CSPs (domain)
- Tập hợp biến rời rạc, hữu hạn: n biến, d → $O(d^n)$
- Tập hợp biến rời rạc, k0 xác định (domain)
- Domain liên tục

6. Các loại CSPs ràng buộc
- Ràng buộc đơn phân (1 quan 1 biến) (Unary)
- Ràng buộc nhị phân (1 quan 2 biến) (Binary)
- Ràng buộc nhiều biến (1 quan 3 hoặc nhiều biến hơn) (Higher-order)
- Ràng buộc toàn cục Alldiff f

7. Lan truyền ràng buộc
- Kiểm tra các ràng buộc để giảm đi giá trị hợp lệ cho biến nào đó → Khi 1 biến có thay đổi về domain, các biến liên quan nó sẽ giảm domain theo.
- Có thể cắm trước khi tìm kiếm lời giải.
- Khi k0 thể thu hẹp domain, thử giá trị nào đó sau đó loại bỏ giá trị đó ra khỏi domain các biến ≠ nhiệp tục thử (interwine with search). (unary constra
- Đảm bảo tính nhất quán cục bộ (nhất quán cấp độ node; nhất quán cấp độ Arc (binary constraints))

8. The AC-3 Algorithm
- Dpt: $O(cd^3)$: c: số ràng buộc binary (số cạnh)
  n: biến với domainsize d
- Khởi đầu bởi cạnh nào đó

Date: ...../...../.....

- Đưa hết các cạnh vào hàng đợi rồi lần lượt xét từng cạnh.
- Với mỗi cạnh có 2 đầu mút là (X_i, X_j) thì gọi hàm revise. (Hàm revise check xem domain của 2 biến X_i, X_j có gì xung đột hay không)

Revise: Với mỗi $x \in D_i$, k° có giá trị y nào $\in D_j$ làm cho cặp (x,y) thoả mãn ràng buộc thì bỏ x ra khỏi D_i, đặt cờ revise = true (domain X_i, X_j thay đổi).

- Sự thay đổi có thể làm D_i có size = 0 thì trả về false (bài toán fail). K° thì xét hiệp, đặt ~ cạnh có liên quan và X_i vào hàng đợi để kiểm tra.
- Dừng khi kiểm tra k° có đ' k° nhất quán nào nữa.

→ Có thể loại bỏ ~ ràng buộc k° nhất quán (ràng buộc n' phần), chạy lại.

8. Backtracking search

* Liên hệ CSP và Search Problem tìm kiếm theo độ sâu
- n biến, lời giải xuất hiện ở độ sâu n → Dùng giải thuật V
- d là kích thước domain, mỗi nhánh $b = (n-\ell)d$ tại độ sâu ℓ, $n! d^n$ là ~~với mọi~~ (chỉ với) d^n phép gán đầy đủ.

* Backtracking search
- Phép gán biến có tính giao hoán.
- Chỉ cần phép gán với biến đơn ở mỗi node →

là branching factor: $b = d$, d^n leaves.

- Depth-first search: Sd giới hạn cho 1 biến O(1) tgian

Date: ...../...../.....

và quay lui khi biến ko có giá trị hợp lệ.

- Thuật giải:
  + Đầu tiên các phép gán rỗng hết
  + Nếu các phép gán đầy đủ thì trả về các phép gán (if assignment is complete then return assignment)
  + Chọn 1 biến chưa được gán giá trị. Sd heuristic để tìm ra biến nào nên được gán tiếp theo:

var ← SELECT_UNASSIGNED_VARIABLE(csp)

(ORDER-DOMAIN-VALUES(var, assignment, csp): Thứ tự ưu tiên mà các giá trị của nó nên thử)

+ for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do

→ Nếu như giá trị phù hợp với phép gán thì
  . thêm {var = value} vào phép gán.
  . inferences ← INFERENCE(csp, var, value): Thực hiện việc suy dẫn, lan truyền ràng buộc ntn.(
  . Nếu phép suy dẫn ko thất bại thì bổ sung vào phép gán của mình

result ← BACKTRACK(assignment, vào csp): suy dẫn cho biến kế tiếp.

  . Nếu phép suy dẫn thất bại thì bỏ (loại bỏ {var=value} và phép suy dẫn ra khỏi phép gán.

→ Nếu sau vòng lặp ko tìm được giá trị thì trả về failure

Date: ...../...../.....

9. Các heuristic giúp đánh giá, lựa chọn con đường đi tốt hơn.

* Minimum-remaining-values (MRV) heuristic: Chọn biến nào có ít giá trị hợp lệ nhất.
- Chúng tỏ ở mãi thực tế tốt hơn hẳn số lượng lần
* Degree heuristic (DH): Chọn biến (Thường sử dụng khi xét MRV bằng nhau) Chọn biến nào có số ràng buộc lớn nhất với các biến chưa được gán. Cần cập nhật sau mỗi lần.
- Hướng tới mục tiêu đúng đó và khó giải nhất
* Least constraining value (LCV) heuristic: Chọn giá trị sao cho cho biến sao cho mà hàng hướng lại mình còn có nhiều khả năng gán giá trị đó nhất. Sắp xếp ngầm định các giá trị chưa được sd
10. Inference: Forward checking
- Yếu hơn AC-3 ở chỗ chỉ đi tới 1 bước
- Kiểm tra hiện đảm bảo sự phù hợp (consistency) giữa biến đang được xét gán giá trị với các biến ± có liên quan (ràng buộc) mức hiệp với nó.
- Ý tưởng: (các biến liên quan tới nó)
+ Lưu lại ở mỗi bước gán giá trị, theo dõi các giá trị hợp lệ (có thể được gán) đối với các biến chưa được gán giá trị.
+ Loại bỏ (dừng) hướng tìm kiếm hiện tại khi có bất kỳ 1 biến chưa được gán giá trị) nào đó k còn giá trị hợp lệ
Thực hiện backtracking

Date: ..../..../....

- So sánh Forward checking vs. Arc consistency
  + Arc consistency có hướng, có hai hướng.
  + Forward checking chỉ kiểm tra các biến liên kết trực tiếp với nó
  + Arc consistency nặng hơn forward checking