ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TỐI ƯU LẬP KẾ HOẠCH

Quy hoạch ràng buộc

Nội dung

- Tổng quan phương pháp quy hoạch ràng buộc
- Thu hẹp không gian tìm kiếm
- Phân nhánh và tìm kiếm quay lui
- Ví dụ minh họa bài toán N-queens
- Bài toán Sudoku
- Bài toán phân bổ môn học

Tổng quan Quy hoạch ràng buộc

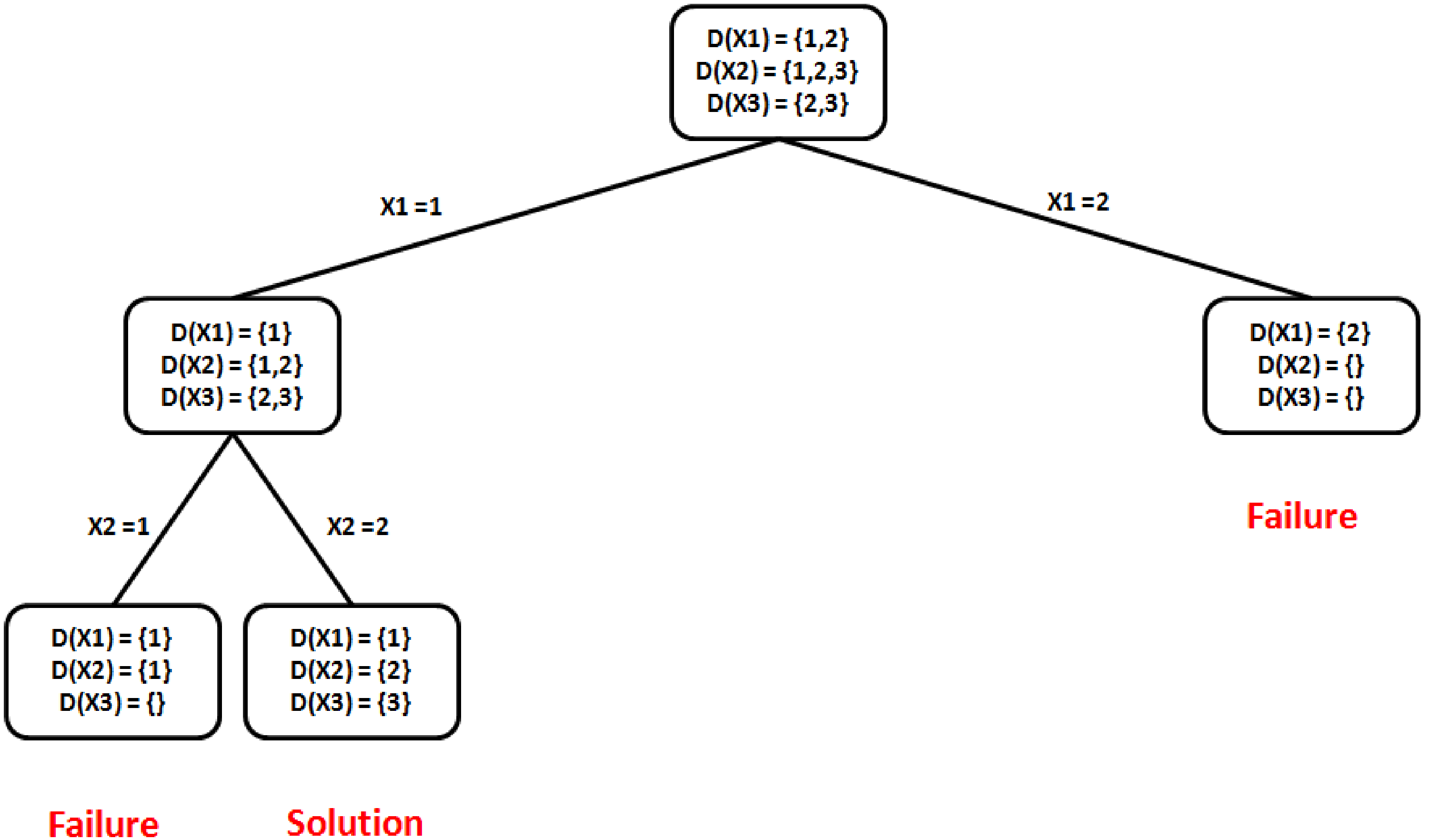
- Quy hoạch ràng buộc: Constraint Programming (hay CP)
- Bài toán tối ưu COP = (X, D, C, f)
  - X = {X_1, …, X_N}: tập các biến
  - D = {D_1, . . ., D_N}: miền giá trị của các biến
  - C = {C_1, . . ., C_K}: tập các ràng buộc
  - f: hàm mục tiêu
- CP: Một phương pháp giải đúng bài toán tối ưu tổ hợp
  - Dùng các ràng buộc để tỉa không gian tìm kiếm: loại bỏ các giá trị thừa ra khỏi miền giá trị của các biến
  - Phân nhánh và tìm kiếm quay lui: Chia không gian tìm kiếm thành các không gian con
    - Liệt kê các giá trị cho biến được lựa chọn
    - Phân hoạch tập giá trị của mỗi biến được lựa chọn thành 2 hoặc nhiều tập con

