Pianificazione dell'orario delle lezioni come CSP

Fabio Luccioletti

Febbraio 2021

1 Introduzione

L'obiettivo di questo elaborato è quello di risolvere il problema dell'*orario delle lezioni*, presentando la possibile programmazione settimanale di una serie di corsi che si tengono nello stesso semestre. Perché la programmazione sia valida è necessario che siano rispettati una serie di **vincoli**: un problema di questo tipo rientra nella categoria dei **Constraint Satisfaction Problems**, ed un linguaggio che si presta alla modellazione e alla risoluzione di tali problemi è **Minizinc**.

1.1 Descrizione dell'ambiente di sviluppo

Le specifiche *hardware* di interesse del dispositivo dove sono stati conseguiti i risultati mostrati in seguito sono:

- **Sistema Operativo:** MacOS Big Sur 11.2
- **Processore:** 2,6 GHz Intel Core i7 6 core
- **Memoria principale:** 16 GB 2400 MHz DDR4

Il *solver* utilizzato da MiniZinc per la risoluzione del problema è stato Gecode.

2 Modellazione del problema

Il problema di soddisfacimento dei vincoli per essere ben definito (e riproducibile) deve descrivere tre componenti:

- i parametri utilizzati
- le variabili e il loro dominio
- i vincoli che devono essere rispettati

2.1 Parametri

Perché il problema sia sensato dobbiamo pensare di poterlo adattare a diverse situazioni. Potremmo decidere organizzare gli orari di pochi corsi di studi che condividono le stesse aule, oppure dell'intero corso di ingengeria, dove molti professori tengono vari insegnamenti (anche in corsi di studio differenti), con un gran numero di aule a disposizione.
Il potere della *scalabilità* e dell'*adattamento* è espresso attraverso l'uso dei parametri.
Oltre a dover definire i **giorni** in cui è possibile insegnare in un certo ateneo, è importante sapere anche quante **ore** giornalmente le aule sono aperte ed ovviamente quante **aule** sono disponibili.
Ogni problema è caratterizzato da un set degli **insegnamenti** e uno dei **corsi**, queste due parametri sono espressi attraverso delle *enumerazioni*, che permettono di associare un nome mnemonico ad ogni parametro del set (è più facile riferirsi all'insegnamento *Intelligenza Artificiale* piuttosto che all'insegnamento *1*). Inoltre è fornito il numero di **professori**, le **ore giornaliere massime** di insegnamento previste per i docenti, le **ore massime e minime giornaliere** previste invece per gli insegnamenti ed un *booleano* che indica se vogliamo che le ore di ogni materia in un'aula siano una di seguito all'altra.
Infine tre liste di interi permettono di associare ogni insegnamento al suo corso, al suo professore, e alle ore settimanali previste dal regolamento.

2.2 Variabili

Per risolvere efficientemente il problema dell'orario delle lezioni, ad ogni insegnamento, per ogni aula ed ogni giorno, è stato associato un **set di orari**, che rappresenta la sua programmazione.
Ad esempio se all'insegnamento *Intelligenza Artificiale*, nel giorno 1 e nella classe 2 è associato il set $\{3,4,5\}$, significa che in quella classe il lunedì alla terza, quarta e quinta ora ci saranno le lezioni di Intelligenza Artificiale.
Ogni elemento del set ha un suo *dominio*, ovvero i valori permessi dalla variabile di decisione (in questo caso il *range* da 1 al numero di ore giornaliere).
La struttura dati che contiene questi set è rappresentata da un array tridimensionale dove gli indici sono il giorno, l'aula e l'insegnamento di interesse.

