Documentatie tema 1 “Placi suprapuse”

Constantin Gabriel-Adrian, grupa 242

1. Modalitatea de apelare a programului

Programul va fi apelat din linia de comanda. Forma generala a unei apelari este *python placi_suprapuse.py input_path output_path nsol timeout* unde:

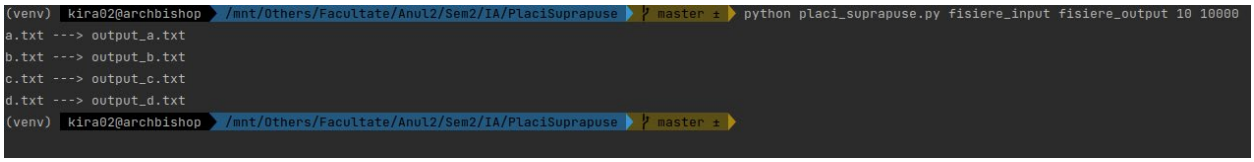
input_path – calea fisierului de input

output_path – calea fisierului de output

nsol – numarul de solutii de calculat

timeout – timpul maxim la care programul sa se opreasca, exprimat in milisecunde

De exemplu, daca avem un folder numit *fisiere_input* in interiorul folderului proiectului (ce contine fisierele de intrare ‘a.txt’, ’b.txt’, ‘c.txt’, ‘d.txt’), vrem sa cream un folder numit *fisiere_output* si dorim sa generam 10 solutii cu timpul timeout de 10 secunde, vom utiliza comanda:



Programul afiseaza in consola si fisierele de input gasite de acesta la calea specificicata si creaza output-ul lor (fisierele fiind denumite dupa modelul ‘*output_’ + ‘file_name’*).

2. Euristicile folosite
    a. Euristica banala

    Euristica banala consta in verificare starii. Daca starea noastra este si finala (i.e. nu exista bile in matrice) atunci aceasta va returna valoarea 0. Altfel, vom intoarce valoarea 1

    b. Euristica admisibila 1

Aceasta euristica se bazeaza pe ideea de numarare a bilelor aflate in matrice. Vom contoriza numai bilele care nu sunt suprapuse. Daca se intalneste acest caz, atunci vom considera stack-ul ca fiind o singura bila.

Corectitudine:

Realizam acest lucru deoarece exista cazul in care, pe o coloana, toate liniile in afara de ultima contin bile, iar o mutare a piesei de pe ultima linie ar duce la disparitia tuturor bilelor, astfel costul de la nodul curent la nodul scop este 1. Daca nu am fi tratat acest caz si am fi mers pe simpla idee de numarare, am fi obtinut de exemplu un cost estimat de n – 1 unde n este numarul de linii ale matricei. Deoarece acesta este mai mare decat costul real, am fi incalcat definitia eursiticii admisibile.

Pentru restul cazurilor, stim ca pentru ca o bila sa coboare avem un cost de 1, deci vom avea nevoie de cel putin m mutari de cost 1, unde m este numarul de bile din matrice. Deci costul estimat de la nodul curent la nodul scop va fi mai mic sau egal cu costul de la nodul curent la nodul scop, indeplinindu-se asftel conditia ca o euristica sa fie admisibila.

c. Euristica admisibila 2

Aceasta euristica pleaca de la ideea celei precedente. In loc sa ne bazam pe numararea bilelor dintr-o anumita stare, vom lua in calcul si pozitia lor. Plecam cu un contor ce reprezinta distanta de la bila la ultima linie deoarece pentru a cobora bila va fi nevoie de cel putin tot atatea miscari de cost 1. Acest lucru este valid deoarece din constrangerile problemei stim ca o bila nu poate cadea mai mult de un nivel.

Astfel, putem distinge mai multe cazuri: o placa deplasata duce la coborarea a 2 bile ce nu sunt suprapuse, nici-o placa din jurul bilei nu conduce printr-o singura miscare la coborarea unei bile.

Corectitudine

In primul caz, vom scadea contorul nostru cu o unitate deoarece a dus la scaderea cu un nivel a 2 bile. Astfel, ne asiguram ca nu vom numara de 2 ori pasii facuti de o bila.

