Comparatie A* - Hill Climbing

Minca Ecaterina-Ioana

334CA

1 Descriere generala

Am avut de implementat problema generarii unui orar, dandu-se datele despre materii si profesori, avand si anumite constrangeri obligatorii de respectat, cat si unele optionale.

Algoritmi Am avut de realizat aceasta cerinta folosind algoritmul A*, cat si Hill Climbing.

2 Reprezentarea starilor

Am decis ca starile mele sa contina atat starea nou generata, cat si pe cele anterioare, practic sa retin intr-o stare o solutie partiala. Astfel, in starea initiala, imi voi lua toate informatiile necesare din dictionarul cu toate specificatiile pe care mi le voi pastra eu in alte dictionare pentru a putea fi mai usor de accesat si sa nu trebuiasca sa fie parcurs acel dictionar cu toate datele de fiecare data. Pentru urmatoarele stari, voi primi ca parametrii starile de pana atunci si starea nou generata. De asemenea, am retinut un dictionar cu asocierea dintre materie si numarul de studenti ramasi de pus in sali, care va fi modificat din stare in stare si transmis mai departe. Si un dictionar in care cheia va fi un tuplu de (zi, interval, sala), iar valoarea un tuplu (profesor, materie). O stare este finala daca toate materiile au acoperirea necesara.

3 A*

3.1 Functia f

Pentru a rezolva aceasta problema folosind A*, am considerat costul g ca fiind numarul de constrangeri optionale pe care solutia partiala o incalca, iar functia de euristica fiind numarul de constrangeri obligatorii incalcate. Am ales acest h cu gandul ca in starea finala va fi 0 si fiind consistenta, deoarece cu cat s-ar apropia de solutia finala, numarul de constrangeri obligatorii incalcate ar deveni din ce in ce mai mic. Acestea nu au fost de ajuns pentru a rezolva problema, astfel in momentul in care 2 stari au acelasi cost(f), voi verifica care dintre ele are mai putini studenti ramasi de plasat in sali si voi lua starea care are numarul minim si numar de constrangri optionale minim, iar daca si acesta este egal, voi

lua pur si simplu starea cu numarul de studenti minim. De asemenea, la fiecare pas voi reordona materiile din dictionarul meu in care le am stocate(materie - numar de studenti ramasi), astfel incat atunci cand parcurg acest dictionar sa iau prima oara materiile cu cel mai mic numar de studenti, cei mai putini profesori si cele mai putine intervale in care poate fi predata materia(practic voi ordona de 3 ori acel dictionar).

3.2 Problema test constrangere incalcata

Dupa modul in care am descris problema, este evident ca pe testul cu constrangeri incalcate voi avea o problema, deoarece la mine este absolut necesar in solutie sa nu incalc vreo constrangere, atat hard cat si soft. De aceea, am fost nevoita ca pentru acel test sa adaug un numar de iteratii in algoritmul de A* care practic acela imi semnaleaza cand programul cicleaza si nu poate gasi o solutie buna. Din aceasta cauza, am decis ca pentru materiile care inca nu si-au acoperit complet numar de studenti, sa le pun iar la dispozitie toate salile libere, dar de aceasta data intr-un mod mult mai putin restrictiv(sa nu aiba nicio constrangere hard incalcata si maxim una soft:)...).Pentru a testa acest numar de constrangeri incalcate, practic voi simula ca adaug o noua intrare de tip (zi, interval, sala) = (prof, materie) in starea mea curenta, pe care o voi sterge ulterior daca aceasta nu indeplineste conditiile.

3.3 Modificare algoritm laborator

Ca modificare asupra algoritmului de la laborator, am sters partea de refacere a drumului, nenecesara acum, deoarce retin deja solutia partiala prin intermediul starilor.

4 Hill Climbing

4.1 Tipul de Hill Climbing

Am decis sa folosesc algoritmul clasic de Hill Climbing deoarece, cu euristicile alese, mergea mai prost, dezavantajul fiind ca dureaza destul de mult pana genereaza toate starile vecine si le verifica pe fiecare pentru a alege minimul.

