Inteligență Artificială

Universitatea Politehnica Bucuresti
Anul universitar 2021-2022

Adina Magda Florea

Curs 3

Strategii de cautare

- Cautari locale
- Cautari on-line
- **Problema satisfacerii restrictiilor** (CSP)

3. Problema satisfacerii restrictiilor

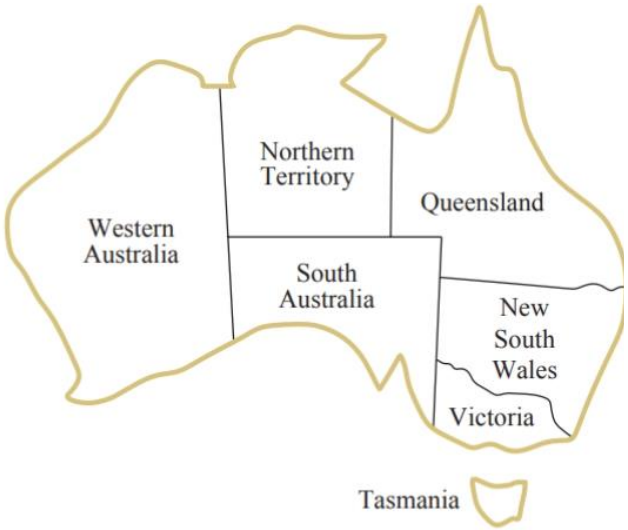
$\{X_1 \dots X_N\}$ $\{(X_1, x_1), \dots., (X_N, x_N)\}$

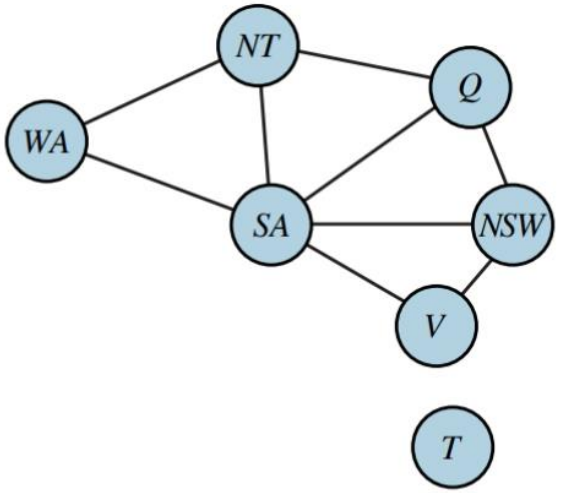
$D = \{D_1 \dots D_N\}$

$R = \{R_1 \dots R_k\}$

- Restrictii explicite sau implicite
- Domeniul de valori
- Restrictii unare, binare, globale

Exemple





	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

$$Alldiff(A1, A2, A3, A4, A5, A6, A7, A8, A9)$$
$$Alldiff(B1, B2, B3, B4, B5, B6, B7, B8, B9)$$
$$\ldots$$
$$Alldiff(A1, B1, C1, D1, E1, F1, G1, H1, I1)$$
$$Alldiff(A2, B2, C2, D2, E2, F2, G2, H2, I2)$$
$$\ldots$$
$$Alldiff(A1, A2, A3, B1, B2, B3, C1, C2, C3)$$
$$Alldiff(A4, A5, A6, B4, B5, B6, C4, C5, C6)$$
$$\ldots$$

Instante ale CSP

- Determinarea unei solutii sau a tuturor solutiilor
- CSP totala
- CSP partiala
- CSP binara – graf de restrictii

CSP – problema de cautare, in NP

- Reducerea timpului (reducerea sp. de cautare)
- Metoda de baza: backtracking

3.1 Imbunatatirea performantelor BKT

Algoritmi care modifica spatiul de cautare prin eliminarea unor portiuni care nu contin solutii

