INTELLIGENZA ARTIFICIALE

PROGETTO CSP

Andrea Leonardo - 6292833

TESTO:

Calendario degli esami
In questo esercizio si costruisce un modello per risolvere un semplice problema di soddisfacimento di vincoli ispirato alla generazione di un calendario per gli esami universitari. Si assume che un certo corso di laurea abbia interamente a disposizione A aule per fare esami (solo prove scritte), ognuna con capienza di Ci posti, per i = 1 . . . A e disponibile ogni giorno dalle 8 alle 12 e dalle 14 alle 18. Ci sono in totale E esami che devono essere tenuti in un periodo di G giorni. Ogni prova scritta ha una durata assegnata di De ore, per e = 1 . . . E, con De ∈ {2, 4} (per semplicità si assuma che la durata includa il tempo necessario a far entrare ed uscire gli studenti). Ci sono S studenti iscritti e ciascuno di essi si è prenotato per sostenere nella sessione un sottoinsieme degli E esami. Il problema consiste nel determinare un calendario (indicando per ogni esame il giorno, l'ora e la data di svolgimento) che rispetti le capienze delle aule e tenga conto delle prenotazioni degli studenti in modo tale che nessuno studente debba partecipare a più di una prova nello stesso giorno. Inoltre, nessuna prova può essere spezzata a cavallo della pausa pranzo. Il modello deve essere abbastanza generico da permettere di risolvere una qualsiasi istanza. Si sviluppi il modello in un ambiente a scelta tra MiniZinc e Numberjack e se ne mostri il funzionamento (con un output intelligibile) su almeno tre istanze distinte.

SVOLGIMENTO:

Per questo problema ho deciso di utilizzare MiniZinc. Lo scopo è quello di generare una serie di informazioni riguardanti giorno, ora e aule di svolgimento di un numero definito a priori di esami. Ogni esame dispone di un unico appello, ma può svolgersi contemporaneamente in più aule. Di seguito riporto la parte essenziale del codice e successivamente ne descriverò le caratteristiche.

```
int: A;
int: E;
int: S;
int: G;

array[1..A] of int: postiAula;
array[1..E] of 2..4: durataEsame;
array[1..S, 1..E] of 0..1: tabEsami;
array[1..E] of var int: iscrittiEsame;

constraint forall(e in 1..E)(iscrittiEsame[e] = sum(s in 1..S)(tabEsami[s, e]));

array[1..E, 1..A] of var 0..1: auleEsame;
array[1..E] of var 1..G: giornoEsame;
array[1..E] of var 8..16: oraInizio;
array[1..E] of var 10..18: oraFine;

constraint forall(e in 1..E)(sum(i in 1..A where auleEsame[e,i] == 1)(postiAula[i]) >=
iscrittiEsame[e]);  %vincoloAule
constraint forall(e in 1..E)(oraFine[e] - oraInizio[e] == durataEsame[e]); %vincoloDurata
```

```
constraint forall(e,f in 1..E where e!=f)(if(giornoEsame[e] == giornoEsame[f] /\
((oraInizio[e]      < oraFine[f] /\ oraInizio[e] >= oraInizio[f]) \/ (oraInizio[f] <
oraFine[e] /\ oraInizio[f] >= oraInizio[e]))) then forall(a in 1..A)(auleEsame[e, a] +
auleEsame[f, a] < 2)endif); %vincoloIncastro
constraint forall(e in 1..E)(oraInizio[e] != 11 /\ oraInizio[e] != 12 /\ oraInizio[e] !=
13); %vincoloOraInizio
constraint forall(e in 1..E)(if(durataEsame[e] == 4)then oraInizio[e] != 10 /\ oraInizio[e]
!= 9 endif); %vincoloOraInizio2
constraint forall(e,f in 1..E, s in 1..S where e!=f)(if(tabEsami[s, e] + tabEsami[s, f] ==
2) then giornoEsame[e] != giornoEsame[f]endif); %vincoloDoppioEsame

solve satisfy ;
```

Dati e Variabili
Sono necessarie una serie di variabili impostate in precedenza riguardanti la caratterizzazione stessa del calendario e dell'ateneo di cui si vuole fare il timetabling degli esami. Le variabili sono le seguenti:

- A -> numero delle aule
- E -> numero degli esami
- S -> numero degli studenti
- G -> numero dei giorni della sessione
- postiAula[*i*] -> numero di posti disponibili dell'aula *i*
- durataEsame[*i*] -> durata in ore (2 o 4) dell'esame *i*
- tabEsami[*i, j*] -> 1 se lo studente *i* è iscritto all'esame *j* ,altrimenti 0

Per poter svolgere l'esercizio ho costruito dei vettori di variabili rappresentati ognuno un attributo di un esame. La loro assegnazione determina chiaramente il corretto svolgimento del problema:

- auleEsame[*i, j*] -> 1 se l'esame *i* occupa l'aula *j*, altrimenti 0
- giornoEsame[*i*] -> giorno di svolgimento dell'esame *i*
- oraInizio[*i*] -> ora di inizio dell'esame *i*
- oraFine[*i*] -> ora di terminazione dell'esame *i*

