Nume: Buza Madalina Gabriela
Master: MSD2
Coordonator: Madalina Raschip

Analiza algoritmilor pentru problema “Meeting Scheduling“ -raport

Parametrii problemei:

- Numarul de meeting-uri - m;
- Numarul de agenti - n;
- Numarul de meeting-uri per agent - k;
- Distanta dintre locatiile meeting-urilor - in unitati de sloturi de timp;
- Dimensiune domeniului - numar de slot-uri de timp - l;

Scop: Organizarea meeting-urilor in slot-uri de timp, astfel incat fiecare agent sa ajunga la toate meeting-urile programate, altfel spus aceste meeting-uri nu trebuie sa se suprapuna. De asemenea, acestea trebuie stabilite intr-un interval specificat de time-sloturi.

Problema “Meeting Scheduling” a fost rezolvata utilizand 3 strategii. Acestea au fost utilizate pe mai multe instante ale problemei, acestea fiind citite ulterior din fisier.
Acesta este un exemplu de instanta a problemei de dimensiuni relativ mici:

5 3 32

0: 1 0 1 0 1
1: 1 1 1 0 0
2: 0 1 1 1 0

0: 0 1 2 1 3
1: 1 0 3 2 2
2: 2 3 0 1 2
3: 1 2 1 0 3
4: 3 2 2 3 0

Primul parametru de pe prima linie (“5”) reprezinta numarul de agenti, apoi “3” reprezinta numarul total de meeting-uri, iar “32” este numarul de timeslot-uri.
Apoi sunt reprezentate printr-o matrice, meeting-urile la care fiecare agent trebuie sa ajunga. In ex. A0->M0,M2,M4; A1->M0,M1,M2; A2->M1,M2,M3;

In fisier sunt specificate de asemenea distantele dintre meeting-uri. La fel, acest lucru este reprezentat printr-o matrice nrMeetings X nrMeetings.

Strategia 1:

Reprezint toate meeting-urile utilizand o structura de date IntVar [] de dimensiune m. Valoarea a meeting[i] corespunde intervalului de timp în care are loc meeting-ul i. Valoarea a meeting[i] pentru i în [0, m) este [0, l). Prin urmare, fiecare meeting are un singur moment de apariţie. Chiar dacă trebuie să participe mai mulţi agenţi la meeting, nu este posibil ca acest meeting să aibă început-uri diferite.

Constrangeri:

Metoda theMeetingsNotOverlaps primeste ca parametri 2 meeting-uri şi verifică dacă se pot suprapune folosind matricea de prezenţă.

- Dacă M1 şi M2 nu pot fi în paralel, atunci există cel puţin un agent care are M1 şi M2 în calendarul său. În acest caz, publicăm o constrângere pentru a ne asigura există suficient timp de călătorie între M1 şi M2.
- Dacă M1 şi M2 nu pot fi în paralel, timpul de călătorie între ele nu mai conteaza (niciun agent nu îi participă pe amândoi, deci niciun agent nu trebuie să călătorească de la M1 la M2 sau de la M2 la M1).

Strategia 2:

In aceasta abordare se pune accent pe matricea in care se reflecta meeting-urile la care trebuie sa participe fiecare agent. Se merge pe principiul ca fiecare agent are un calendar de dimensiune l (nr. de timeslot-uri). Prezenta valoarei -1 pe un timeslot inseamna ca nu exista niciun meeting in timeslot-ul respectiv, iar in rest inafara de aceasta valoare, vor aparea in vector valorile meeting-urilor la care trebuie sa participe un agent. Pozitia in vector indica time-slotul in care are loc meeting-ul cu valoarea de pe pozitia respectiva.

Asadar de ex. pentru un A0->M1,M3,M5, si l=5; o reprezentare pentru acest agent poate fi : -1,-1,3,5,1; acest lucru inseamna ca M1 are loc in timeslot-ul 5, M3 are loc in timeslot-ul 3, iar M5 are loc in timeslot-ul 4.

Constrangeri:

Metoda addConstrainForTimeslot are ca parametri de intrare lista de agenti si un meeting m si se asigura ca meeting-ul m va fi in acelasi timeslot pentru fiecare agent care participa la meeting-ul m.

