Inteligenta artificiala: Tema 1-Generare Automata Orar

**Universitatea Politehnica București Facultatea de Automatica și Calculatoare**

**Descrierea problemei**

Problema consta in cautarea unui orar care imbina conditii hard cu cele soft, incercand sa gaseasca orarul perfect.

Conditiile hard se refera la posibilitatea existentei orarului si contine verificari de tipul:

* un profesor poate preda doar o materie intr-un interval
* un profesor are maxim sapte ore pe saptamana
* intr-o sala incape un numar de studenti mai mic sau egal cu capacitatea salii
* toti studentii de la o materie trebuie sa fie alocati
* toti profesorii predau doar pe ce sunt specializati
* in sali se pot preda doar materiile compatibile cu sala

Conditiile soft se refera preferintele profesorilor legate de zile si intervale. Acestea pot fi incalcate, la nevoie.

**Hill Climbing**

**Reprezentarea starilor si a restrictiilor**

***Stari:***

Pentru a reprezenta starile am folosit clasa State inspirata din laborator. Aceasta contine orarul, numarul de intervale libere per profesor, numarul de studenti nealocati la fiecare materie, numarul de conflicte soft si disponibilitatea salilor si a profesorilor pe interval.

***Restrictii:***

Restrictiile hard sunt respectate automat prin generarea succesorilor care indeplinesc cerintele hard. Pentru a indeplini cerintele sunt folosite dictionare in stare, ce retin studentii nealocati, disponibilitatea salilor si a profesorilor pe interval si numarul de intervale alocate fiecarui profesor. Acestea sunt actualizate pentru fiecare stare noua generata si pentru fiecare noua adaugare in orar.

Restrictiile soft sunt stocate tot in stare si sunt actualizate in acelasi timp cu cele hard.

**Hill Climbing**

**Ideea generala**

Algoritmul porneste cu o stare initiala in care orarul este gol. Succesorii starii initiale sunt stari care contin orare in care este adaugat o singura materie intr-un interval. Toti acesti succesori respecta conditiile hard. Functia de evaluare a succesorilor este formata din numarul de studenti ramasi nealocati la materie plus conflictele soft. Folosind aceasta functie se ajunge la rezultate bune. Mai este aplicat inca o data algoritmul hill climbing, cu o noua functie de generare a succesorilor. Noua functie genereaza stari cu modificari ale orarului hard generat de primul apel. De aici, hill climbing foloseste aceasi functie de evaluare(studenti ramasi plus conflicte soft), dar studentii ramasi sunt zero deci vor conta doar conflictele soft.

Am folosit 2 variante de Hill Climbing: first choice si stocastic.

Pentru first choice este ales primul succesor mai bun decat starea actuala. Aceasta varianta este mai rapid decat cea stocastica cu %TODO. Eficienta sa vine din utilizarea generatorului din python, nu se mai genereaza toate starile.

Pentru varianta stocastica este ales un succesor aleator, din lista celor mai buni decat starea actuala. Aceasta strategie poate duce la rezultate mai bune, dar are un timp de rulare mai mare, detaliat in grafice.

**Hill Climbing**

**Optimizari pentru hill climbing:**

* Generez doar stari care nu incalca cerintele hard pana cand ajung la un orar hard ce acopera toti studentii cu toate materiile.
* Pentru varianta de first choice folosirea generatorului scade timpul de rulare deoarece nu mai este nevoit sa genereze toate starile succesoare.

O alta optimizare este faptul ca aplic o data hill climbing pentru a imi gasi o solutie care imbina costul hard plus soft. Aceasta gaseste o solutie care respecta cerintele hard si le minimizeaza pe cele soft. De la aceasta solutie hard mai aplic o data hill climbing pentru a minimiza si cerintele soft.

In tabel sunt evidentiate diferentele de cost inainte de a doua rulare de optimizare a hill climb ului, cat si dupa rularea optimizarii.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dummy | orar\_mic\_exact | orar\_mediu\_relaxat | orar\_mare\_relaxat | orar\_constrans\_incalcat |
| 0 | 18 | 22 | 35 | 29 |
| 0 | 0 | 3 | 2 | 1 |

Obs! Datele din tabel sunt pentru hill climbing first choice.

