LABORATOR 9

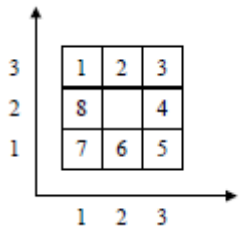
Problema "8-puzzle"

Se considera un tablou patratic de dimensiune 3x3, impartit in 9 campuri patrate de latura 1. Opt dintre aceste campuri contin cate o placuta etichetata cu un numar intre 1 si 8, in timp ce al noualea camp nu contine nimic. O placuta poate fi deplasata de pe campul sau pe un camp vecin, daca astfel nu se suprapune peste alta placuta, deci daca acest ultim camp nu contine nici o placuta. Fiind date doua configuratii C1 si C2 ale placutelor pe tabloul 3x3, se cere sa se spuna daca si cum este posibil sa se ajunga din configuratia C1 in configuratia C2 printr-un numar minim de mutari.

Vom trata aceasta problema ca pe una de cautare euristica. Un nod in spatiul starilor este de fapt o configuratie a placutelor in tablou. Deoarece ne intereseaza rezolvarea problemei printr-un numar minim de mutari (adica daca ne propunem sa
minimizam lungimea solutiilor), este firesc sa consideram ca lungimile arcelor din spatiul starilor sunt egale cu 1. Pentru rezolvarea problemei pot fi considerate mai multe functii euristice. Astfel, pentru o stare (configuratie) S, putem defini:
$\widehat{\boldsymbol{h}}_1(S)$ = numarul de placute care nu sunt la locul lor in starea S fata de starea finala.
$\widehat{\boldsymbol{h}}_2(S)$ = suma distantelor pe verticala si pe orizontala de la pozitia fiecarei placute in configuratia curenta (S), la pozitia respectivei placute in configuratia finala

Vom reprezentata in Prolog o configuratie printr-o lista a pozitiilor curente corespunzatoare locului liber, respectiv placutelor, fiecare pozitie fiind specificata printro pereche de coordonate, de forma X/Y. Ordinea acestor obiecte in lista este urmatoarea:
(1) pozitia curenta a locului liber (neocupat de nici o placuta),
(2) pozitia curenta a placutei etichetate cu 1,
(3) pozitia curenta a placutei etichetate cu 2, etc.

Exemplu:



[2/2, 1/3, 2/3, 3/3, 3/2, 3/1, 2/1, 1/1, 1/2]

In cele ce urmeaza, vom considera configuratia ilustrata in exemplul de mai sus, drept configuratia finala C2 din enuntul problemei, indiferent de alegerea configuratiei initiale. Prin urmare, predicatul scop este definit astfel:
scop ([2/2, 1/3, 2/3, 3/3, 3/2, 3/1, 2/1, 1/1, 1/2]).

1	3	4
8		2
7	6	5

a)

2	8	3
1	6	4
7		5

b)

2	1	6
4		8
7	5	3

c)

Figura 1 Trei exemple de configuratii initiale

In acest caz, putem defini o noua euristica $\hat{h}_3$ astfel

$\hat{h}_3(S) = \hat{h}_2(S) + 3 * secv(S)$, pentru o stare S,

unde secv(S) este un indice al secventialitatii, adica al gradului in care placutele sunt deja ordonate in configuratia actuala, in raport cu ordinea lor in configuratia finala. Acest indice este calculat ca si suma indicilor corespunzatori fiecarei placute, dupa cum urmeaza:

- indicele placutei aflate in centrul tabloului este 1;
- indicele unei placute aflate intr-o pozitie diferita de centru este 0, daca acea placuta este urmata (in sens invers-trigonometric) de succesorul sau din configuratia finala;
- in orice alta situatie, indicele este egal cu 2

De exemplu, pentru configuratia a S din **Figura 1** a), secv(S_a) = 6

.

Predicatele specifice problemei eight-puzzle:

%Definim relatia de succesiune; consideram ca toate arcele au costul 1

s([Gol|Placute],[Gol1|Placute1],1):-

h([Gol|Placute],H):-

arata_solutie([Gol|Placute], H):-

scop([2/2,1/3,2/3,3/3,3/2,3/1,2/1,1/1,1/2]).
initial([2/1,1/2,1/3,3/3,3/2,3/1,2/2,1/1,2/3]).
puzzle:-tell('C:\\best_first_output.txt'),
initial(Poz),bestfirst(Poz,Sol),arata_solutie(Sol),told.

