UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013


# Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

## Proiectarea Algoritmilor

## 6. Restricții, rețele de restricții, probleme de prelucrarea restricțiilor

# Bibliografie

http://ktiml.mff.cuni.cz/~bartak/constraints/intro.html

# Restricţii, reţele de restricţii, probleme de prelucrarea restricţiilor

- Definiție: O restricţie c este o relaţie între una sau mai multe variabile $v_1, \ldots, v_m$, (denumite nodurile sau celulele restricţiei). Fiecare variabila $v_i$ poate lua valori într-o anumita mulţime $D_i$, denumită domeniul ei (ce poate fi finit sau nu, numeric sau nu).

- Definiție: Se spune că un tuplu (o atribuire) de valori $(x_1, \ldots, x_m)$ din domeniile corespunzătoare celor m variabile satisface restricţia $c(v_1, ..., v_m)$, dacă $(x_1, \ldots, x_m) \in c(v_1, ..., v_m)$.

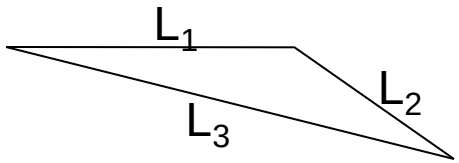
# Exprimarea restricţiilor

- Enumerarea tuplelor restricţiei.
  - **(5,2,3,8,1,6,7,4,9); (6,2,3,8,1,5,7,4,9); etc.**
- Formule matematice, cum ar fi ecuaţiile sau inecuaţiile.
  - $0 < V_{1j} < 10$; $V_{1j} \neq V_{1k}$; $\forall\ j \neq k,\ 0 < j, k < 10$
- Precizarea unei mulţimi de reguli.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | | 8 | 1 | | 7 | 4 | |
| 7 | | | | | 3 | 1 | | |
| | 9 | | | | 2 | 8 | | 5 |
| | | 9 | | 4 | | | 8 | 7 |
| 4 | | | 2 | | 8 | | | 3 |
| 1 | 6 | | | 3 | | 2 | | |
| 3 | | 2 | 7 | | | | 6 | |
| | | 5 | 6 | | | | | 8 |
| | 7 | 6 | | 5 | 1 | | 9 | |

# Tipuri de restricţii

- Unare
  - Specificarea domeniului variabilei.
  - $V_{11} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \setminus \{2, 8, 1, 7, 4, 3, 9\}$
- Binare
  - Între 2 variabile.
  - $V_{1j} \neq V_{1k}\ \forall j \neq k, 0 < j, k < 10$
- N-are
  - Între n variabile.
  - Regula triunghiului: $L_1 + L_2 > L_3$.



# Problemă de satisfacerea restricţiilor

- Definiție: O problemă de satisfacere a restricţiilor (PSR) este un triplet <V,D,C>, format din:
  - o mulţime V formată din n variabile V = {$v_1$, …, $v_n$};
  - mulţimea D a domeniilor de valori corespunzătoare acestor variabile: D = {$D_1$, … $D_n$} ;
  - o mulţime C de restricţii C = {$c_1$, …, $c_p$} între submulţimi ale V ($c_i$ ($v_{i1}$, . . ., $v_{ij}$) ⊆ $D_{i1}$ x $D_{i2}$ x ... x $D_{ij}$).

- Conform Definiției tuplului, o restricţie $c_i$ ($v_{i1}$, ..., $v_{ij}$), este o submulţime a produsului cartezian $D_{i1}$ x $D_{i2}$ x ... x $D_{ij}$, constând din toate tuplele de valori considerate că satisfac restricţia pentru ($v_{i1}$, . . ., $v_{ij}$).

# Problemă de satisfacerea restricţiilor

- Definiție: O soluţie a unei PSR <V,D,C> este un tuplu de valori $<x_1, \ldots, x_n>$ pentru toate variabilele V, din domeniile corespunzătoare D, astfel încât toate restricţiile din C să fie satisfăcute.

