RAGIONAMENTO AUTOMATICO

Due modelli per la generazione automatica degli orari di una università durante una pandemia/epidemia

Name	Federico Bulzoni
Student No.	142242
Department	Informatica
Email	bulzoni.federico@spes.uniud.it
Date	27/07/2020

1. Introduzione

Il problema del *timetabling* per una università, ossia di determinare un orario per le lezioni settimanali all'interno di un'università è un problema classico nell'informatica teorica, e approcci che rendono automatica la generazione di un tale orario vengono esplorati sin dagli anni sessanta dello scorso secolo (Gotlieb, 1963). Nella sua formulazione più comune il problema richiede dati un certo numero di aule, un certo numero di professori, un certo numero di classi ed una finestra temporale, di trovare un assegnamento per la finestra temporale considerata delle classi e dei professori nelle aule che soddisfi un insieme di vincoli. I vincoli possono essere suddivisi in *vincoli forti* e *vincoli deboli*. Considerato un possibile orario settimanale, un vincolo forte è tale per cui l'orario considerato è una soluzione valida al problema se e solo se è tale per cui tale vincolo viene soddisfatto; un esempio di vincolo forte è il fatto che non è possibile che una stessa classe svolga contemporaneamente lezione in due aule diverse. Dall'altra parte un vincolo debole è tale per cui l'orario considerato è valido anche se non soddisfa il vincolo debole, tuttavia dati due orari settimanali validi di cui uno soddisfa il vincolo debole e l'altro no, viene giudicato come *soluzione migliore* tra le due l'orario che soddisfa il vincolo debole; un esempio di vincolo debole può essere che considerando una classe, il numero di ore di lezione che svolge durante i diversi giorni della settimana dovrebbe essere equilibrato.

In questo report andremo ad analizzare una versione modificata di tale problema, nella quale si prendono in considerazione nella generazione degli orari le necessità comportate dalla convivenza con una pandemia in corso, quali ad esempio la necessità di sanificare un aula prima di ogni cambio di classe. Per tale problema vengono proposte due soluzioni distinte, una nella quale il problema viene codificato in `MiniZinc` (Reference needed) e un altra nella quale il problema viene codificato in `ASP` (Reference needed). Le soluzioni proposte oltre a differenziarsi per il linguaggio utilizzato, si differenziano anche per l'approccio seguito, nella codifica *MiniZinc* viene fatto uso di un automa deterministico a stati finiti (**DFA**) per risolvere buona parte dei vincoli forti emersi nella modellazione del problema, mentre nella codifica *ASP* tali vincoli vengono modellati con un approccio puramente dichiarativo.

2. Specifica del problema

Si richiede di organizzare l'orario per una università di piccole dimensioni tenendo conto delle misure di contenimento in atto per una pandemia.

2.1 Dati in input

Le **aule** a disposizione sono suddivise in $G = 4$ **gruppi** distinti, questa assunzione deriva dal fatto che le aule si trovano tutte su di un corridoio con al centro delle scale, abbiamo quindi una suddivisione delle aule in base al lato del corridoio ed in base al lato in cui si trovano rispetto alle scale. Ogni gruppo, è formato dallo stesso numero di aule $K \in \mathbb{N}$, pertanto il numero totale di aule a disposizione è $G*K$. Ad ogni aula $r \in [1, G*K]$ è associata una **capacità** $cap \in [30, 60]$ che indica il numero di studenti che l'aula può contenere. Vi sono $N \in \mathbb{N}$ **coorti** di studenti da allocare. Ad ogni coorte $c \in [1, N]$ è associato il **numero di studenti** che vi sono immatricolati $nStud \in [50, 300]$, e l'**anno** di carriera $y \in [1, 3]$ corrispondente, vengono considerate solamente lauree triennali.
Le coorti sono suddivise in **dipartimenti**, sia $D \in \mathbb{N}$ il numero di dipartimenti distinti all'interno dell'università. Ad ogni coorte $c \in [1, N]$ è associato il dipartimento $dep \in [1, D]$ a cui appartiene. Infine, ad ogni coorte $c \in [1, N]$ è associato il numero di ore a settimana $reqT \in [40, 60]$ che richiede.

2.2 Requisiti

Sapendo che l'orario di apertura dell'università è dalle 8:00 alle 19:00 e lavorando con una granularità di 30 minuti, si richiede di organizzare l'orario delle lezioni rispettando i seguenti vincoli forti:

- Gli slot delle lezioni sono di 2 ore,
- ogni volta che una coorte lascia un aula c'è bisogno di una sanificazione,
- gli slot delle sanificazioni sono di 1 ora,
- per ogni istante temporale, in un gruppo di aule possono essere allocate unicamente coorti dello stesso dipartimento e sanificazioni,
- tutti gli studenti delle coorti al primo anno devono avere almeno una lezione in presenza a settimana,

- la percentuale di ore che non vengono allocate alle coorti degli anni successivi al primo rispetto a quelle richieste deve essere ben bilanciata tra i dipartimenti, in particolare si assume che le percentuali tra i diversi dipartimenti siano bilanciate con uno scarto del 10%.