Tổng quan Quy hoạch ràng buộc

- Biến
  - $X = \{X_0, X_1, X_2, X_3, X_4\}$
- Miền giá trị
  - $X_0, X_1, X_2, X_3, X_4 \in \{1,2,3,4,5\}$
- Ràng buộc
  - C_1: $X_2 + 3 \neq X_1$
  - C_2: $X_3 \leq X_4$
  - C_3: $X_2 + X_3 = X_0 + 1$
  - C_4: $X_4 \leq 3$
  - C_5: $X_1 + X_4 = 7$
  - C_6: $X_2 = 1 \Rightarrow X_4 \neq 2$

Tổng quan Quy hoạch ràng buộc

X = {X1,X2,X3}
D(X1) = {1,2,3,4}, D(X2) = {1,2,3}, D(X3) = {1,2,3}
C1: X1 ≤ X2
C2: X3 = X1+X2
C3: X2+X3 = 5



Thu hẹp không gian tìm kiếm

- Định nghĩa: Bài toán CSP = (X,D,C), trong đó:
  - $X = \{X_1,...,X_N\}$ – tập các biến
  - D = $\{D(X_1),\ldots, D(X_N)\}$ – tập miền giá trị của các biến
  - $C = \{C_1,..,C_K\}$ – tập các ràng buộc
  - Ký hiệu $X(c)$ – tập các biến tham gia vào ràng buộc c

Thu hẹp không gian tìm kiếm

- Domain consistency (DC)
  - Cho bài toán thỏa mãn ràng buộc CSP = (X,D,C), một ràng buộc $c \in C$ được gọi là domain consistent nếu với mỗi biến $X_i \in X(c)$, mỗi giá trị $v \in D(X_i)$, tồn tại bộ giá trị cho các biến $\in X(c) \setminus \{X_i\}$ sao cho ràng buộc c được thỏa mãn
  - Bài toán CSP được gọi là domain consistent nếu c là domain consistent với mọi ràng buộc $c \in C$
- Thuật toán DC là thuật toán nhằm loại bỏ các giá trị dư thừa khỏi miền giá trị của các biến để đưa bài toán CSP ban đầu về bài toán CSP domain consistent tương đương

Thu hẹp không gian tìm kiếm

- Ví dụ: CSP = (X,D,C) trong đó:
  - $X = \{X_1, X_2, X_3, X_4\}$
  - $D(X_1) = \{1,2,3,4\}$, $D(X_2) =\{1,2,3,4,5,6,7\}$, $D(X_3) = \{2,3,4,5\}$, $D(X_4) = \{1,2,3,4,5,6\}$
  - $C = \{c_1,c_2,c_3\}$ với
    - $c_1 \equiv X_1 + X_2 \geq 5$
    - $c_2 \equiv X_1 + X_3 \geq X_4$
    - $c_3 \equiv X_1 + 3 \geq X_3$

→ CSP này là domain consistent

  - Khi phân nhánh, xét $X_1 = 1$, thuật toán DC sẽ đưa CSP đã cho về CSP^1 tương đương các là domain consistent với miền giá trị được thu hẹp như sau : $D^1(X_1) = \{1\}$, $D^1(X_2) = \{4,5,6,7\}$, $D^1(X_3) = \{2,3,4\}$, $D^1(X_4) = \{1,2,3,4,5\}$

Thu hẹp không gian tìm kiếm

- Một bài toán CSP là domain consistent chưa đảm bảo luôn có lời giải,
- Ví dụ xét CSP sau:
  - $X = \{X_1, X_2, X_3\}$
  - $D(X_1) = D(X_2) = D(X_3) = \{0,1\}$
  - $c_1 \equiv X_1 \neq X_2$, $c_2 \equiv X_1 \neq X_3$, $c_3 \equiv X_2 \neq X_3$

  → Rõ ràng CSP này là domain consistent nhưng lại không có lời giải chấp nhận được (lời giải thỏa mãn ràng buộc)

Thu hẹp không gian tìm kiếm

```
Algorithm AC3(X,D,C){
  Q = {(x,c) | c ∈ C ∧ x ∈X(c)};
  while(Q not empty){
    select and remove (x,c) from Q;
    if ReviseAC3(x,c) then{
      if D(x) = {} then
        return false;
      else
        Q = Q ∪{(x',c') | c'∈C\{c} ∧ x,x' ∈X(c') ∧ x≠ x'}
    }
  }
  return true;
}
```

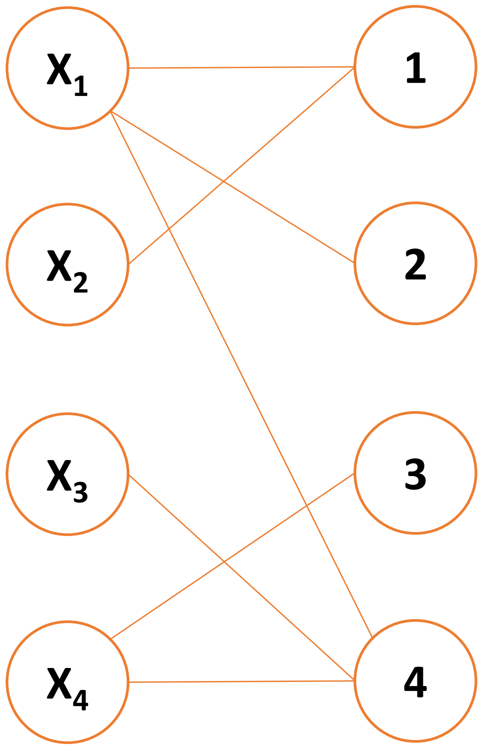
```
Algorithm ReviseAC3(x,c){
  CHANGE = false;
  for v ∈ D(x) do{
    if there does not exists other values
      of X(c) \ {x} such that c
        is satisfied then{
      remove v from D(x);
      CHANGE = true;
    }
  }
  return CHANGE;
}
```