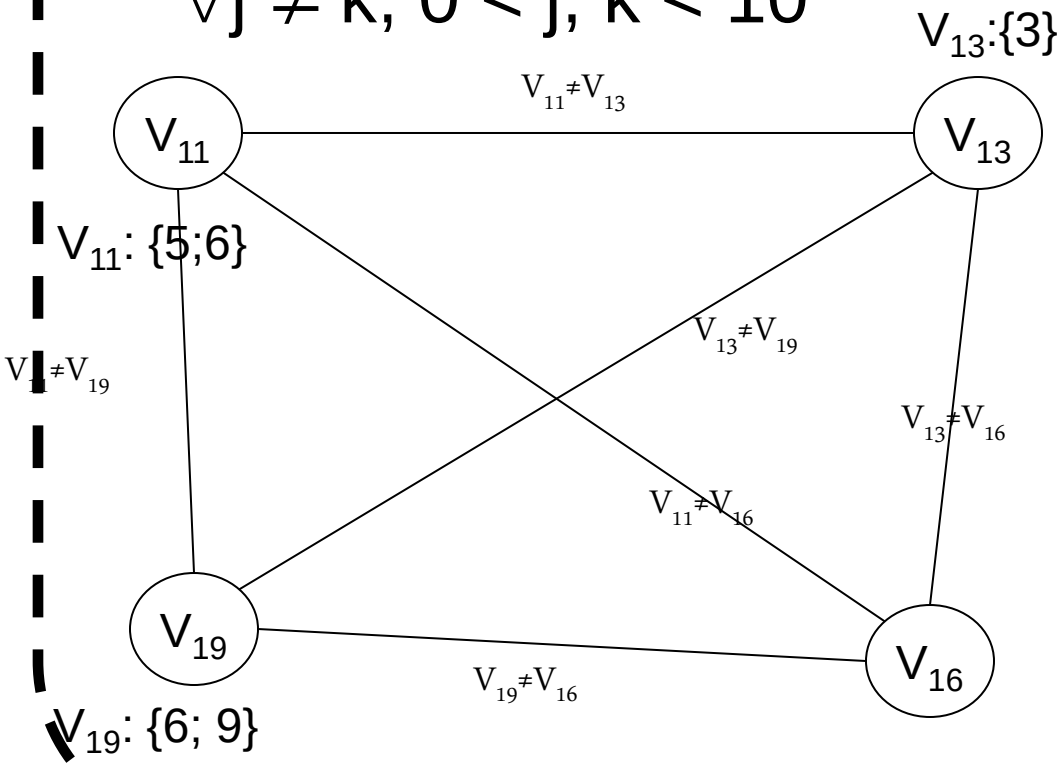
- Definiție: PSR binară este o PSR ce conţine doar restricţii unare şi binare.

# Probleme de satisfacere a restricţiilor - reprezentare

- Reprezentare PSR prin reţele de restricţii.
- Reprezentarea folosită: graf
  - nodurile: variabilele restricţiei
  - arcele: restricţiile problemei
  - valabilă doar pentru PSR binare
- Altă reprezentare posibilă: graf
  - nodurile: restricţiile problemei
  - arcele: variabilele restricţiei
  - Valabilă pentru orice tip de PSR

# Exemplu de reprezentare a PSR

- $0 < V_{1j} < 10$; $V_{1j} \neq V_{1k}$; $\forall j \neq k, 0 < j, k < 10$




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5;6 | 2 | 3 | 8 | 1 | 5;6;9 | 7 | 4 | 6;9 |
| 7 | | | | | 3 | 1 | | |
| | 9 | | | | 2 | 8 | | 5 |
| | | 9 | | 4 | | | 8 | 7 |
| 4 | | | 2 | | 8 | | | 3 |
| 1 | 6 | | | 3 | | 2 | | |
| 3 | | 2 | 7 | | | | 6 | |
| | | 5 | 6 | | | | | 8 |
| | 7 | 6 | | 5 | 1 | | 9 | |

# Algoritmi de rezolvare a PSR

- Caracteristici
  - Rezolvă PSR binare;
  - Variabilele au domenii finite de valori;
  - Prin propagarea restricţiilor se filtrează mulţimile de valori (se elimină elementele din domeniu conform unui criteriu dat);
  - Procesul de propagare se opreşte când:
    - O mulţime de valori este vidă → EȘEC;
    - Nu se mai modifică domeniul vreunei mulţimi.

