Inteligență artificială
Laborator 4

Probleme de satisfacere a constrângerilor

O **problemă de satisfacere a constrângerilor** (*Constraint Satisfaction Problem*):
- o mulțime de variabile $X=\{X_1,..., X_n\}$
- fiecare variabilă X_i poate lua valori dintr-un domeniu D_i
- o mulțime de constrângeri $C=\{C_1 ,..., C_t\}$ care specifică combinațiile permise de valori.
O **soluție** pentru o astfel de problemă este o asignare de valori variabilelor a.î. toate constrângerile să fie satisfăcute.

În definiție nu se impune nici o condiție asupra tipului variabilelor. Acestea pot fi întregi, logice, mulțimi, sau de orice alt tip. Nici modul de definire a constrângerilor nu este limitat. Constrângerile pot fi date atât explicit, prin specificarea tuplelor de valori permise, cât şi implicit, prin relații (ex: $X_i > 2$).

Graful restricțiilor: nodurile reprezintă variabilele, muchiile reprezintă restricțiile între variabile.

Exemplul 1: Problema colorării unei hărți
Considerăm o hartă cu *n* țări. Fiecare regiune/țară poate fi colorată cu o culoare dintr-o mulțime de culori asociate. Să se coloreze harta a.î. regiunile/țările vecine să fie colorate diferit.

Modelare: asignăm fiecărei regiuni/țări de pe hartă o variabilă; domeniul fiecărei variabile este o mulțime de culori specificată. Restricțiile sunt de forma: $X_i \neq X_j$ (două țări vecine au culori diferite). Obs: între două țări vecine în graful constrâns vom adăuga o muchie.

Exemplul 2: problema reginelor
Dată fiind o tablă de şah *n*x*n* dorim să plasăm *n* regine pe tablă a.î. nici o regină să nu fie atacată de nici o altă regină.

O posibilă formulare a problemei reginelor ca o problemă CSP: pentru fiecare coloană a tablei de şah asociem o variabilă X_i, iar domeniul variabilei sunt liniile, adică $D_i = \{1,.., n\}$. Constrângerile sunt asociate fiecărei perechi de coloane şi precizează că două regine nu se pot afla pe aceeaşi linie sau pe aceeaşi diagonală; se exprimă prin relațiile:
$X_i \neq X_j$
$|X_i – X_j| \neq |i-j|$, pentru orice i, j din intervalul 1, ..,n.

Abordări
- Algoritmul **Backtracking**: menține o soluție parțială (o mulțime de variabile instanțiate corect) pe care o extinde pas cu pas. Inițial, mulțimea este vidă. La fiecare pas se selectează următoarea variabilă din ordonare şi se încearcă asignarea variabilei cu o valoare consistentă cu instanțierea parțială. Dacă este găsită o astfel de valoare, algoritmul continuă procedeul cu următoarea variabilă din ordonare. În caz contrar, ne întoarcem la variabila anterioară şi îi asignăm o altă valoare consistentă.

- **Propagarea constrangerilor**
Forward-checking: verifică ca fiecare valoare să fie compatibilă cu cel puțin o valoare din domeniul fiecărei variabile viitoare. Se instanțiază variabila cu o valoare şi apoi se elimină valori

din domeniul variabilelor viitoare care sunt în conflict cu instanțierea curentă. Dacă domeniul unei variabile viitoare devine vid, algoritmul consideră următoarea valoare posibilă pentru variabila curentă.

Ordonarea variabilelor
MRV (*Minimum remaining values*): alege variabila cu cele mai puține valori rămase în domeniu

Temă
Considerăm o tablă de dimensiune nxn și n regine ce trebuiesc așezate pe tablă. Unele locații sunt blocate și nu putem așeza o regină. Problema constă în plasarea reginelor pe tablă a.i. acestea nu se atacă. 2 regine se atacă atunci când se află pe aceeași linie, coloană sau diagonală.
Aplicați algoritmul Forward checking împreună cu o metodă de ordonare a variabilelor pentru a determina soluțiile unei instanțe date.
(Instanțe de test se găsesc la adresa: https://www.csplib.org/Problems/prob080/data/)

Exemple de instanțe și soluții:
- n=4, block(1,1), block(2,2), block(4,3)
Soluție: queen(1,3), queen(2,1), queen(3,4), queen(4,2)

- n=5, block(1,1), block(2,2), block(3,3), block(4,5), block(5,5)
Soluție: queen(1,5), queen(2,3), queen(3,1), queen(4,4), queen(5,2).

Considerăm instanța prezentată în primul exemplu.
Aplicarea algoritmului *Forward-checking + MRV:*

	X1	X2	X3	X4
Initial	2,3,4	1,3,4	1,2,3	1,2,3,4
Dupa X1=2	2	4	1,3	1,3,4
Dupa X2=4	2	4	1	1,3
Dupa X3=1	2	4	1	3

Etape
(0.3) 1. Modelarea problemei ca o problemă de satisfacere a constrângerilor
- identificarea variabilelor, domeniilor, restricțiilor
- citirea și inițializarea acestora pentru o instanță

(0.5) 2. Implementarea metodei FC
(0.2) 3. Implementarea metodei de ordonare a variabilelor

În cadrul laboratorului 4 trebuie rezolvat punctul 1.

Resurse
Artificial Intelligence: A Modern Approach, capitolul 5 http://aima.cs.berkeley.edu/newchap05.pdf
Forward checking
https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/2020_WQ_CS171/chap_6_b_CSPs_Constraint_Propagation_Structure.pdf

Homework

Consider a board of size nxn and n queens to be placed on the board. Some locations are blocked and we cannot place a queen. The problem is to place the queens on the board s.t. they do not attack. 2 queens attack each other when they are on the same line, column or diagonal.
Apply the Forward checking algorithm along with a variable ordering method to determine the solutions of a given instance.
(Test instances can be found at: https://www.csplib.org/Problems/prob080/data/)

Examples of instances and solutions:
- n=4, block(1,1), block(2,2), block(4,3)
Solution: queen(1,3), queen(2,1), queen(3,4), queen(4,2)

- n=5, block(1,1), block(2,2), block(3,3), block(4,5), block(5,5)
Solution: queen(1,5), queen(2,3), queen(3,1), queen(4,4), queen(5,2).

Consider the case presented in the first example. The steps of the Forward-checking + MRV algorithm are:

	X1	X2	X3	X4
Initial	2,3,4	1,3,4	1,2,3	1,2,3,4
After X1=2	2	4	1,3	1,3,4
After X2=4	2	4	1	1,3
After X3=1	2	4	1	3

Steps
(0.3) 1. Model the problem as a constraint satisfaction problem
- identify the variables, the domains, the constraints
- read and initialize them for an instance
(0.5) 2. Implement the FC method
(0.2) 3. Implement a variable ordering method.

Solve 1 in seminar 4.

Resources
Artificial Intelligence: A Modern Approach, Chapter 5 http://aima.cs.berkeley.edu/newchap05.pdf
Forward checking example
https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/2020_WQ_CS171/chap_6_b_CSPs_Constraint_Propagation_Structure.pdf