In cel de-al doilea caz, este clar ca vom avea un cost de deplasare mai mare ca 1 deoarece nu exista o singura mutare care sa duca la coborarea cu un nivel a bilei. Vom avea nevoie deci de minim 2 mutari, una de cost x si cealalta de cost 1 pentru a cobora apoi bila. Astfel pe caz general, costul necesar mutarii bilei va fi cel putin costul deplasarii celei mai mici placi fara a muta bila pe orizontala, si anume 1 + dimensiunea placii minime. Acest cost va fi adunat contorului initial respectand astfel conditia conform careia costul estimat de la nodul curent la nodul scop este mai mic sau egal cu costul de la nodul curent la nodul scop.

d. Euristica neadmisibila

Bazandu-ne pe cazurile discutate la euristicile admisibile anterioare, putem deduce usor o euristica neadmisibila netratand cazul in care bilele sunt suprapuse. Vom aduna distanta pana la ultima linie de la fiecare bila, indiferent daca aceasta este suprapusa sau nu. Astfel, vom obtine un cost estimat mai mare decat cel real.

De exemplu, daca ne aflam in starea in care avem n-1 bile suprapuse pe aceeasi coloana(unde n reprezinta numarul de linii din matrice) si avem posibilitatea sa mutam a placa de pe ultimul nivel astfel incat toate acestea sa cada, algoritmul nsotru va estima h-ul ca fiind n-1 + n-2 + ... + 1 = (n-1)n/2. In realitatea insa, costul de la starea noastra la starea finala este 1. Deci obtinem costul estimat de la nodul curent la nodul scop este mai mare decat costul de la nodul curent la nodul scop. Astfel, invalidam conditia de admisibilitate a euristicii.

3. Validari si optimizari

Am realizat o functie care verifica corectitudinea datelor i.e daca matricea data este valida. Aceasta verifica daca exista placi/bile in aer, daca numarul de coloane din input nu este consistent, daca o bila cade mai mult de 1 nivel. Astfel nu vom mai continua inutil cu o stare care nu este valida.

Starile le-am reprezentat sub forma de matrice, adica o lista de liste de char-uri. Am considerat ca aceasta reprezentare este optima deoarece trebuie retinuta pozitia relativa a tuturor obiectelor.

4. Compararea algoritmilor

Fisier 1:

aa*bb*

.ddde*

..ffgg

..jj.i

Fisier 2:

aa**bb.

.cccd..

..eeef.

..ggii.

Tabela:

Fisier	Metrica	UCS	A*(banal)	A*(adm1)	A*(adm2)	A*(neadm)	A*opt(banal)	A*opt(adm1)	A*opt(adm2)	A*opt(neadm)	IDA*(banal)	IDA*(adm 1)	IDA*(adm 2)	IDA*(neadm)
1 (sol k=7)	Lungime	9	9	9	9	10	9	9	9	9	9	9	9	10
	Cost	15	15	15	15	17	15	15	15	15	15	15	15	17
	Nr maxim noduri	3816	1493	1234	1000	704	307	282	207	148	794	698	591	53
	Nr total noduri	5130	2023	1618	1318	894	555	495	371	229	5528	4352	3329	1945
	Timp (ms)	811	143	108	80	63	28	26	20	11	204	168	140	65
2 (sol k=1)	Lungime	7	7	7	7	9	7	7	7	9	7	7	7	9
	Cost	9	9	9	9	11	9	9	9	11	9	9	9	11
	Nr maxim noduri	1406	396	274	194	219	167	132	107	126	198	166	133	126
	Nr total noduri	1702	479	328	229	259	235	182	147	164	1108	672	277	297
	Timp (ms)	122	23	15	10	11	11	10	8	8	39	26	12	11

Astfel, cu ajutorul tabelului putem identifica diferente importante intre algoritmi. Cel mai lent este UCS, neavand nici-un avantaj fata de ceilalti algoritmi.

Astar cu euristica banala reuseste sa obtina un timp mult mai bun decat UCS, cu un numar semnificativ mai mic de noduri calculate si memorate comparativ cu cel din urma. Acesta poate fi insa imbunatati cu ajutorul euristicilor admisibile 1 si 2, cea din urma performand mai bine decat prima. In mod evident, euristica neadmisibile ne ofera si drumuri care nu au costul minim. in primul fisier, putem observa un astfel de drum la a 7-a solutie generata, insa in cel de-al 2-lea inca de la inceput obtinem o solutie cu un cost mai mare.

O varianta mult mai optima este Astar optimizat, trend-ul pastrandu-se intre euristici. Dezavantajul este reprezentat de faptul ca acesta ne genereaza doar o solutie, in timp ce UCS, Astar si IDA* obtin un numar dat de solutii.

Astfel, IDA* are un timp de rulare mai mare decat celelalte doua variante de Astar, insa are avantajul ca utilizeaza mai putina memorie. Chiar daca a fost mai lent, observam in primul fisier de exemplu ca acesta a retinut un numar maxim de noduri cu aproximativ 300 mai mic decat Astar.