Ionescu Cristina
333CC

Tema 1 – IA

1. Hill-Climbing

- **Procesari in main:**

Imi creez niste dictionare care ma vor ajuta in program pentru interval orare, zile, materii, Sali, profesori, VARIANTE_SALI in care vor fi pentru fiecare sala toate tuplurile de (nume_prof, materie) care pot fi facute.

Pe langa asta mai am si o matrice de dictionare de marimea Orarului pe care il vom face. Se numeste VARIANTE_ORAR si contine pentru fiecare zi si interval un dictionary cu salile care pot fi alese. Fiecare sala va avea o lista de tupluri (nume_prof, materie) care se potivesc in orar in locul respective.

- ***Starea initiala:***

Prima data am incercat sa plec de la orarul gol, dar dura foarte mult gasirea unei solutii valide cu cost mic, asa ca am decis ca starea mea initiala sa fie o varianta de orar care bifeaza toate constrangerile hard.

Starea initiala se creaza in [generate_first_table]. Incep prin a face o matrice numita ORAR, pentru fiecare zi si interval continand un dictionary gol.

Voi avea un:

- dictionar copie_nr_studenti_materii, care va contine materiile si cati studenti mai avem de acoperit. Din dictionarul asta se va face o lista ordonata descrescator cu numele materiilor dupa numarul de elevi care mai sunt de acoperit.

- dictionar prof_nr_clase_orar, care va contine numele profesorilor si de cate ori apar ei in orar. La fel ca mai sus, dictionarul se va folosi pentru crearea unei liste de nume ale profesorilor ordonata crescator dupa numarul de materii pe care acestia le predau si dupa numarul de aparitii pe care le au in orar

- o lista cu numele salilor ordonate descrescator dupa capacitatile lor

In ordinea urmatoare de prioritati: sala, interval_orar, zi, materie, prof, se va alege cate un element din VARIANTE_ORAR (din intervalul afferent – ora, zi, sala), in ordinea in care respecta conditiile de mai sus. (am grija sa nu incalc constrangerile hard) si voi popula tot orarul. La funal voi avea o varianta de orar care bifeaza toate constrangerile hard, ramane sa le rezolvam pe cele soft.

Aceasta este abordarea aleasa de mine, desigur, in functie de cum se genereaza la inceput orarul, vor fi alte rezultate. Nu stiu daca este cea mai buna abordare dar este mult mai eficienta, poate daca creezi mai multe variante diferite de orare initiale sunt mai multe sanse sa se ajunga la o solutie care nu incalca nicio constrangere. Prima mea incercare a fost sa plec de la un orar gol, dar dura prea mult sa ajung la un orar valid.

- ***Reprezentarea starilor:***

Am incercat sa ma iau dupa modelul facut la laborator si sa imi fac o “lista de vecini”. Aceasta lista nu va fi retinuta undeva, dar voi retine cel mai bun vecin gasit. Un vecin reprezinta orarul curent dupa ce a fost aplicat o schimbare pe el.

O schimbare poate fi un switch intre 2 zile sau interval orare pe aceesi sala sau o schimbare intre 2 profesori care pot preda aceleazi materii. Pentru ambele abordari se vor incerca toate variantele posibile pentru profesorii din WRONG_PLACEMENTS. Pentru fiecare orar creat se calculeaza evaluarea, cel cu cea mai buna evaluare din runda respectiva va fi intors in functia de hill climbing si comparat cu “starea initiala” din momentul respectiv.

La chimbarea starii initiale se verifica cu evaluate_for_first care verifica doar constrangerile hard si care genereaza acest WRONG_PLACEMENTS in el se retin grupuri de forma (ora, zi, sala, prof, materie) care incalca constrangeri soft.