- Distanţa dintre 2 meeting-uri consecutive < intervalul de timp dintre ele;
- Sincronizarea calendarelor: pentru toţi agenţii care participa la un anumit meeting, meeting-ul trebuie să aibă loc la acelaşi interval de timp din calendarul lor;

Pentru strategia 1 si stategia 2 s-a folosit libraria Choco Solver care este dedicata programarii ce satisfac un set de constrangeri.

Strategia 3:

Aceasta varianta se bazeaza pe matricea din care reiese pentru fiecare agent, meeting-urile la care trebuie sa participe acestia, la fel ca si in strategia 2. Insa aici avem un vector de dimensiune m-nr de meeting-uri totale, iar apoi pornim de la primul agent si incepem sa programam meeting-urile la care trebuie sa participe primul agent, dupa care continuam sa parcurgem toate listele de meeting-uri ale tuturor agentilor si modificam vectorul in functie de constrangeri. In acest vector pozitia i reprezinta meeting-ul i, iar valoarea de pe pozitia i reprezinta timeslotul in care este alocat meeting-ul respectiv.

Constrangeri:

- meeting-urile nu trebuie sa se suprapuna:
  |time(ti)−time(tj)|−duration i > =TravellingTime(location(mi),location(mj))
  Unde:
  -ti,tj sunt timesloturile programate pentru meeting i, resectiv j;
  -duration i este durata meeting-ului i;
  -mi, mj sunt meetingurile i si j;
- meeting-ul mi care este comun pentru mai multi agenti, acesta va avea acelasi timeslot pentru fiecare agent;

Analiza timpilor de executie pentru fiecare strategie folosita, aplicata pe mai multe instante de problema.

Am rulat algoritmii pe urmatoarele instante (22):

Nr. of meetings X Nr. of agents X Nr. of timeslots

5 X 3 X 8	5 X 3 X 16	5 X 3 X 24	5 X 3 X 32	10 X 5 X 8	10 X 5 X 16	10 X 5 X 24	10 X 5 X 32	20 X 10 X 8	20 X 10 X 16	20 X 10 X 32	30 X 15 X 32	30 X 15 X 40	30 X 15 X 50	30 X 15 X 68	30 X 15 X 78	30 X 15 X 88	40 X 10 X 48	40 X 10 X 58	40 X 10 X 68	40 X 10 X 78	40 X 10 X 88

Un timp de executie pentru fiecare instanta poate fi gasit in fisierul Statistics.xlsx;
Instantele sunt citite din fisiere, ce se afla in surse, in fisierul “resources”.
Solutiile generate de strategii sunt afisate in consola, iar timpii de executie sunt scrisi si in fisierul “tests.csv”. La fiecare rulare a algoritmilor, exista o diferenta neglijabila intre timpii de executie.
Fiecare strategie genereaza o solutie diferita, care satisface constrangerile specificate.

Un exemplu de solutii generate pentru o instanta metionata mai sus cu m=5, n=3, l=32 este urmatoarea:

S1:
Meeting: 0 Timeslot: 2
Meeting: 1 Timeslot: 0
Meeting: 2 Timeslot: 5
Meeting: 3 Timeslot: 3
Meeting: 4 Timeslot: 8

S2:
Meeting: 0 Timeslot: 17
Meeting: 1 Timeslot: 6
Meeting: 2 Timeslot: 20
Meeting: 3 Timeslot: 4
Meeting: 4 Timeslot: 21

S3:
Meeting: 0 Timeslot: 0
Meeting: 1 Timeslot: 1
Meeting: 2 Timeslot: 5
Meeting: 3 Timeslot: 6
Meeting: 4 Timeslot: 7

Aceasta este o medie totala a timpului de executie.