# Algoritmi de rezolvare a PSR

Notaţii:

- n = număr variabile = număr restricţii unare;
- r = număr restricţii binare;
- G = reţeaua de restricţii cu variabile drept noduri si restricţii drept arce;
- $D_i$ = domeniul variabilei i;
- $Q_i$ = predicat care verifică restricţia unară pe variabila i;
- $P_{ij}$ = predicatul care reprezintă restricţia binară pe variabilele i si j (O muchie între i și j se înlocuiește cu arcele orientate de la i la j și de la j la i);
- a = max $|D_i|$.

# NC-1 (Node Consistency -1)

- Algoritm de consistenţa nodurilor (pentru restricţii unare).

- Procedura NC(i) este:
  - **Pentru fiecare** $x \in D_i$
    - **Dacă** not $Q_i$ (x) // nu este satisfăcută restricția unară
      - **Atunci** șterge x din $D_i$
  - **Sfârșit**.

- Algoritm NC-1 este:
  - **Pentru** i =1 → n **Execută** NC(i) // pentru fiecare var
  - **Sfârșit**.

Complexitate NC-1?

# NC-1: Exemplu (I)

- Elimină din domeniul de valori al fiecărui nod valorile care nu satisfac restricţiile care au ca argument variabila din nodul respectiv.
- În cazul Sudoku variabilele iniţial iau valori între 1-9.
- Algoritmul NC-1 elimină pentru fiecare variabilă acele valori din domeniu care nu sunt consistente cu valorile fixe (celulele deja fixate).

# NC-1: Exemplu (II)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5;6 | 2 | 3; | 8 | 1 | 5;6;9 | 7 | 4 | 6;9; |
| 7 | 4;5;8; | 4;8; | | | 3 | 1 | | |
| 6; | 9 | 1;3;4; | | | 2 | 8 | | 5 |
| | | 9 | | 4 | | | 8 | 7 |
| 4 | | | 2 | | 8 | | | 3 |
| 1 | 6 | | | 3 | | 2 | | |
| 3 | | 2 | 7 | | | | 6 | |
| | | 5 | 6 | | | | | 8 |
| | 7 | 6 | | 5 | 1 | | 9 | |

# Algoritmi de consistenţă a arcelor

- Algoritmii de consistenţă a arcelor înlătură toate inconsistenţele submulţimilor de 2 elemente ale reţelei de restricţii.

- Funcţia REVISE ((i,j)) este:
  - ȘTERS ← fals // nu am modificat domeniul de valori
  - **Pentru fiecare** $x \in D_i$
    - **Dacă** nu există $y \in D_j$ a.i. $P_{ij}(x,y)$ // nu se respectă restricția
      - **Atunci**
        - șterge x din $D_i$;
        - ȘTERS ← adevărat; // am făcut modificări
  - **Întoarce** ȘTERS.

# Exemplu funcţionare REVISE

x

y

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5;~~6~~ | 2 | 3; | 8 | 1 | 5;6;9 | 7 | 4 | 6;9; |
| 7 | 4;5;8; | 4;8; | | | 3 | 1 | | |
| 6; | 9 | 1;~~3~~;4; | | | 2 | 8 | | 5 |
| 5;6;2; | | 9 | | 4 | | | 8 | 7 |
| 4 | | | 2 | | 8 | | | 3 |
| 1 | 6 | | | 3 | | 2 | | |
| 3 | | 2 | 7 | | | | 6 | |
| 9; | | 5 | 6 | | | | | 8 |
| 8; | 7 | 6 | | 5 | 1 | | 9 | |

Complexitate Revise ?