Thu hẹp không gian tìm kiếm

- Một số ràng buộc, ví dụ ràng buộc nhị phân (ràng buộc giữa 2 biến) → thuật toán DC hiệu quả
- Ràng buộc AllDifferent(X_1,X_2,…,X_N), thuật toán DC dựa trên thuật toán hiệu quả cặp ghép cực đại (Max-Matching)
  - Tập đỉnh bên phải là các biến, tập đỉnh bên trái là các giá trị
  - Với mỗi cạnh (X_i, v), (với $v \in D(X_i)$), nếu không tồn tại phương án cặp ghép kích thước N trong đó (X_i, v) là một thành phần của phương án thì có thể loại bỏ v khỏi $D(X_i)$

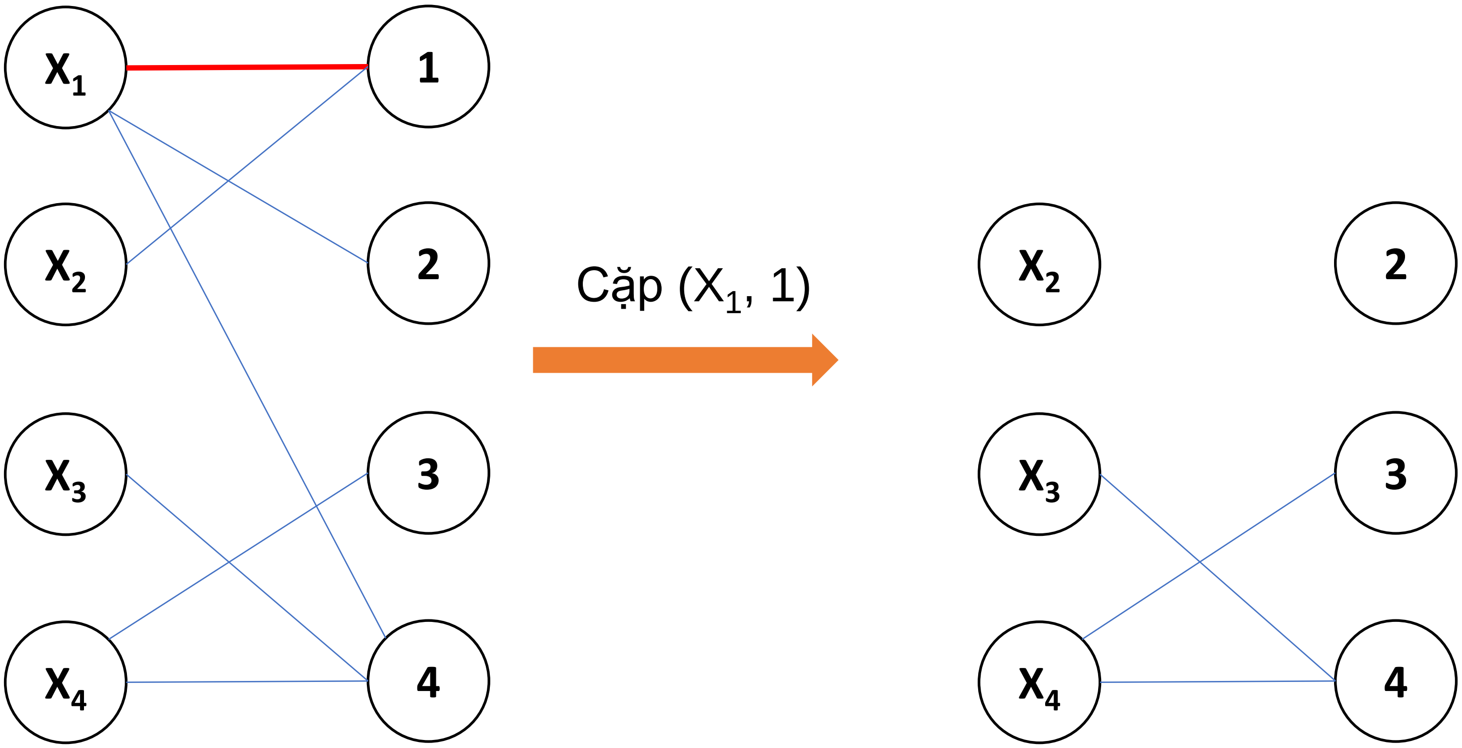
Ràng buộc AllDifferent

- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1) = \{1,2,4\}$, $D(X_2) = \{1\}$, $D(X_3) = \{4\}$, $D(X_4) = \{3,4\}$