Si richiede di **minimizzare la non occupazione delle aule**.

2.3 Assunzioni

Si assume che i giorni di apertura siano 5: dal Lunedì al Venerdì e che alla fine di ogni giornata, immediatamente dopo la chiusura dell'università ci sia una sanificazione generale di tutte le aule; questa assunzione implica che in un orario ottimale non ha senso avere una sanificazione all'orario di apertura (8:00) o all'orario di chiusura (19:00). Riguardo alla richiesta di minimizzare la non occupazione delle aule, si assume che non ci siano mai istanti temporali in cui un aula è vuota, al massimo può ritenersi "non occupata" quando al suo interno viene svolta una sanificazione, pertanto tale richiesta si traduce in *"minimizzare il numero di sanificazioni settimanali"*. Si assume che non porti alcun vantaggio effettuare una sanificazione in una data aula se tra l'istante prima della sanificazione e quello successivo non c'è un cambio di coorte all'interno dell'aula, pertanto questa eventualità viene impedita.

2.4 Funzione di costo

Oltre a minimizzare il numero di sanificazioni settimanali, si è scelto di dare peso anche al *numero di coorti che non riescono ad avere assegnate il numero di ore settimanali richieste* e alla *percentuale di ore mancanti rispetto alle ore settimanali richieste*.
Sia K il numero di sanificazioni settimanali, Z il numero di coorti che non riescono ad avere assegnate il numero di ore richieste, e $[P_1, \ldots, P_N]$ la lista di percentuali di ore mancanti rispetto a quelle richieste per ogni coorte $i \in [1, N]$.
La funzione di costo considerata è funzione di K, Z e $[P_1, \ldots, P_N]$, in particolare si assumono le seguenti priorità: $3@Z$, $2@[P_1, \ldots, P_N]$ e $1@K$ dove con il numero intero alla sinistra della @ indichiamo la priorità associata al valore alla destra della @. Questa scelta viene giustificata, da una maggiore attenzione che si è voluta porre sul rendere l'orario più equo possibile per tutti a discapito di un possibile maggior numero di sanificazioni.

3. Scelte comuni alle due soluzioni

Considerando che l'orario di apertura dell'università è dalle 8:00 alle 19:00 e che lavoriamo con una granularità di 30 minuti, la giornata universitaria è stata suddivisa in 22 unità di tempo, dove ogni unità di tempo corrisponde a 30 minuti reali. La soluzione al problema consiste in un **assegnamento** che ad ogni aula, ad ogni giorno considerato e ad ogni istante di tempo considerato associa un **occupante**, dove un occupante può essere una **coorte** o una **sanificazione**. Tale assegnamento viene chiamato `scheduling` in entrambe le soluzioni, il predicato $scheduling(r,d,t,o)$ se vero, indica che nell'aula $r \in [1, G*K]$, al giorno $d \in [1,5]$, all'istante $t \in [1,22]$ è presente l'occupante $o \in \{0\} \cup [1,N]$. La sanificazione viene identificata con il numero intero 0, mentre la coorte $i \in [1,N]$ è identificata con l'intero i. Data un aula $r \in [1, G*K]$, il gruppo $g \in [0, G-1]$ a cui appartiene è tale per cui $r\%G = g$, dove con % in questo caso indichiamo l'operazione modulo. Riguardo al vincolo sulle coorti al primo anno, per cui si richiede che tutti gli studenti abbiano almeno una lezione in presenza a settimana, data una coorte $i \in [1,N]$ il numero di studenti che ha almeno una lezione in presenza a settimana è stato modellato come la somma delle capacità delle aule $r \in [1, G*K]$ in cui la coorte i svolge lezione, chiaramente se la coorte i viene assegnata più volte all'aula r, la capacità di r verrà considerata più volte nel calcolo.
Nei piani iniziali, l'idea era quella di introdurre come ulteriore vincolo debole il fatto che il maggior numero possibile di studenti per ogni coorte vada a lezione in presenza, tuttavia il calcolo del numero di studenti con lezioni in presenza per ogni coorte è estremamente dispendioso, soprattutto nella codifica ASP in cui assegnare il risultato di un'operazione aggregata ad un predicato, per poi effettuarci confronti sopra fa esplodere la complessità; per questi motivi l'idea è stata abbandonata in entrambe le codifiche.
Il tempo assegnato ad una coorte, così come quello assegnato alle operazioni di sanificazione, non è altro che il conteggio delle volte in cui la coorte (o sanificazione) viene assegnata ad una tripla $(r,d,t) \in [1, G*K] \times [1,5] \times [1,22]$, dove r è un'aula, d è un giorno della settimana e t è un istante temporale.
Il vincolo:

Vincolo aule-dipartimenti

Per ogni istante temporale, in un gruppo di aule possono essere allocate unicamente coorti dello stesso dipartimento e sanificazioni.