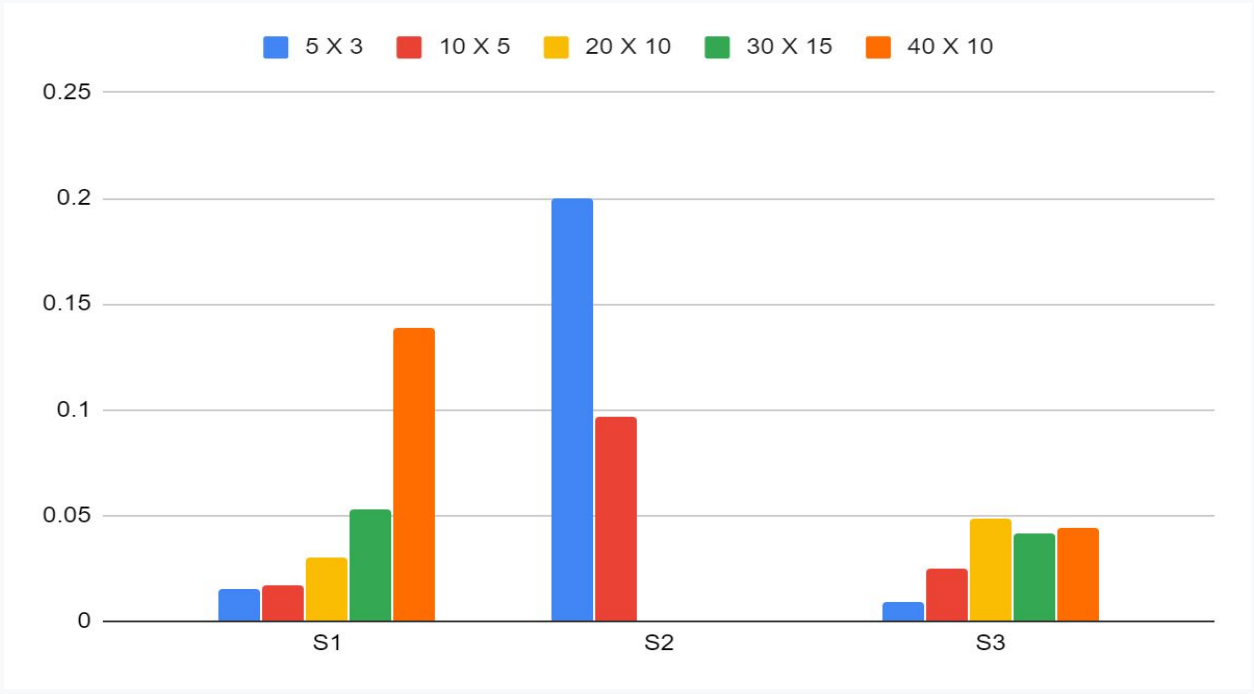
Strategy	Average time
S1	0.05602535473
S2	irelevant*
S3	0.03425

*aceasta observatie se datoreaza faptului ca S2 este are un timp de executie mai mare de 120 sec. pentru instantele de nr. of meetings >=20 && nr.of agents>=10.

Mai jos este timpul mediu pentru instantele specificate.

Nr. of meetings X Nr. of agents

Strategy	5 X 3	10 X 5	20 X 10	30 X 15	40 X 10
S1	0.015295125	0.0168164	0.03014716667	0.05303551667	0.1390914208
S2	0.200367175	0.0964249275	TIMEOUT	TIMEOUT	TIMEOUT
S3	0.009125	0.02475	0.04833333333	0.04166666667	0.0446



Din grafic observam, ca S2 este algoritmul care merge cel mai greu, dar acest lucru este si de inteles deoarece in aceasta strategie se folosesc vectori de dimensiuni mari, pentru fiecare agent fiind alocat un vector de dimensiune l-nr. de timeslot-uri. Cu cat nr. de agenti si nr. de sloturi sunt mai mari, cu atat algoritmul gaseste mai greu o solutie care sa satisfaca constrangerile specificate.

Strategiile S1 si S3 se comporta mai bine pe diferite cazuri, de aici deducem ca durata timpului de executie depinde de dimensiunea unor parametri diferiti de intrare. S1 se comporta mai bine atunci cand nr. de meeting-uri, nr. de agenti si nr. de timeslot-uri este mai mic, insa atunci cand este vorba de dimensiuni mai mari, S2 este mai rapid decat S1 in gasirea unei solutii corecte pentru instanta data.