4.2 Functia de evaluare

Functia de evaluare am pastrat-o de la A*, practic e tot numar de constrangeri obligatorii adunat cu numar de constrangeri optionale. De data aceasta in schimb, in momentul cand eval(S) = eval(S'), am comparat numar de constrangeri obligatorii si daca si acesta era egal, comparam daca minState sau vecinul sau au mai putini studenti ramasi de pus in sali si iau vecinul in considerare doar daca are mai putini.

4.3 Modificare algoritm initial

Spre deosebire de algoritmul de la laborator, unde ne foloseam de egalitate pentru a vedea daca suntem intr-un minim local, eu am mers pe cazul de >=, practic daca niciun vecin nu e mai bun decat cea curenta.

4.4 Problema test constrangeri incalcate

Folosind acelasi mod de a gandi euristicile, evident ca si la Hill Climbing am intampinat problema la acest test, pe care am rezolvat-o exact ca mai sus. In plus, am modificat algoritmul astfel incat el sa vada daca se afla in acel minim local doar daca nu are nicior constrangere obligatorie incalcata si cel mult una optionala. In cazul in care starea curenta si starea minstate sunt la fel de cateva iteratii (50 a fost de ajuns de mult) inseamna ca am intrat in aceeasi gaura neagra din care nu mai poate gasi solutia finala si atunci voi aplica metoda descrisa mai sus, sa trec iar prin zile, intervale, sali, profesori sa incerc sa gasesc o solutie care nu incalca retrictii hard.

5 Comparatia celor 2 algoritmi

5.1 Stari

Din punct de vedere al numarului de stari, cei 2 algoritmi nu sunt foarte diferiti.

A*

- dummy: 540
- mic: 16266
- mediu: 116796
- mare: 322920
- constrans: 48546 (lasandu-l eu sa genereze pana la un numar de 100 de iteratii, dupa care intervine modalitatea de revenire)

Hill Climbing

- dummy: 506
- mic: 16266
- mediu: 116796
- mare: 322920
- constrans: 48606

Se poate observa ca numarul de stari este cam acelasi in mare.

5.2 Timp

Voi arata in medie care este timpul pe 5 rulari ale fiecarui test

A*

- dummy: 0.11s
- mic: 7.37s
- mediu: 1m 22s
- mare: 8m 0.1s
- constrans: 30.68s

Hill Climbing

- dummy: 0.18s
- mic: 14.8s
- mediu: 2m 39s
- mare: 17m 27s
- constrans: 58.6s

5.3 Numar restrictii incalcate

Atat la A*, cat si la Hill Climbing, am obtinut pe dummy, orar mic, mediu, mare 0 constrangeri hard si 0 soft, iar pe orar-constrans-incalcat am obtinut 0 hard si 1 soft.

6 Concluzii

Concluzia este ca dintre cei 2 algoritmi este de preferat A*, ruleaza si mult mai repede, desi exploreaza la fel de multe stari ca Hill Climbing.

6.1 Implementarea mea

Initial cand am gandit toata tema, nu am tinut cont deloc de exista acelui test cu constrangerea incalcata care mi-a cauzat si problemele. Acea problema apare mai ales pentru ca abordarea mea este una destul de greedy, luand in calcul mereu starea cu numar mai mic de copii ramasi daca numarul de constrangeri e la fel(iar acest lucru se intampla des) plus euristica si costul ales si de aici vine si imposibilitatea sa mai revina sa incerce alta solutie. De aceea si decizia de a reparcurge de la inceput pentru a incerca sa gaseasca o solutie in care toate constrangerile hard sunt respectate. De asemenea, tot din cauza modului de gestiune pe cazul de egalitate se poate observa ca se aleg mai ales pe testele mari sa fie plasati majoritatea in salile mari pe tot parcursul saptamanii, cele mici ramanand multe goale.