O alta observatie legata de optimizari este despre functia de evaluare. Aceasta este compusa din: numarul de studenti neasignati + numarul de conflicte soft \* x. Am observat experimental faptul ca modificarea coeficientului „x” duce la urmatoarele rezultate:

Pentru un coeficient mai mare de 80 algoritmul gaseste cu greu o solutie care respecta hard, iar cand gaseste aceste solutii au soft-uri foarte mici. Pentru un coeficient mai mic, precum 10 algoritmul gaseste mereu solutii hard dar cu soft-uri mai mari.

**Hill Climbing**

O imagine care conține linie, captură de ecran, diagramă, pantă

Descriere generată automat**Rezultatele algoritmului hill climbing folosind First Choice**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | dummy | orar\_mic\_exact | orar\_mediu\_relaxat | orar\_mare\_relaxat | orar\_constrans\_incalcat |
| iteratii | 18 | 59 | 94 | 126 | 91 |
| stari | 119 | 2946 | 6957 | 31317 | 12250 |
| timp | 0.01s | 1.41s | 6.62s | 41.57s | 7.08s |
| soft | 0 | 0 | 3 | 2 | 1 |

**Hill Climbing**

O imagine care conține text, linie, captură de ecran, diagramă

Descriere generată automat**Rezultatele algoritmului hill climbing folosind Stocastic**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | dummy | orar\_mic\_exact | orar\_mediu\_relaxat | orar\_mare\_relaxat | orar\_constrans\_incalcat |
| iteratii | 19 | 49 | 91 | 117 | 84 |
| stari | 887 | 23544 | 171640 | 562210 | 72117 |
| timp | 0.07s | 4.37s | 42.08s | 286.93s | 18.69s |
| soft | 0 | 0 | 0 | 0 | 1 |

**Hill Climbing**

**Comparatie intre stocastic si first choice hill climbing**

Din cele doua tabele cu timpurile de rulare corespunzatoare fiecarui algoritm reiese ca versiunea de stocastic are un timp de rulare mai mare si produce rezultate mai bune(in general). Utilizand stocastic se poate cauta specific un cost anume, pe cand la first choice nu este valabila aceasta optiune. Avantajul first choice este in schimb timpul. Acesta ruleaza mai rapid si produce rezultate bune.

In concluzie, pentru o solutie acceptabila intr-un timp rapid este recomandata first choice, iar pentru o solutie buna este de preferat versiunea stocastic cu un timp de rulare mai mare.

**Constraint satisfaction problem**

**Ideea generala**

Primul pas este sa pornesc dintr-o stare initiala goala, iar variabilele contin domeniile initiale. Pe urma se aplica algoritmul de PCSP din laborator modificat si se cauta folosind backtracking o solutie care se incadreaza in costul acceptabil dat ca parametru. Atunci cand se verifica o noua asignare pentru o variabila, mai intai se verifica daca asignarea ar respecta cerintele hard. Daca respecta cerintele hard si costul soft este mai mic decat cel acceptabil atunci se continua backtracking-ul pentru a asigna si urmatoarea variabila. In caz de esec, se apeleaza recursiv tot pentru aceasi variabila, dar pe domeniul actualizat.

**Constraint satisfaction problem**

**Reprezentarea starilor si a restrictiilor**

***Stari:***

Reprezentarea stariilor este identica cu cea de la hill climbing, refolosind aceiasi clasa pentru modularitate.

***Restrictii:***

Restrictiile hard sunt respectate de algoritm prin verificarea solutiei cu noua adaugare de materie si profesor, prin parcurgerea dictionarelor salvate in stare. Se foloseste de aceiasi idee de verificare precum la hill climbing.

**Constraint satisfaction problem**

**Reprezentarea variabilelor si domeniului**

***Variabilele:***

Variabilele le-am reprezentat ca un tuplu de zi, interval si sala de curs. Aceasta reprezentare ajuta in eliminarea multor combinatii de profesor si sala din domeniu. Din punct de vedere al implementarii am facut o clasa ce contine drept atribute zi, interval, sala si o domeniul.

***Domeniul:***

Domeniul este reprezentat de tuplul profesor si sala. Acesta fiind atribut fiecarei variabile, se pot face multe taieri de domeniu in partea de initializare deoarece materiile au doar anumite sali in care pot fi predate.

***Exemplu:***

Variabila este ilustrata prin cerc, iar domeniul prin dreptunghi.