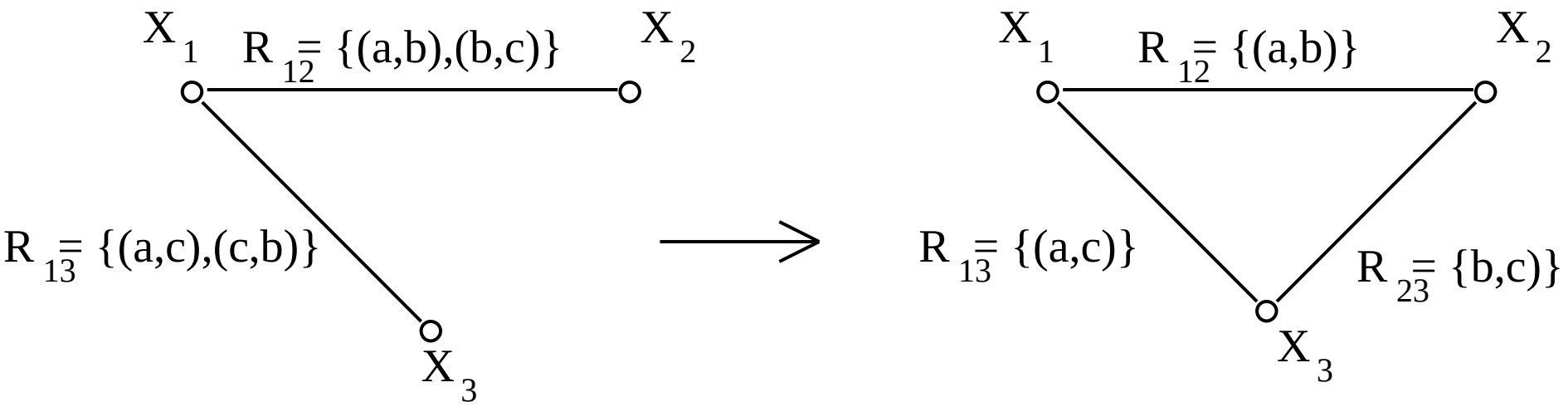
- **Algoritmi de imbunatatire a consistentei reprezentarii (utilizati inainte de inceperea cautarii)**
  - Consistenta locala a arcelor sau a cailor in graful de restrictii
- **Algoritmi de cautare (cauta solutia si elimina portiuni din spatiul de cautare)**
  - Imbunatatesc performantele rezolvarii prin reducerea numarului de teste.

Utilizarea euristicilor in cautare

3.2 Propagarea locala a restrictiilor

- Propagarea restrictiilor

$$D_{X_1} = D_{X_2} = D_{X_3} = \{a, b, c\}$$



Propagarea locala a restrictiilor

$V=\{X,Y,Z\}$ $D_X=D_Y=D_Z=\{1,2,3,4,5,6\}$

$R=\{ X < Y, Z < X-2 \}$

Deoarece $X < Y$ avem $D_X= \{1,2,3,4,5\}$

$D_Y= \{2,3,4,5,6\}$

$D_Z= \{1,2,3,4,5,6\}$

Deoarece $Z < X-2$ avem $D_X= \{4,5\}$

$D_Y= \{2,3,4,5,6\}$

$D_Z= \{1,2\}$

Apoi consideram din nou restrictia $X < Y$

si avem $D_X= \{4,5\}$

$D_Y= \{5,6\}$

$D_Z= \{1,2\}$

Propagarea locala a restrictiilor

- Combinatia de valori ***x*** si ***y*** pentru variabilele X_i si X_j este permisa de restrictia explicita $R_{ij}(x,y)$.