È stato modellato in entrambe le soluzioni tramite il predicato al primo ordine:

$$\forall r \forall dep \forall t (scheduling(r,d,t,c) \wedge department(c,dep) \rightarrow \\ \neg \exists r' (r' \neq r \wedge scheduling(r',d,t,c') \wedge r\%G = r'\%G \wedge department(c',dep') \wedge dep' \neq dep)) \tag{3.1}$$

dove $r, r' \in [1, G*K]$, $d \in [1,5]$, $t \in [1,22]$, $c, c' \in [1,N]$, $dep, dep' \in [1,D]$.

Il vincolo:

Vincolo coorti al primo anno - studenti

Tutti gli studenti delle coorti al primo anno devono avere almeno una lezione in presenza a settimana.

È stato modellato in entrambe le soluzioni tramite il predicato al primo ordine:

$$\forall c(year(c,1) \wedge satisfiedStudents(c,x) \wedge nStudents(c,y) \rightarrow x \geq y) \tag{3.2}$$

dove $c \in [1,N]$, $year(c,1)$ se e solo se la coorte c è al primo anno, $satisfiedStudents(c,x)$ è vero se e solo se il numero di studenti che hanno lezione in presenza della coorte c è uguale ad x ed x viene calcolato come specificato in precedenza, e $nStudents(c,y)$ è vero se e solo se il numero di studenti della coorte c è pari ad y.

Il vincolo:

Vincolo bilanciamento insoddisfazione dipartimenti

La percentuale di ore che non vengono allocate alle coorti degli anni successivi al primo rispetto a quelle richieste deve essere ben bilanciata tra i dipartimenti, in particolare si assume che le percentuali tra i diversi dipartimenti siano bilanciate con uno scarto del 10%.

è quello che ha creato più problemi nella fase di modellazione e anche di implementazione dati i calcoli non basilari che richiede. In particolare richiede di:

1. isolare i dipartimenti per cui esiste almeno una coorte non al primo anno di studi che vi appartiene,
2. calcolare il numero totale di unità di tempo richieste per ogni dipartimento considerando solamente le coorti non al primo anno,
3. calcolare il numero totale di unità di tempo assegnate per ogni dipartimento considerando solamente le coorti non al primo anno,
4. calcolare la percentuale di ore assegnate alle coorti al primo anno rispetto a quelle richieste (o equivalentemente non assegnate) per ognuno dei dipartimenti,
5. verificare che non esistano due dipartimenti diversi per cui il valore assoluto della differenza tra le loro due percentuali così calcolate superi il 10%.

Per questo vincolo non staremo a dare la formalizzazione in logica al primo ordine, ma è facilmente ricavabile dalla procedura appena descritta.

Per entrambe le soluzioni si è cercato di utilizzare il minor numero di vincoli possibile cercando al contempo di tenere intatta la semantica delle soluzioni, per questo motivo tutti gli altri vincoli che possono essere raggruppati come *vincoli riguardanti la regolarità* dello scheduling, sono stati condensati in entrambe le soluzioni in un unico vincolo che può essere chiamato **vincolo di regolarità**. Tale vincolo può essere descritto come:

Vincolo di regolarità

Non può essere che una sanificazione sia schedulata al primo o all'ultimo istante di una giornata, dato che a fine giornata avviene la sanificazione generale di tutte le aule. Una sanificazione dura 2 unità di tempo (1 ora) e viene effettuata **se e solo se** tra l'istante prima dell'inizio della sanificazione e l'istante successivo alla sanificazione all'interno della stanza interessata c'è stato un cambio di coorte. Uno slot di lezione dura 4 unità di tempo (2 ore).

La soluzione in `MiniZinc` e quella in `ASP` si differenziano principalmente per come il vincolo di regolarità è stato implementato.

4. La soluzione in MiniZinc

La soluzione in `MiniZinc` non ha creato grossi problemi durante la fase implementativa, anche se è passata attraverso diversi refactoring. La prima versione dell'implementazione in MiniZinc lavorava con una ricerca binaria, e il predicato *scheduling* era implementato come un'array di variabili booleane associate alle tuple (r,d,t,o) dove r è un'aula, d è un giorno, t è un istante temporale e o è una coorte o una sanificazione. Tale versione tuttavia si è rivelata generalmente meno performante della successiva versione che lavora con una ricerca intera ed in cui il predicato *scheduling* è implementato come un array di occupanti (sanificazione o una coorte), questo risultato non è stato una sorpresa dato che passare ad una versione di scheduling fatta in questo modo mi ha permesso di utilizzare constraint globali implementati nella libreria `globals.mzn` quali ad esempio `count` e `regular` in modo più semplice e ipotizzo efficiente.
Un altro refactoring importante attraversato da questa soluzione ha riguardato il DFA utilizzato per implementare il vincolo di regolarità, si rimanda alla prossima sezione per i dettagli.

4.1 Implementazione del vincolo di regolarità

Il vincolo di regolarità è stato implementato in modo naturale sfruttando il constraint globale `regular` offerto dalla libreria `globals.mzn`. Per implementare tale vincolo si è resa necessaria la progettazione di un automa deterministico a stati finiti,

Appendix

[*Report the config files of the software used (i.e. SU2 and the mesher). Also attach to this report an archive with the mesh files, solutions and the reference solution data (e.g. data points of a Cp plot ...)*]