- ***Optimizari fata de variant de la laborator:***
  - incep cu o stare initiala partial corecta pe care o ajustez cu hill climbing, scade foarte mult timpul de procesare
  - la generarea orarului de inceput incerc sa balansez putin numarul maxim de profesori care pot sa predea intr-un orar complet, deoarece daca sunt prea multi profesori si foarte putine materii se va face o acoperire mai buna cu o limita mai mica
  - tot pentru orarul initial incerc sa refac ordinea de prioritati pentru materii si profi la fiecare completare a orarului pentru a ma asigura ca am o oarecare uniformizare in acoperirea lui
  - fac doua tipuri de creare de vecini pentru a acoperii cat mai multe cazuri (reusesc sa fac asta si pentru ca nu dureaza mult operatiile)
  - la generarea vecinilor, incerc sa fac schimbari doar pentru locurile unde sunt problem (WRONG_PLACEMENTS) in loc sa iau fiecare element din orar la rand

- ***Observatii:***
  - Se pot ajunge la solutii mai bune in functie de starea initiala generata sau daca se imbunatateste “eusistica” facuta de mine
  - Cu toate ca teoretic se poate ajunge la o solutie fara constrangeri incalcate algoritmul acesta are problema majora, se poate bloca intr-un minim local (ceea ce se poate observa ca se si intampla). O optimizare cred ca ar putea fi sa generezi mai multe variante de orar initial cu constrangeri hard neincalcate si sa rulezi hill climbing pe fiecare in speranta ca in macar una din variante se ajunge la solutia ideala, dar nu imi pare prea fezabil. Alta solutie ar putea fi varianta de hill climbing cu restart, dar eu nu am reusit sa fac o abordare pentru ea care sa si mearga, de aceea am ramas la variant actuala.

2. Monte-Carlo Tree Search

- **Starea initiala**:

Este aceeasi ca cea folosita mai sus la Hill Climbing. Pornesc tot de la un orar deja generat, care nu incalca nicio constrangere Hard si pentru el incerc sa gasesc o variant mai buna

- **Reprezentarea starilor**:

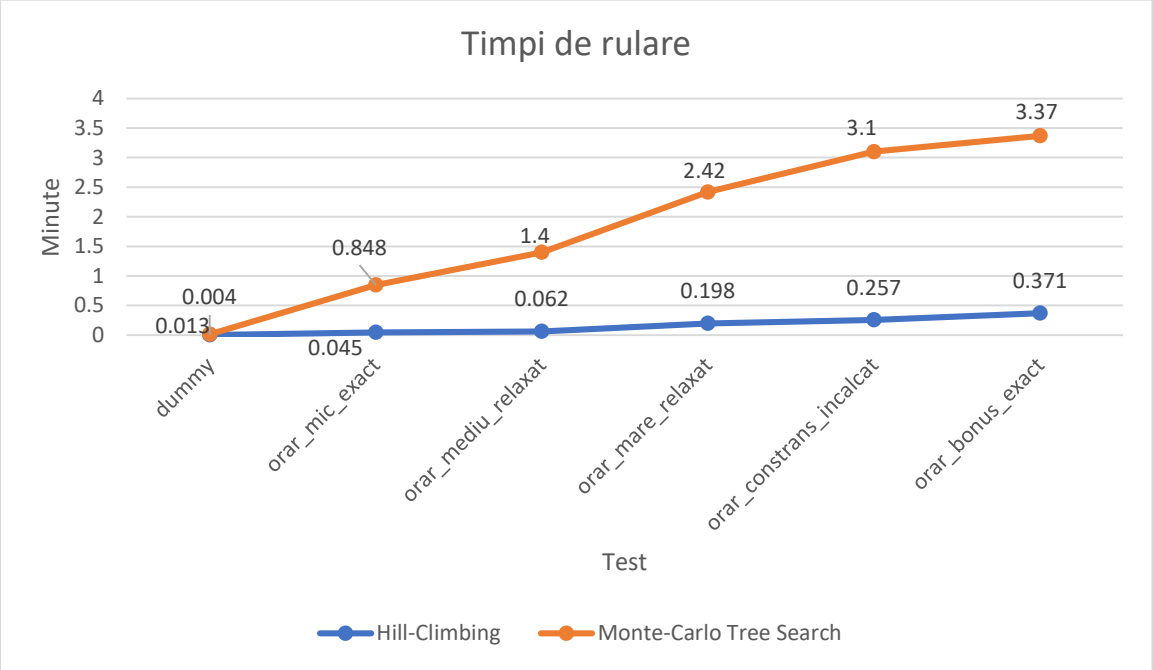
M-am folosit foarte mult de cum am generat eu vecini la HC, aici insa, nu il mai retin pe cel mai bun, fac o lista cu cele mai bune incercari gasite de la inceputul cautarii respective de vecini. Apoi iau random din aceasta lista o variant de orar. Aceasta variant va devein copilul nodului anterior.