Ràng buộc AllDifferent

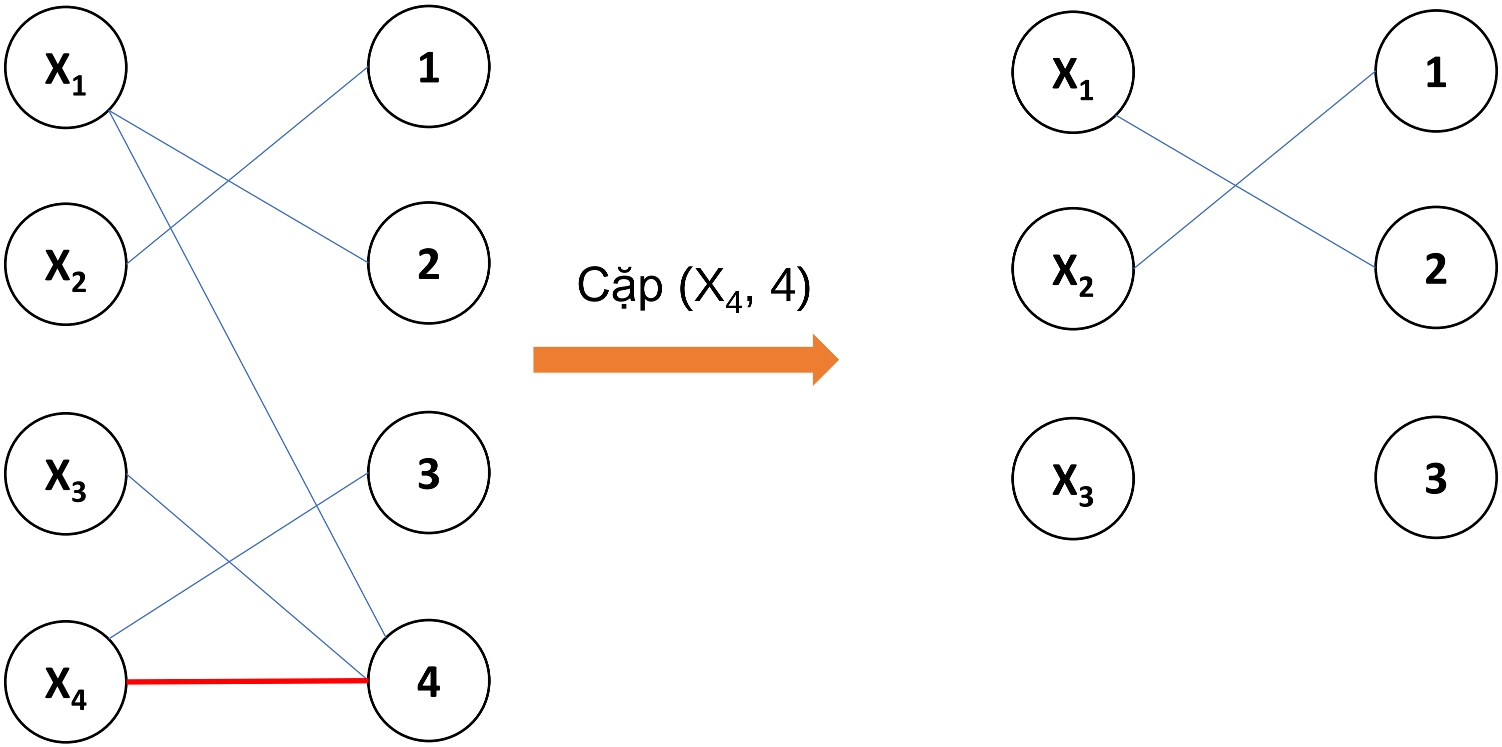
- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1) = \{1,2,4\}$, $D(X_2) = \{1\}$, $D(X_3) = \{4\}$, $D(X_4) = \{3,4\}$