%Strategia best-first

%Predicatul bestfirst(Nod_initial,Solutie) este adevarat daca Solutie este un drum (obtinut %folosind strategia best-first) de la nodul Nod_initial la o stare scop

bestfirst(Nod_initial,Solutie):- expandeaza([],l(Nod_initial,0/0),9999999,_,da,Solutie).
expandeaza(Drum,l(N,_),_,_, da,[N|Drum]):-scop(N).

%Caz 1: daca N este nod scop, atunci construim o cale solutie

expandeaza(Drum,l(N,F/G),Limita,Arb1,Rez,Sol):- F=<Limita,
(bagof(M/C,(s(N,M,C), \+ (membru(M,Drum))),Succ),!,
listasucc(G,Succ,As),cea_mai_buna_f(As,F1),
expandeaza(Drum,t(N,F1/G,As),Limita,Arb1, Rez,Sol);
Rez=imposibil).

% Caz 2: Daca N este nod frunza a carui $\hat{f}$-valoare este mai mica decat Limita,atunci ii %generez succesorii si ii expandez in limita Limita

expandeaza(Drum,t(N,F/G,[A|As]),Limita,Arb1,Rez,Sol):-F=<Limita,
cea_mai_buna_f(As,BF),min(Limita,BF,Limita1),
expandeaza([N|Drum],A,Limita1,A1,Rez1,Sol),
continua(Drum,t(N,F/G,[A1|As]),Limita,Arb1,Rez1,Rez,Sol).

% Caz 3 Daca arborele de radacina N are subarbori nevizi si $\hat{f}$-valoarea este mai mica decat % Limita, atunci expandam cel mai "promitator" subarbore al sau; in functie de rezultatul %obtinut Rez vom decide cum anume vom continua cautarea prin intermediul procedurii %(predicatului) continua

expandeaza(_,t(_,_,[]),_,_,imposibil,_):-!.

%Caz 4: pe aceasta varianta nu o sa obtinem niciodata o solutie

expandeaza(_,Arb,Limita,Arb,nu,_):-f(Arb,F),F>Limita.

%Caz 5: In cazul unor $\hat{f}$ -valori mai mari decat Bound, arborele numai poate fi extins

continua(_,_,_,_,da,da,Sol).

continua(P,t(N,F/G,[A1|As]),Limita,Arb1,nu,Rez, Sol):-
insereaza(A1,As,NAs),cea_mai_buna_f(NAs,F1),
expandeaza(P,t(N,F1/G,NAs),Limita,Arb1,Rez,Sol).
continua(P,t(N,F/G,[_|As]),Limita,Arb1,imposibil,Rez,Sol):- cea_mai_buna_f(As,F1),
expandeaza(P,t(N,F1/G,As),Limita,Arb1,Rez,Sol).
listasucc(_,[],[]).
listasucc(G0,[N/C|NCs],Ts):-G is G0+C,h(N,H),F is G+H,listasucc(G0,NCs,Ts1),
insereaza(l(N,F/G),Ts1,Ts).
%Predicatul insereaza(A,As,As1) este utilizat pentru inserarea unui arbore A intr-o lista de %arbori As, mentinand ordineaimpusa de $\hat{f}$-valorile lor
insereaza(A,As,[A|As]):-f(A,F),cea_mai_buna_f(As,F1),F=<F1,!.
insereaza(A,[A1|As],[A1|As1]):-insereaza(A,As,As1).
min(X,Y,X):-X=<Y,!.
min(_,Y,Y).
f(l(_,F/_),F). % f-val unei frunze
f(t(_,F/_,_),F). % f-val unui arbore
% Predicatul cea_mai_buna_f(As,F) este utilizat pentru a determina cea mai buna $\hat{f}$-valoare %a unui arbore din lista de arbori As, daca aceasta lista este nevida; lista As este ordonata %dupa f^ -valorile subarborilor constituenti
cea_mai_buna_f([A|_],F):-f(A,F).
cea_mai_buna_f([],999999).
% In cazul unei liste de arbori vide, $\hat{f}$-valoarea determinata este foarte mare

Output:

```
-------
2 8 3
1 6 4
7   5
-------
2 8 3
1   4
7 6 5
-------
2   3
1 8 4
7 6 5
-------
  2 3
1 8 4
7 6 5
-------
1 2 3
  8 4
7 6 5
-------
1 2 3
8   4
7 6 5
```