Vincoli
I vincoli sono i seguenti:

- vincoloAule = controlla che i posti prenotati siano superiori agli studenti iscritti all'esame
- vincoloDurata = controlla che la durata di un esame (oraFine-oraInizio) sia giusta
- vincoloIncastro = verifica che 2 esami fissati lo stesso giorno e a orari non disgiunti, non abbiano aule prenotate in comune
- vincoloOraInizio/2 = controllano che un esame non sia prenotato in un orario che includa la pausa pranzo (12:00 – 14:00)
- vincoloDoppioEsame = verifica che ogni studente non partecipi a più di un esame al giorno

OUTPUT:

In un linguaggio dichiarativo come MiniZinc l'utilizzo di for-loop per controllare il flusso della struttura dell'output non è possibile. E' però possibile creare manualmente un output facilmente comprensibile utilizzando un modello simile e/o amplificato di quello che mostrerò in seguito.

```
output ["|Esame | Aula1 | Aula2 | Aula3 | Aula4 | Aula5 | Giorno | Orario", "\n",
"| 1 | ", show(auleEsame[1,1]), " | ", show(auleEsame[1,2]), " | ", show(auleEsame[1,3]), "
| ", show(auleEsame[1,4]), " | ", show(auleEsame[1,5]), " | ", show(giornoEsame[1]), " | ",
show(oraInizio[1]), ":00 - ", show(oraFine[1]), ":00 \n",
"| 2 | ", show(auleEsame[2,1]), " | ", show(auleEsame[2,2]), " | ", show(auleEsame[2,3]), "
| ", show(auleEsame[2,4]), " | ", show(auleEsame[2,5]), " | ", show(giornoEsame[2]), " | ",
show(oraInizio[2]), ":00 - ", show(oraFine[2]), ":00 \n",
"| 3 | ", show(auleEsame[3,1]), " | ", show(auleEsame[3,2]), " | ", show(auleEsame[3,3]), "
| ", show(auleEsame[3,4]), " | ", show(auleEsame[3,5]), " | ", show(giornoEsame[3]), " | ",
show(oraInizio[3]), ":00 - ", show(oraFine[3]), ":00 \n",
"| 4 | ", show(auleEsame[4,1]), " | ", show(auleEsame[4,2]), " | ", show(auleEsame[4,3]), "
| ", show(auleEsame[4,4]), " | ", show(auleEsame[4,5]), " | ", show(giornoEsame[4]), " | ",
show(oraInizio[4]), ":00 - ", show(oraFine[4]), ":00 \n",
"| 5 | ", show(auleEsame[5,1]), " | ", show(auleEsame[5,2]), " | ", show(auleEsame[5,3]), "
| ", show(auleEsame[5,4]), " | ", show(auleEsame[5,5]), " | ", show(giornoEsame[5]), " | ",
show(oraInizio[5]), ":00 - ", show(oraFine[5]), ":00 \n",];
```

Questo è un esempio in caso di 5 esami e 5 aule. L’output generato sarà il seguente (con una serie di dati specifici):

```
|Esame |  Aula1  |  Aula2  |  Aula3  |  Aula4  |  Aula5  | Giorno | Orario
| 1    |    1    |    1    |    1    |    1    |    1    |   4    | 8:00 - 10:00
| 2    |    1    |    1    |    1    |    1    |    1    |   1    | 8:00 - 10:00
| 3    |    0    |    1    |    0    |    1    |    1    |   2    | 8:00 - 12:00
| 4    |    0    |    1    |    0    |    0    |    0    |   3    | 8:00 - 12:00
| 5    |    1    |    0    |    1    |    0    |    0    |   2    | 8:00 - 12:00
```

In ogni caso l’output generato automaticamente da MiniZinc risulta comunque facilmente comprensibile e può essere comunque utilizzato. Seguendo l’esempio precedente si otterrebbe il seguente output:

```
iscrittiEsame = array1d(1..5, [80, 80, 40, 20, 40]);
auleEsame = array2d(1..5, 1..5, [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0]);
giornoEsame = array1d(1..5, [4, 1, 2, 3, 2]);
oraInizio = array1d(1..5, [8, 8, 8, 8, 8]);
oraFine = array1d(1..5, [10, 10, 12, 12, 12]);
```

ESEMPI E RISULTATI:

1° ESEMPIO:

dati:

```
int: A = 5;
int: E = 5;
int: S = 100;
int: G = 3;

array[1..A] of int: postiAula = [10, 20, 30, 10 ,15];
array[1..E] of 2..4: durataEsame = [2, 2, 4, 4, 4];
array[1..S, 1..E] of 0..1: tabEsami =
array2d(1..S,1..E,[0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,
1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,
0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0,0,1,1,1,0,0,0]);
```