Không tồn tại cặp ghép kích thước 3 → loại bỏ được 1 khởi miền giá trị của X_1

Ràng buộc AllDifferent

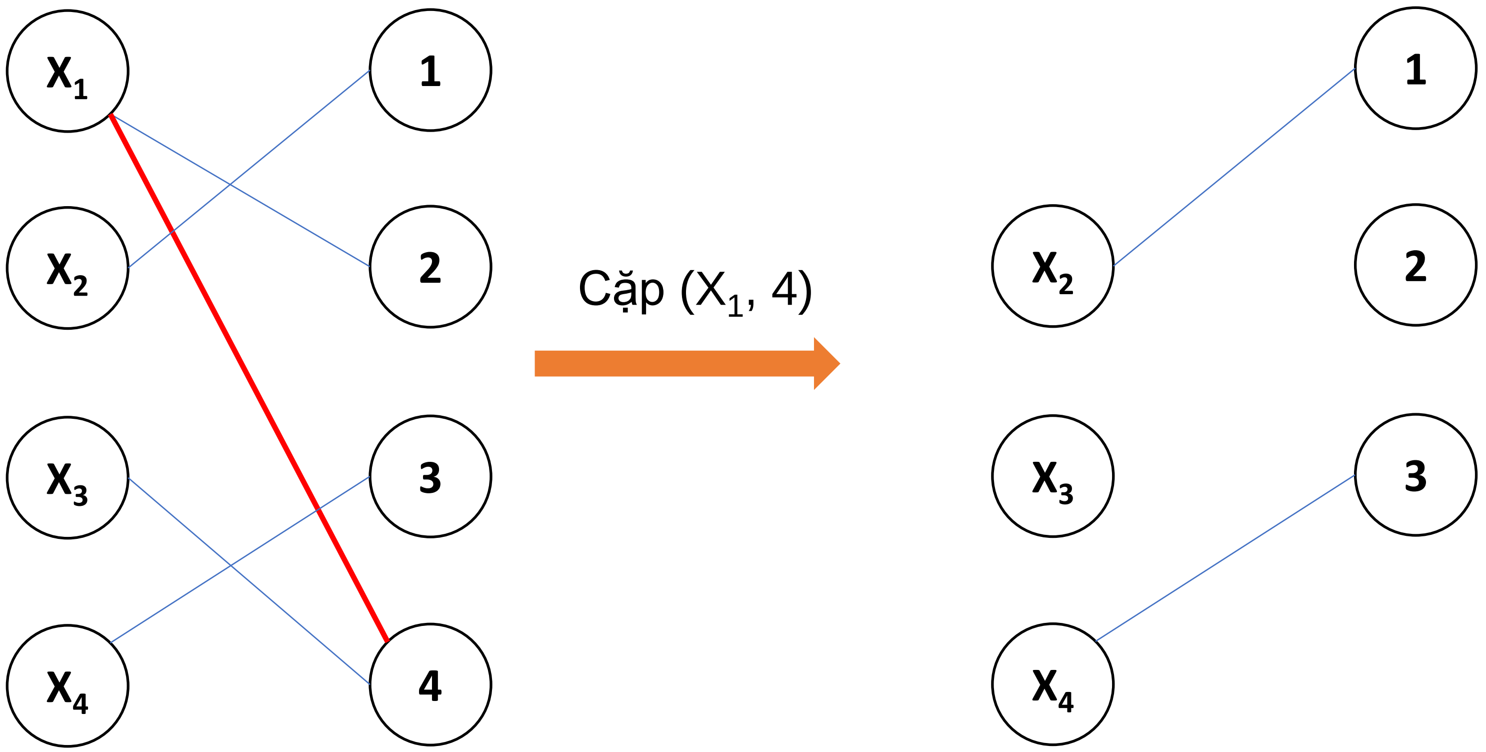
- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1) = \{2,4\}$, $D(X_2) = \{1\}$, $D(X_3) = \{4\}$, $D(X_4) = \{3,4\}$



Không tồn tại cặp ghép kích thước 3 → loại bỏ được 4 khỏi miền giá trị của X_4

Ràng buộc AllDifferent

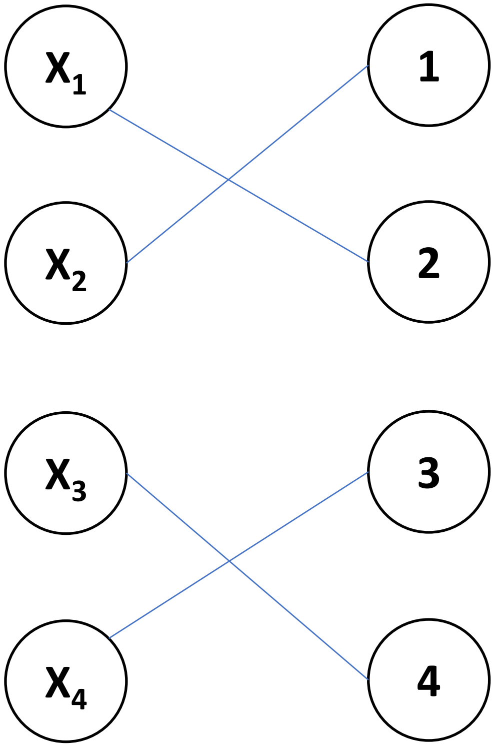
- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1) = \{2,4\}$, $D(X_2) = \{1\}$, $D(X_3) = \{4\}$, $D(X_4) = \{3\}$



Không tồn tại cặp ghép kích thước 3 → loại bỏ được 4 khỏi miền giá trị của X_1

Ràng buộc AllDifferent

- $X = \{X_1, X_2, X_3, X_4\}$
- $D(X_1) = \{2\}$, $D(X_2) = \{1\}$, $D(X_3) = \{4\}$, $D(X_4) = \{3\}$



Phương án chấp nhận được

Phân nhánh và tìm kiếm quay lui

- Việc tỉa không gian tìm kiếm (Propagation) không đủ để tìm ra lời giải tối ưu thỏa mãn ràng buộc
- Cần thiết phải kết hợp giữa tỉa không gian tìm kiếm với phân nhánh và tìm kiếm quay lui