O imagine care conține text, carte de vizită, cerc, linie

Descriere generată automat

**Constraint satisfaction problem**

**Optimizari pentru constraint satisfaction problem**

Optimizarile cele mai mari pentru un algoritm de acest tip sunt eliminarea din domenii, cu cat domeniul este mai mic, cu atat algoritmul gaseste solutia mai rapid.

Lista cu optimizari concrete:

* eliminarea tuturor materiilor care si-au indeplinit acoperirea necesara din toate domeniile variabilelor
* eliminarea tuturor profesorilor care si-au indeplinit numarul de intervale maxim posibil de predat din toate domeniile variabilelor
* sortarea domeniului dupa numarul de studenti ramasi

Ideea de sortare a domeniului ajuta mult algoritmul deoarece se evita explorarile inutile ale spatiului de stari si se poate gasi mai repede o aproximare a solutiei. Prin sortarea domeniilor, pe anumite probleme se poate realiza o demonstratie pentru a arata corectitudinea. In cazul problemei de cautare a orarului nu se poate deoarece doar prin sortarea numarului de studenti ramasi nu este garantata gasirea unui orar.

Am incercat si alte optimizari precum dupa cele mai putine intervale ramase per profesor, dupa cel mai mic domeniu, dar nu a avut niciuna efecte bune pe toate orarele.

**Constraints satisfaction problem**

**Rezultatele algoritmului constraints satisfaction problem**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| input | dummy | orar\_mic\_exact | orar\_mediu\_relaxat | orar\_mare\_relaxat | orar\_constrans\_incalcat |
| iteratii | 68 | 60229 | 668 | 1454 | mult |
| stari | 68 | 60229 | 668 | 1454 | mult |
| timp | 0.01 | 28.27 | 1.48 | 6.47 | mult |
| soft | 0 | 0 | 0 | 0 | mult |

O imagine care conține linie, Interval, diagramă, text

Descriere generată automat

In cazul testului orar\_constrans\_incalcat algoritmul dureaza prea mult. Consider ca va gasi solutia, dar avand un numar mari de verificari pe care le parcurge chiar folosind acele optimizari. Sunt convins ca nu am gasit cea mai optima solutie pentru algoritmul PCSP, existand o metoda mai buna de a minimiza domeniul, sau o sortare mai eficienta.

Comparatia intre cei doi algoritmi

Hill climbing first choice si partial constraint satisfaction problem

Timp de executie

|  |  |  |
| --- | --- | --- |
| Input | Hill climbing | PCSP |
| Dummy | 0.01s | 0.01s |
| Orar\_mic\_exact | 1.41s | 28.27s |
| Orar\_mediu\_relaxat | 6.62s | 1.48s |
| Orar\_mare\_relaxat | 41.57s | 6.47s |
| Orar\_constrans\_incalcat | 7.08s | mult |

Analiza:

Din punct de vedere al timpului se observa ca ambii algoritmi se descurca mai bine pe inputuri diferite, dar hill climb reuseste sa gaseasca o solutie pe orarul constrans intr-un timp acceptabil.

Comparatia intre cei doi algoritmi

Hill climbing first choice si partial constraint satisfaction problem

Numar de stari

|  |  |  |
| --- | --- | --- |
| Input | Hill climbing | PCSP |
| Dummy | 119 | 68 |
| Orar\_mic\_exact | 2946 | 60229 |
| Orar\_mediu\_relaxat | 6957 | 668 |
| Orar\_mare\_relaxat | 31317 | 1454 |
| Orar\_constrans\_incalcat | 12250 | mult |

Analiza:

Din punct de vedere al starilor este aceiasi idee ca la timp, variaza in functie de test, fiecare algoritm are plusuri si minusuri.

Calitate a solutiei(soft incalcate)

|  |  |  |
| --- | --- | --- |
| Input | Hill climbing | PCSP |
| Dummy | 0 | 0 |
| Orar\_mic\_exact | 0 | 0 |
| Orar\_mediu\_relaxat | 3 | 0 |
| Orar\_mare\_relaxat | 2 | 0 |
| Orar\_constrans\_incalcat | 1 | mult |

Analiza:

Din punct de vedere al calitatii solutiei PCSP gaseste mereu cu costul pe care este setat, in schimb hill climb-ul gaseste o solutie buna, dar nu obligatorie perfecta mereu. Mai ales pentru hill climb-ul stocastic, solutia este variabila.