- Un **arc (X_i, X_j)** intr-un graf de restrictii orientat se numeste ***arc-consistent*** daca si numai daca pentru orice valoare $x \in D_i$, domeniul variabilei X_i, exista o valoare $y \in D_j$, domeniul variabilei X_j, astfel incat $R_{ij}(x,y)$.
- **Graf de restrictii orientat *arc-consistent* – orice arc din graf este arc-consistent**

AC-3: Realizarea arc-consistentei pentru un graf de restrictii

/* *Intoarce fals daca inconsistenta, adevarat in caz contrar* */

Creeaza o coada Q ← { $(X_i, X_j) \mid (X_i, X_j) \in$ Multime arce, $i \neq j$}

cat timp Q nu este vida **executa**

Elimina primul arc (X_k, X_m) din Q

daca Verifica(X_k, X_m) **atunci**

daca $|D_{Xk}| = 0$ **atunci intoarce** fals

$Q \leftarrow Q \cup \{ (X_i, X_k) \mid (X_i, X_k) \in$ Multime arce, $i \neq k,m\}$

intoarce adevarat

Verifica (X_k, X_m) /* *intoarce adevarat daca se modifica* D_{Xk} */

modif ← fals

pentru fiecare $x \in D_{Xk}$ **executa**

daca nu exista nici o valoare $y \in D_{Xm}$ astfel incat $R_{km}(x,y)$

atunci elimina x din D_{Xk}

modif ← adevarat

/* Modifica domeniul D_{Xk} prin efect lateral */

intoarce modif

Complexitate

- **N -** numarul de variabile
- **a -** cardinalitatea maxima a domeniilor de valori ale variabilelor
- **e -** numarul de restrictii.
- Algoritmului de realizare a arc-consistentei - *AC-3*: complexitate timp este **$O(e*a^3)$**
- S-a gasit si un algoritm de complexitate timp **$O(e*a^2)$** – *AC-4*

m-Cale-consistență

- O cale de lungime *m* prin nodurile $i_0,\ldots,i_m$ ale unui graf de restrictii orientat se numeste ***m-cale-consistenta*** daca si numai daca pentru orice valoare $x \in D_{i0}$, domeniul variabilei i_0 si o valoare $y \in D_{jm}$, domeniul variabilei i_m, pentru care $R_{i0im}(x,y)$, exista o secventa de valori $z_1 \in D_{i1} \ldots z_{m-1} \in D_{im-1}$ astfel incat $R_{i0i1}(x,z_1), \ldots, R_{im-1im}(z_{m-1},y)$
- **Graf de restrictii orientat *m-cale-consistent* – daca orice cale din graf este cale-consistenta**
- Algoritmul de realizare - *PC-4*: complexitatea timp **$O(N^3*a^3)$**
- **2-cale consistenta**
- A face un graf de restrictii m-cale-consistent are o complexitate exponentiala in m

2-Cale-consistență

$V=\{A,B,C\}$ $D_A=D_B=D_C=\{1,2,3\}$

$R=\{B>1, A<C, A=B, B>C-2\}$

Putem exprima restrictiile sub forma de matrici booleene

B>1	A=B	A<C	B>C-2
000	100	011	110
010	010	001	111
001	001	000	111

$R_{i,j}$ – intre X_i si X_j

$R_{k,k}$ – domeniu D

Consistenta pe calea (i,k,j) poate fi detereminata recursiv utilizand ecuatia

$$R_{i,j} \leftarrow R_{i,j} \wedge (R_{i,k} * R_{k,k} * R_{k,j})$$

De ex $R_{AC} \leftarrow R_{A,C} \wedge (R_{A,B} * R_{B,B} * R_{B,C})$

Cale consistență (ordin 2)

/* *Intari: Multime de variabile V, n=|V|, multime de matrici de restrictii R*
Iesiri: Matrici de restrictii cale-consistente */