2.3 Vincoli

I vincoli definiscono, nel loro insieme, i valori che le variabili possono assumere perché l'assegnamento sia consistente.
In particolare è stato imposto che il numero di ore di ogni insegnamento fosse quello prefissato (tramite un controllo sulle cardinalità dei set), che in uno stesso momento non fosse presente la stessa lezione in due aule differenti, così come lezioni tenute dallo stesso professore o dello stesso corso di laurea ed ovviamente che ogni aula ad ogni ora ospitasse al massimo una lezione.
Questi ultimi vincoli sono stati implementati con l'ausilio del **global constraint**

all_disjoint, che obbliga i set in una lista di essere disgiunti a due a due.
Inoltre, è stato evitato che un professore potesse avere lezione più di H ore ogni giorno, e che ogni insegnamento, se presente in un dato giorno, figurasse per meno di *t1* o più di *t2* ore (Figura 1), con H, t1, t2 parametri.

```
% there can't be less than t1 or more than t2 hours
    of the same teaching in a day (or there can be 0 hours)

constraint forall(day in 1..days, room in 1..rooms, teaching in teachings)(
    card(teaching_schedule[day, room, teaching]) = 0 \/
    (card(teaching_schedule[day, room, teaching]) >= t1 /\
    card(teaching_schedule[day, room, teaching]) <= t2)
);
```

Figure 1: Il vincolo sul numero massimo e minimo di ore di ogni insegnamento giornalmente

3 Risultati

Il modello è stato testato con due diversi set di parametri, uno per simulare la programmazione delle lezioni nel caso di una piccola scuola (2 aule, 3 corsi, 15 insegnamenti, 6 professori) e l'altro considerando invece un ateneo più frequentato (8 aule, 10 corsi, 50 insegnamenti, 25 professori).
L'*output* del programma restituisce l'orario di ogni aula per ogni giorno, associando alle ore la materia insegnata o alternativamente indicando che l'aula è libera (*FREE*).
Di seguito il risultato ottenuto dalla risoluzione del problema più piccolo (per brevità è presentata la programmazione solo dei primi due giorni), seguito dalle statistiche di risoluzione:

```
DAY 1

ROOM 1
HOUR 1: Statistica
HOUR 2: Statistica
HOUR 3: Statistica
HOUR 4: Elaborazione_dei_Segnali
HOUR 5: Elaborazione_dei_Segnali
HOUR 6: Sistemi_di_Controllo
HOUR 7: Sistemi_di_Controllo
HOUR 8: Sistemi_di_Controllo
HOUR 9: FREE
```

```
ROOM 2
HOUR 1: Antenne_e_Propagazione
HOUR 2: Antenne_e_Propagazione
HOUR 3: Antenne_e_Propagazione
HOUR 4: Intelligenza_Artificiale
HOUR 5: Intelligenza_Artificiale
HOUR 6: Applicazioni_di_Matematica
HOUR 7: Applicazioni_di_Matematica
HOUR 8: Applicazioni_di_Matematica
HOUR 9: FREE

DAY 2

ROOM 1
HOUR 1: Meccanica_Razionale
HOUR 2: Meccanica_Razionale
HOUR 3: Elettronica_Applicata
HOUR 4: Elettronica_Applicata
HOUR 5: Elaborazione_dei_Segnali
HOUR 6: Elaborazione_dei_Segnali
HOUR 7: Ingegneria_del_Software
HOUR 8: Ingegneria_del_Software
HOUR 9: Ingegneria_del_Software

ROOM 2
HOUR 1: Basi_di_dati
HOUR 2: Basi_di_dati
HOUR 3: Basi_di_dati
HOUR 4: Matematica_Discreta
HOUR 5: Matematica_Discreta
HOUR 6: Elettronica_Applicata
HOUR 7: Elettronica_Applicata
HOUR 8: Elettronica_Applicata
HOUR 9: FREE
----------
%%%mzn-stat: initTime=0.05606
%%%mzn-stat: solveTime=0.011818
%%%mzn-stat: solutions=1
%%%mzn-stat: variables=7050
%%%mzn-stat: propagators=8780
%%%mzn-stat: propagations=121545
%%%mzn-stat: nodes=393
%%%mzn-stat: failures=172
%%%mzn-stat: restarts=0
%%%mzn-stat: peakDepth=57
%%%mzn-stat-end
```