Il primo esempio, di tipo più generico, serve a verificare il corretto funzionamento dei vincoli. L’assegnazione delle prenotazioni fa in modo che sia possibile la coincidenza di 2 esami lo stesso giorno, cosa che non sarebbe praticamente possibile se anche solo 1 studente fosse iscritto a tutti gli

esami. Per questo il test consta di 5 esami e 3 giorni, in modo da constatare il funzionamento di tutti i vincoli. L'output generato è il seguente:

Esame	Aula1	Aula2	Aula3	Aula4	Aula5	Giorno	Orario
1	1	1	1	1	1	3	14:00 - 16:00
2	1	1	1	1	1	1	8:00 - 10:00
3	0	1	0	1	1	2	8:00 - 12:00
4	0	1	0	0	0	3	8:00 - 12:00
5	1	0	1	0	0	2	8:00 - 12:00

Come si può notare nessun esame si incastra con quegli altri ma sono comunque presenti più esami nello stesso giorno (combinazione decisa grazie alle particolari prenotazioni degli studenti.

2° ESEMPIO:

dati:
```
int: A = 5;
int: E = 10;
int: S = 200;
int: G = 4;


array[1..A] of int: postiAula = [10, 20, 30, 10 ,15];
array[1..E] of 2..4: durataEsame = [2,2,4,2,2,4,4,2,4,4];
array[1..S, 1..E] of 0..1: tabEsami =
array2d(1..S,1..E,[1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,
1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,...,1,1,1,0,0,0,0,0,0,0]);
```

Questo secondo esempio è concentrato sulla verifica del vincolo del doppio esame per uno studente. Sono stati creati appositamente 200 studenti di solo 3 categorie differenti:

- studente del primo anno iscritto ai primi 3 esami (1,1,1,0,0,0,0,0,0,0)
- studente del secondo anno iscritto agli esami 4..7 (0,0,0,1,1,1,1,0,0,0)
- studente del terzo anno iscritto agli ultimi 3 esami (0,0,0,0,0,0,0,1,1,1)

Questo permette alle 3 tipologie diverse di esame di sovrapporsi tranquillamente negli stessi giorni poiché parteciperebbero sicuramente studenti diversi. Aumentando il numero di studenti, inoltre si aumenta notevolmente l'occupazione delle aule, ma questo permette comunque al programma di distribuire i 10 esami in soli 4 giorni. L'output generato è il seguente:

Esame	Aula1	Aula2	Aula3	Aula4	Aula5	Giorno	Orario
1	1	1	1	1	0	3	16:00 - 18:00
2	0	1	1	1	1	2	16:00 - 18:00
3	1	1	1	0	1	1	14:00 - 18:00
4	1	1	1	1	0	4	14:00 - 16:00
5	0	1	1	1	1	2	14:00 - 16:00
6	1	1	1	0	1	1	8:00 - 12:00
7	1	1	1	1	0	3	8:00 - 12:00
8	0	1	1	1	1	3	14:00 - 16:00
9	1	1	1	0	1	4	8:00 - 12:00
10	1	1	1	1	0	2	8:00 - 12:00

3° ESEMPIO:

dati:
```
int: A = 5;
int: E = 8;
int: S = 50;
int: G = 8;

array[1..A] of int: postiAula = [10, 20, 30, 10 ,15];
array[1..E] of 2..4: durataEsame = [2,2,4,2,2,4,4,2];
array[1..S, 1..E] of 0..1: tabEsami =
array2d(1..S,1..E,[1,1,1,1,1,1,1,1,1,1,0,0,1,1,0,0,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,
                   1,1,1,…,1,1,1,1,1,1,1,1,1,1,0,0,1,1,0,0]);
```

In questo esempio possiamo invece verificare che la presenza di studenti iscritti a molti esami (ad esempio tutti come si può vedere dal primo studente dei dati), comporti un necessario bisogno di tanti giorni quanti il numero di esami totali. Come si può vedere , nonostante siano diminuiti anche il numero di studenti rispetto agli esempi precedenti, se mettiamo G=8 l'output sarà il seguente:

Esame	Aula1	Aula2	Aula3	Aula4	Aula5	Giorno	Orario
1	1	1	0	0	1	8	8:00 - 10:00
2	0	0	1	1	0	7	8:00 - 10:00
3	1	1	0	0	0	6	8:00 - 12:00
4	0	0	0	1	1	5	8:00 - 10:00
5	1	0	1	0	0	4	8:00 - 10:00
6	0	1	0	1	1	3	8:00 - 12:00
7	0	0	1	0	0	2	8:00 - 12:00
8	1	1	0	0	0	1	8:00 - 10:00

Come si può vedere ogni esame è collocato di conseguenza in un giorno differente, e per praticità sono tutti alle 8:00, poiché sono possono infastidirsi tra loro. Se proviamo a mettere 7 giorni il risultato sarà inevitabilmente il seguente:

```
Compiling Exams_Timetabling.mzn
Running Exams_Timetabling.mzn
=====UNSATISFIABLE=====
Finished in 5s 746msec
```