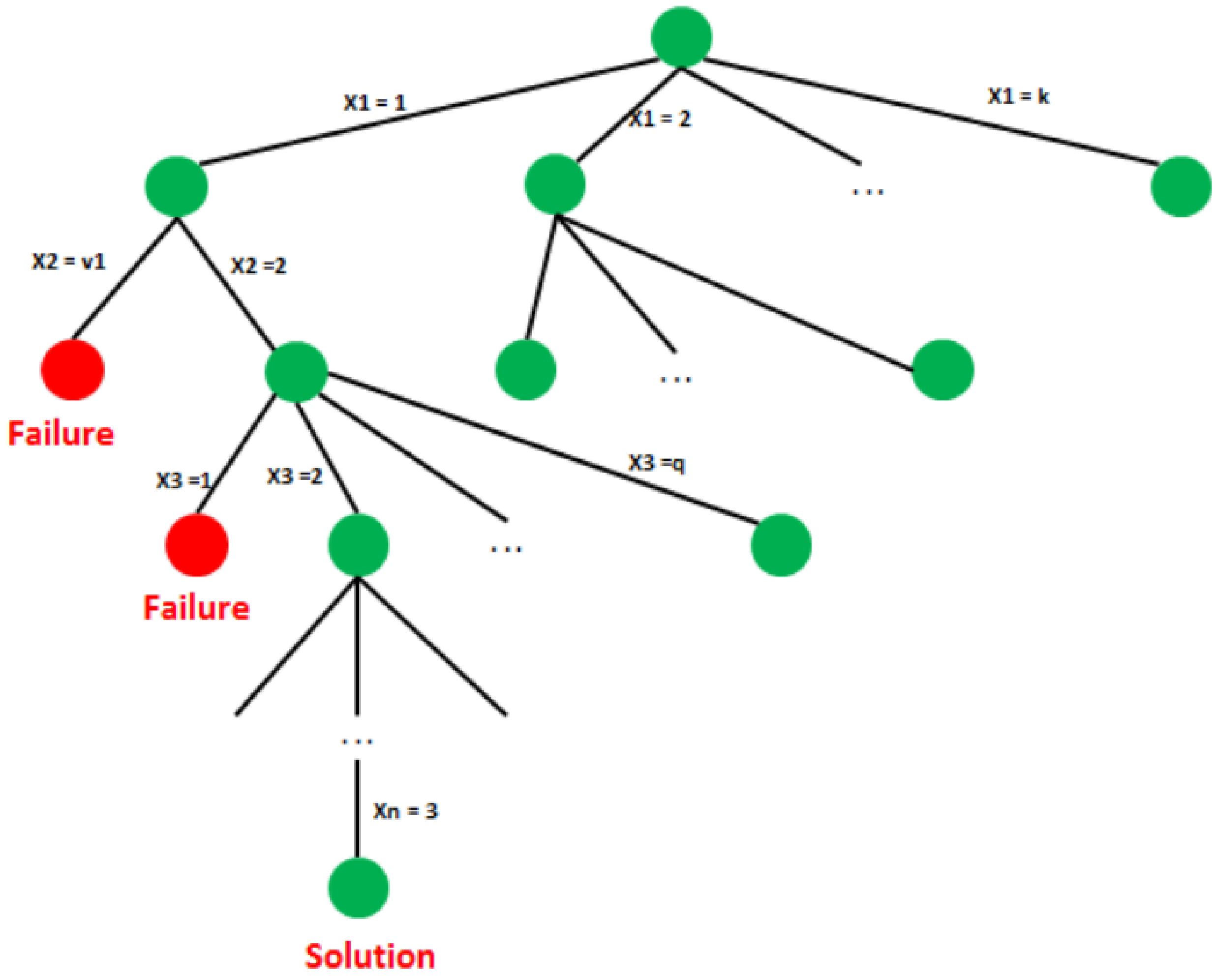
Phân nhánh và tìm kiếm quay lui

- Việc tỉa không gian tìm kiếm (Propagation) không đủ để tìm ra lời giải tối ưu thỏa mãn ràng buộc
- Cần thiết phải kết hợp giữa tỉa không gian tìm kiếm với phân nhánh và tìm kiếm quay lui
  - Phân ra bài toán CSP P_0 ban đầu thành các CSP $P_1,\ldots,P_M$
    - Tập các lời giải của P_0 bằng hợp của tập các lời giải của $P_1,\ldots,P_M$
    - Miền giá trị mỗi biến trong $P_1,\ldots,P_M$ không lớn hơn miền giá trị P_0

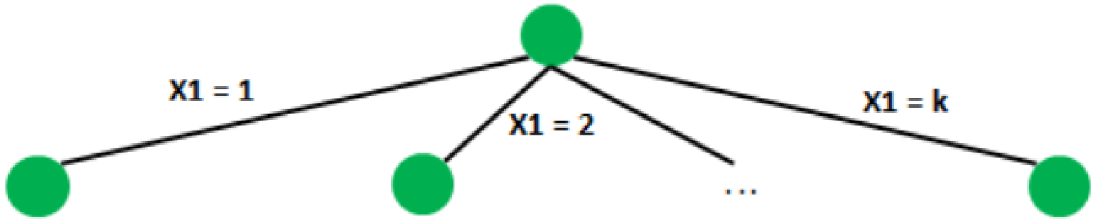
Phân nhánh và tìm kiếm quay lui

- Việc tỉa không gian tìm kiếm (Propagation) không đủ để tìm ra lời giải tối ưu thỏa mãn ràng buộc
- Cần thiết phải kết hợp giữa tỉa không gian tìm kiếm với phân nhánh và tìm kiếm quay lui
  - Phân ra bài toán CSP P_0 ban đầu thành các CSP $P_1,\ldots,P_M$
    - Tập các lời giải của P_0 bằng hợp của tập các lời giải của $P_1,\ldots,P_M$
    - Miền giá trị mỗi biến trong $P_1,\ldots,P_M$ không lớn hơn miền giá trị P_0
  - Cây tìm kiếm (Search Tree)
    - Nút gốc là CSP P_0 ban đầu
    - Mỗi nút của cây là 1 CSP
    - Nếu $P_1,\ldots,P_M$ là các nút con của P_0 thì tập các lời giải của P_0 sẽ bằng với hợp của tập các lời giải của $P_1,\ldots,P_M$
    - Nút lá
      - Một lời giải thỏa mãn ràng buộc
      - Failure (tồn tại biến của miền giá trị rỗng)