# REVISE - Concluzie

- Funcţia Revise este apelată pentru un arc al grafului de restricţii (binare) si șterge acele valori din domeniul de definiţie al unei variabile pentru care nu este satisfăcută restricţia pentru nici o valoare corespunzătoare celeilalte variabile a restricţiei.

- Complexitate Revise: $O(a^2)$

# AC-1 (Arc Consistency -1)

- Algoritm AC-1 este:
  - NC-1; // reduc domeniul de valori
  - Q ← {(i,j) | (i,j) ∈ arce(G), i ≠ j} // adaug restricțiile
  - **Repetă**
    - SCHIMBAT ← fals // nu am modificat niciun domeniu
    - **Pentru fiecare** (i,j) ∈ Q // pentru fiecare restricție
      - SCHIMBAT ← (REVISE ((i,j)) sau SCHIMBAT)
  - ***Până când*** *non* SCHIMBAT // nu am mai făcut
    // modificări
  - **Sfârșit.**

# AC-1 Caracteristici & Complexitate

- Se aplică algoritmul de consistenţa nodurilor şi apoi se aplică REVISE până nu se mai realizează nici o schimbare.

- Complexitate: $O(na * 2r * a^2)$

La fiecare iteraţie eliminăm o singură valoare (şi avem maxim na valori posibile).

Numărul maxim de apelări al Revise.

Complexitate Revise.

# Exemplu AC-1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5,~~6~~ | 2 | 3; | 8 | 1 | 5;6;9 | 7 | 4 | 6;9; |
| 7 | 4;5;8; | 4;8; | | | 3 | 1 | | |
| 6; | 9 | 1;~~3~~,4; | | | 2 | 8 | | 5 |
| ~~5;6;2;~~ | ~~3,5~~ | 9 | | 4 | | | 8 | 7 |
| 4 | 5 | 7 | 2 | | 8 | | | 3 |
| 1 | 6 | 8 | | 3 | | 2 | | |
| 3 | | 2 | 7 | | | | 6 | |
| 9; | | 5 | 6 | | | | | 8 |
| 8; | 7 | 6 | | 5 | 1 | | 9 | |

# AC-3 (Arc Consistency -3)

- Algoritm AC-3 este:
  - NC-1; // reduc domeniul de valori
  - Q ← {(i,j) | (i,j) ∈ arce(G), i ≠ j} // adaug restricțiile
  - **Cât timp** Q nevid
    - Selectează și șterge un arc (k,m) din Q;
    - **Dacă** REVISE ((k,m)) // am modificat domeniul
      - **Atunci** Q ← Q ∪ {(i,k) | (i,k) ∈ arce(G), i ≠ k, i ≠ m}

        // verific dacă nu se modifică si alte domenii

# AC-3 Caracteristici

- Se elimină pe rând arcele (constrângerile).
- Dacă o constrângere aduce modificări în reţea adăugăm pentru reverificare nodurile care punctează către nodul de plecare al restricţiei verificate.
  - Scopul: Reverificarea nodurilor direct implicate de o constrângere din reţea.
- Avantaj: Se fac mult mai puţine apeluri ale funcţiei REVISE.
- Complexitate: $O(a^3r)$.

# Backtracking + propagarea restricţiilor

- În general, propagarea restricţiilor nu poate rezolva complet problema dată.
- Metoda ajută la limitarea spaţiului de căutare (foarte importantă în condiţiile în care backtracking-ul are complexitate exponenţială!).
- În cazul în care propagarea restricţiilor nu rezolvă problema se foloseşte:
  - Backtracking pentru a genera soluţii parţiale;
  - Propagarea restricţiilor după fiecare pas de backtracking pentru a limita spaţiul de căutare (şi eventual găsi că soluţia nu este validă).