$Y^n \leftarrow R$

repeta

| $Y^0 \leftarrow Y^n$

| **pentru** k=1,n **repeta**

| | **pentru** i=1,n **repeta**

| | | **pentru** j=1,n **repeta**

| | | | $Y_{ij}^{k} \leftarrow Y_{i,i}^{k-1} \wedge (Y_{i,k}^{k-1} * Y_{k,k}^{k-1} * Y_{k,j}^{k-1})$

pana $Y^n = Y^0$

intoarce Y^0

sfarsit

Este o extindere a algoritmului Floyd Warshall (all shortest path) in care minimizarea este inlocuita cu conjunctia si aduarea cu multiplicarea matricilor

3.3 Cautare cu BKT

- BKT – algoritmul de baza
- Imbunatatiri
- Realizarea arc-consistentei pe parcursul cautarii
- Backjumping

Backtracking

```
/* Intrari: variabile V, restrictii R
   Iesiri: Atribuiri pt X si adev daca R satisfacute, fals in caz contrar */
/* L- variabile instantiate, U – variabile neinstantiate */
(b,L) ← BKT(V, {}, R)
daca b atunci intoarce L
intoarce fals
sfarsit
BKT(U,L,R)
daca U={} atunci intoarce (adevarat, L)
X ← Selectie(U)
pentru fiecare x ∈ D_X repeta
    b ← Consistent(L∪{X=x},R)
    daca b atunci
        (b,L) ← BKT(U\{X}, L ∪{(X,x)},R)
        daca b atunci intoarce (b,L)
intoarce fals
sfarsit
```

```
Consistent(L,R)
pentru fiecare r ∈R repeta
  daca Variabile(r) ⊆ L)
  atunci
      daca not(Satisfacut(r,L)
      atunci intoarece fals
  intoarce adevarat
```

Realizarea arc-consistentei pe parcursul cautarii

- Forward checking
- **Maintaining Arc-Consistency** (MAC)
- Dupa ce s-a alocat o valoare unei variabile X, algoritmul apeleaza AC-3 dar in loc sa faca o coada cu toate arcele din problema, se introduce in coada numai arcele (X,Y_i) pentru toate variabilele Y_i neinstantiate care sunt legate de X printr-o restrictie
- AC-3 face propagarea restrictiilor si, daca un domeniul este redus la multimea vida, intoarce fals si se intra in backtracking

Backtracking cu MAC

/* *Intrari: variabile V, restrictii R*
Iesiri: Atribuiri pt X si adev daca R satisfacute, fals in caz contrar */
/* *L- variabile instantiate, U – variabile neinstantiate* */

(b,L) ← BKT_MAC(V, {}, R,D)

daca b **atunci intoarce** L

intoarce fals

sfarsit

BKT_MAC(U,L,R,D)

daca U={} **atunci intoarce** (adevarat, L)

X ← Selectie(U)

pentru fiecare x ∈ D_X **repeta**

    (b,D') ← AC-3'(L ∪{(X,x),R,D)

    daca b **atunci** (b,L) ← BKT_MAC(U\{X}, L ∪{(X,x)},R,D')

        daca b **atunci intoarce** (b,L)

intoarce fals

sfarsit

Backjumping

- BKT cu salt – determina nivelul in care apare eventual un conflict

Exemplu

$V = \{A,B,C,D\}$ $D_A=D_B=D_C=D_D=\{1,2,3,4\}$

$R=\{A > D\}$

Backjumping

/* *Intrari: variabile V, restrictii R*
Iesiri: Atribuiri pt X si adev daca R satisfacute, fals in caz contrar */
(n,L) ← Backjump(V, {}, R, 0) /* *n – variabila intreaga* */
daca n=|V|+1 **atunci intoarce** L
intoarce fals
sfarsit

Backjump(U,L,R,p)

/**Intrari: U, L, R, nivel anterior p, Iesiri: L sau nivelul de salt pt variabila conflict**/
daca U={} **atunci intoarce** (|L+1|, L)
X ← Selectie(U) m ← 0 /* *init variabila jump* */
pentru fiecare x ∈ D_X **repeta**
(b,j) ← Consistent(L∪{(X,x,p+1), R, p+1) /* *b var booleana, j var intraga* */
daca b **atunci**
m ← p
(n,L) ← Backjump(U\{X}, L ∪{(X,x,p+1)}, R, p+1) /* *n intreg* */
daca (n ≠ p+1) **atunci intoarce** (n,L)
altfel
m ← max{m,j}
intoarce (m,.)
sfarsit