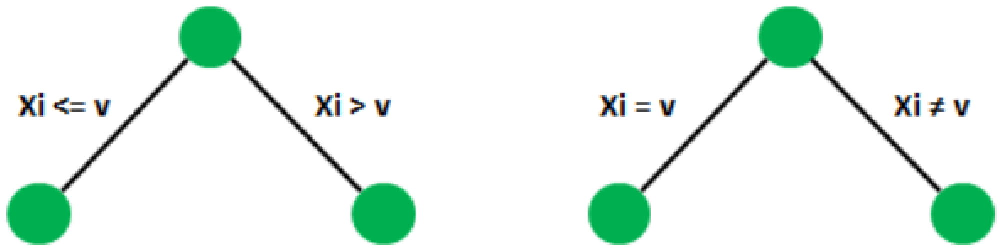
Phân nhánh và tìm kiếm quay lui



Phân nhánh và tìm kiếm quay lui



Enumeration



Splitting a domain

Phân nhánh và tìm kiếm quay lui

- Một số chiến lược tìm kiếm
  - Chọn biến
    - **dom** heuristic: chọn biến có miền giá trị nhỏ nhất
    - **deg** heuristic: chọn biến tham gia vào nhiều ràng buộc nhất
    - **dom+deg** heuristic: áp dụng dom trước, sau đó áp dụng deg khi có nhiều biến có cùng kích thước miền giá trị nhỏ nhất
    - **dom/deg**: chọn biến có tỉ số dom/deg nhỏ nhất (kích thước miền giá trị/số ràng buộc mà biến tham gia vào)
  - Chọn giá trị
    - Chọn giá trị theo thứ tự tăng dần
    - Chọn giá trị theo thứ tự giảm dần
    - Chọn giá trị ở giữa miền giá trị nhất

Một số thư viện

- Gecode: https://www.gecode.org/
- Minizinc: https://www.minizinc.org/
- Google OR-tools: https://developers.google.com/optimization
- CHOCO: https://github.com/chocoteam/choco-solver

Ví dụ

- Biến
  - $X = \{X_0, X_1, X_2, X_3, X_4\}$
- Miền giá trị
  - $X_0, X_1, X_2, X_3, X_4 \in \{1,2,3,4,5\}$
- Ràng buộc
  - C_1: $X_2 + 3 \neq X_1$
  - C_2: $X_3 \leq X_4$
  - C_3: $X_2 + X_3 = X_0 + 1$
  - C_4: $X_4 \leq 3$
  - C_5: $X_1 + X_4 = 7$
  - C_6: $X_2 = 1 \Rightarrow X_4 \neq 2$

Ví dụ

```
Model model = new Model("Example");
IntVar[] X = new IntVar[5];
for(int i = 0; i < 5; i++)
   X[i] = model.intVar("X[" + i + "]",1,5);

model.arithm(model.intOffsetView(X[2],3),"!=",X[1]).post();
model.arithm(X[3], "<=", X[4]).post();
model.arithm(model.intOffsetView(X[0], 1),"=", X[2],"+",X[3]).post();
model.arithm(X[4], "<=", 3).post();
model.arithm(X[1],"+",X[4],"=",7).post();
model.ifThen(model.arithm(X[2], "=", 1), model.arithm(X[4], "!=", 2));

model.getSolver().solve();
for(int i = 0; i < 5; i++)
   System.out.println(X[i]);
```

Bài toán N-queen

```
Model model = new Model("Queen");
IntVar[] x = new IntVar[n];
IntVar[] d1 = new IntVar[n];
IntVar[] d2 = new IntVar[n];
for(int i = 0; i < n; i++){
  x[i] = model.intVar("X" + i,1,n);
  d1[i] = model.intOffsetView(x[i],i);
  d2[i] = model.intOffsetView(x[i], -i);
}

model.allDifferent(x).post();
model.allDifferent(d1).post();
model.allDifferent(d2).post();

model.getSolver().solve();

for(int i = 0; i < n; i++)
  System.out.println("x[" + i + "] = " + x[i].getValue());
```

Bài toán Sudoku

```
Model model = new Model("Sudoku");
IntVar[][] x = new IntVar[9][9];
for (int i = 0; i < 9; i++)
  for (int j = 0; j < 9; j++)
    x[i][j] = model.intVar("x[" + i + "," + j + "]", 1, 9);

for (int i = 0; i < 9; i++) {
  for (int j1 = 0; j1 < 9; j1++){
    for (int j2 = j1 + 1; j2 < 9; j2++) {
      model.arithm(x[i][j1], "!=", x[i][j2]).post();
      model.arithm(x[j1][i], "!=", x[j2][i]).post();
    }
  }
}
```