- **Optimizari fata de variant de la laborator**:
  - Am incercat ca variantele de stari pe care le fac sa nu fie create luand efectiv cu random o variant posibila pentru oricare pozitie din orar. Incerc sa fac mutari doar pentru pozitiile care stiu ca sunt proaste si aleg urmatoarea stare (copilul) doar dintre acestea
  - Am incercat sa fac o optimizarea pentru UCT care seaman cu UCB1 (Upper Confidence Bound), acum nu stiu cat de bine mi-a iesit, dar pare ca imbunatateste putin situatia

- **Observatii**:
  - Din cate am observant, algoritmul meu are tendinta sa mearga mai mult in andancime, am incercat sa ii fac modificari pentru a regla comportamentul acesta dar nu mai ajungeam la solutii atat de bune.
  - Solutia gasita de mine nu este cea mai buna, dar gaseste o solutie cat de cat buna intr-un timp relative scurt
  - Am incercat sa randomizez mai mult cautarea si sa pornesc de la un orar initial gol. Astfel trebuia sa nu ma mai bazez pe faptul ca constrangerile hard nu sunt incalcate, dar nu am reusit sa gasesc o solutie care sa fie optima ca timp si sa ajunga la o solutii macar la fel de bune ca cele gasite cu algorimul actual.
  - Dat find faptul ca MCTS se bazeaza mult pe randomizare pentru problema data (cea de generare de orar) nu mi se pare cea mai buna alegere, pentru problem mari poate dura foarte mult gasirea unei solutii
  - Am folosit 500 de iteratii

3. Comparatie

- Timp de executie:
  - Ce am pus mai jos sunt timpii de rulare care au iesit la rularile mele
  - Pentru problema data (cea de generare de orar) mi se pare mult mai portivit un Hill Climbing cu Restart decat un MCTS – asta spun strict din experienta acumulata in incercarea rezolvarii temei. Probabil folosind euristici si abordari mai bune, realitatea sa fie alta, doar ca eu nu am reusit sa le fac din pacate.



- **Numar de stari construite**:
  - HC: ma voi raporta la numarul vecinilor construiti, pentru ca pe baza lor se allege cea mai buna variant de mutare, care va devein stare initiala => pentru o stare initiala se vor face nr_pozitii_cu_constrangeri_incalcate * nr_ore * nr_zile * nr_clase * 2 = stare_pe_iteratie atatea stari se fac la o verificare, numarul_total_stari = stare_pe_iteratie * iteratii_pana_la_min_local

- MCTS: numarul de stari construite va fi egal cu numarul de noduri copil create la fiecare iteratie (in codul meu asta variaza mult in functie de cate probleme sa gasesc la rularea respective pentru orar)

- **Calitatea solutiei**:
  - Aici se pot observa rezultatele pentru rularile mele. Am explicat mai sus motivele pentru care nu sunt perfecte(fara constrangeri soft incalcate)
  - Pentru MCTS am pus cele mai bune valori date in mai multe rulari, calitatea solutiei variaza putin in functie de rulare cu o diferenta +-1/2 constrangeri soft incalcate