Consistent(L,R,nivel)

/ intoarce nivelul variabilei conflictuale */*

j ← nivel

b ← fals

pentru fiecare r ∈R **repeta**

  daca Variabile(r) ⊆ L)

  atunci

    daca not(Satisfacut(r,L)

    atunci

      b ← adevarat

      j ← min{j, $\max_k$ (X∈Variabile(r) **si** (X,x,l)∈L **si** k < l)}

daca b

atunci intoarce (fals, j)

intoarce (adevrat,.)

sfarsit

3.4 Euristici

Euristici generale

Ordonarea variabilelor – vezi functie Selectie(U)

- aleator
- Minimum remaining value (MRV) – se incepe cu variabila cea mai restrictionata intai (fail-first) – variabila cu cele mai putine valori legale
- Degree heuristic – selectie variabila care este implicata in cel mai mare numar de restrictii cu variabile neinstantiate

Ordonarea valorilor

- Least-constrained value – selectie valoarea care elimina cele mai putine valori din domeniul variabilelor neinstantiate cu care este legata prin restrictii (fail-last)

O solutie: variabila fail-first, valoare fail last

Toate solutiile: variabila fail-last, valoare fail first

3.5 CSP partiala

- Memoreaza cea mai buna solutie gasita pana la un anumit moment (gen IDA*) – *distanta d* fata de solutia perfecta
- Abandoneaza calea de cautare curenta in momentul in care se constata ca acea cale de cautare nu poate duce la o solutie mai buna
- NI - numarul de inconsistente gasite in "cea mai buna solutie" depistata pana la un moment dat – *limita necesara*

CSP partiala

- S - *limita suficienta* - specifica faptul ca o solutie care violeaza un numar de S restrictii (sau mai putine), este acceptabila.
- PBKT(Cale, Distanta,Variabile,Valori)
  - Semnificatie argumente
  - Rezultat: GATA sau CONTINUA
- variabile globale: CeaMaiBuna, NI, S

Algoritm: **CSP Partiala**
PBKT(Cale, Distanta, Variabile, Valori)

/ intoarce GATA sau CONTINUA */*

1. **daca** Variabile = {}
   atunci
   1.1 CeaMaiBuna ← Cale
   1.2 NI ← Distanta
   1.3 **daca** NI ≤ S **atunci intoarce** GATA
   altfel intoarce CONTINUA
2. **altfel**
   2.1 **daca** Valori = {} **atunci** CONTINUA

/ s-au incercat toate valorile si se revine la var ant. */*

2.2 **altfel**
2.2.1 **daca** Distanta ≥ NI
atunci intoarce CONTINUA

/ revine la var ant pentru gasirea unei solutii mai bune*/*

2.2.2 **altfel**

i. Var ← first(Variabile)

ii. Val ← first(Valori)

iii. DistNoua ← Distanta

iv. Cale1 ← Cale

v. **cat timp** Cale1 <> {} **si** DistNoua < NI **executa**

- (VarC, ValC) ← first(Cale1)
- **daca** Rel(Var,Val,VarC,ValC) = fals
- **atunci** DistNoua ← DistNoua + 1
- Cale1 ← Rest(Cale1)

vi. **daca** DistNoua < NI **si**
PBKT(Cale+(Val,Var),DistNoua,Rest(Variabile),ValoriNoi)
= GATA

/* ValoriNoi - domeniul de valori asociat primei variabile din Rest(Variabile) */

atunci intoarce GATA

altfel intoarce
PBKT(Cale,Distanta,Variabile,Rest(Valori)

sfarsit