Bài toán Sudoku

```
for (int I = 0; I < 3; I++)
  for (int J = 0; J < 3; J++)
    for (int i1 = 0; i1 < 3; i1++)
      for (int j1 = 0; j1 < 3; j1++)
        for (int i2 = 0; i2 < 3; i2++)
          for (int j2 = 0; j2 < 3; j2++)
            if (i1 < i2 || i1 == i2 && j1 < j2)
              model.arithm(x[3 * I + i1][3 * J + j1],
                 "!=", x[3 * I + i2][3 * J + j2]).post();
model.getSolver().solve();
for(int i = 0; i < 9; i++){
  for(int j = 0; j < 9; j++)
    System.out.print(x[i][j].getValue() + " ");
  System.out.println();
}
```

Bài toán Phân bổ môn học

- Có *N* môn học 1, 2, …, *N* cần được phân bổ vào *P* học kỳ 1, 2, …, *P*
- Mỗi môn học *i* có số tín chỉ là *credit*(*i*)
- *L* = {(*i*, *j*)}: tập các cặp môn học (*i*, *j*) trong điều kiện tiên quyết (môn *i* phải được xếp và học kỳ trước học kỳ của môn *j*)
- Cho trước các hằng số α, β, λ, γ. Hãy tìm cách xếp *N* môn học vào *P* học kỳ sao cho
    - Tổng số môn học trong mỗi học kỳ phải lớn hơn hoặc bằng α và nhỏ hơn hoặc bằng β
    - Tổng số tín chỉ các môn học trong mỗi học kỳ phải lớn hơn hoặc bằng λ và nhỏ hơn hoặc bằng γ

Bài toán Phân bổ môn học

Môn	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Số tín chỉ	2	1	2	1	3	2	1	3	2	3	1	3

2	1
6	9
5	6
5	8
4	11
6	12
2	7
3	10
5	7
8	11
4	12

$3 \leq$ Số môn học mỗi học kỳ ≤ 3

$5 \leq$ Số tín chỉ các môn học mỗi học kỳ ≤ 7

Phương án phân bổ

Học kỳ	**1**	**2**	**3**	**4**
Danh sách môn	2, 5, 3	1, 6,10	4,7,8	9,11,12

Bài toán Phân bổ môn học

- Biến
  - X[p,i] = 1: môn i được phân vào học kỳ p
  - D(X[p,i])= {0,1}
- Ràng buộc
  - X[q,i] = 1 → X[p,j] = 0, (i,j) $\in$ L, 1 ≤ p ≤ q ≤ P
  - $\sum_{p=1}^{P} X[p,i] = 1$, với mọi i = 1,2,…,N
  - $\alpha \leq \sum_{i=1}^{N} X[p,i] \leq \beta$, với mọi p 1,2,.., P
  - $\lambda \leq \sum_{i=1}^{N} X[p,i] credit(i) \leq \gamma$, với mọi p = 1,2,.., P

Bài toán Phân bổ môn học

```
int N = 9;// number of courses: 0,1,2,...,8
int P = 4;// number of semesters: 0,1,2,3
int[] credits = {3, 2, 2, 1, 3, 3, 1, 2, 2};
int alpha = 2;
int beta = 4;
int lamda = 3;
int gamma = 7;
int[] I = {0,0,1,2,3,4,3};
int[] J = {1,2,3,5,6,7,8};// prerequisites

int[] oneN = {1,1,1,1,1,1,1,1,1};
int[] oneP = {1,1,1,1};

Model model = new Model("BACP");
IntVar[][] x =  new IntVar[P][N];// x[j][i] = 1 indicates that course i is assigned
                                 // to semester j
for(int j = 0; j < P; j++)
  for(int i = 0; i < N; i++)
    x[j][i] = model.intVar("x[" + j + "," + i + "]",0,1);
```

Bài toán Phân bổ môn học

```
for(int j = 0; j < P; j++){
  model.scalar(x[j], credits, ">=", lamda).post();
  model.scalar(x[j], credits, "<=", gamma).post();

  model.scalar(x[j], oneN, ">=", alpha).post();
  model.scalar(x[j], oneN, "<=", beta).post();
}

for(int i = 0; i < N; i++){
  IntVar[] y = new IntVar[P];
  for(int j = 0; j < P ; j++) y[j] = x[j][i];
  model.scalar(y, oneP, "=", 1).post();// each course is assigned to
                                       // exactly one semester
}
```

Bài toán Phân bổ môn học

```
for(int k = 0; k < I.length; k++){
  int i = I[k];  int j = J[k];
  for(int q = 0; q < P; q++)
    for(int p = 0; p <= q; p++)
      model.ifThen(model.arithm(x[q][i], "=", 1),
                                    model.arithm(x[p][j], "=", 0));
}
model.getSolver().solve();


for(int j = 0; j < P; j++){
  System.out.print("semester " + j + ": ");
  for(int i = 0; i < N; i++)if(x[j][i].getValue() == 1){
    System.out.print("[course " + i + ", credit " + credits[i] + "] ");
  System.out.println();
}
```

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

soict.hust.edu.vn/   fb.com/groups/soict