Orario delle lezioni

Ankur Meozzi

March 23, 2019

1 Codice

- Il programma scritto per organizzare l'orario prende in ingresso i valori del numero di professori = p, aule = n, semestri = s, giorni di lezione in una settimana = d, ore in un giorno = ore, il numero di corsi = k. Inoltre prende in ingresso un array delle capacità delle aule = capAule, un'altro del numero si studenti per corso = numStud e un' array per il numero di ore per ogni corso = oreInsg. Si deve fornire anche una matrice di dimensione k*s per il numero di corsi per ogni semestre = numInseg e un'altra matrice di dimensioni s*p per indicare i 2 corsi tenuti da un professore = pInsg.
- Il codice oltre alla parte dichiarativa, ha anche una parte dove si creano diversi array per l'indicizzazione. Questi array ci saranno utili in seguito quando verranno utilizzati per i vincoli. La sintassi per creare questi indici non è di facile comprensione, ho creato un file indexGenrator per vedere cosa c'è in questi array. In alternativa gestire gli indci all'interno dei vincoli sarebbe risultato complicato e avrebbe reso la lettura molto difficile.
- sono stati espressi tutti i vincoli richiesti dal problema ed ho aggiunto un ulteriore vincolo che obbliga le ore di una materia in un giorno ad essere attaccate almeno 2 a 2, ma per la prima colonna non sempre viene rispettato tale vincolo ad esempio per l'istanza dei dati orario10E.dzn si ottiene:

m22	00000000	11000000	11100000	00000000	00000000
m23	30003330	00000000	00000000	00000000	00000000
.....					
m29	00000000	00333000	00000000	00000000	00000000
m30	20000222	00000000	00000000	00000000	00000000

Table 1: esecuzione con orario10E.dzn

2 Sviluppo del codice

- La tabella degli orari al inizio era un' array 3D, dove si mostrava una tabella dell'arario per ogni aula. Il vantaggio di tale scelta stava nel fatto che il dominio delle variabili era solo 0 o 1; cioè dei valori booleani che indicano la presenza di una materia a una certa ora, ma era brutto da vedersi e dispersivo. Infatti solo un elemento per colonna può essere a 1 mentre gli altri dovevano restare a zero. D'altro canto esprimere tale vincolo era semplice: bastava fare la *sum* delle colonne di ogni tabella e porla <= a 1, e cosi similmente per altri vincoli. Le variabile da gestire sono $m*t*n$ con m = materie, t = ore e n = numero di aule; all'aumentare di n aumentano le variabili da gestire

- La tabella 2D ha il vantaggio di esprimere l'orario di tutte le aule in un unica tabella, ma non possiamo usare il metodo *alldifferent* sulla colonna per esprimere l'impossibilità di tenere due o più corsi nella stessa aula, cioè per la stesso t al variare del corso m il valore della cella cioè le aule : 1...n devono essere diversi. Tale problema è stato risolto implementando i vincoli singolarmente considerando solo le celle > 0.

3 Velocità di escuzione

Eseguendo il codice per istanze di dati diversi, o con *searchannotations* diversi il tempo di esecuzione varia molto: si nota che eseguendo con search annotations $first\_fail, indomain\_random$ e $first\_fail, indomain\_median$, si ha garanzia di esecuzione in tempi brevi per tutte e tre casi, inoltre con $first\_fail, indomain\_random$ i tempi di esecuzione dei casi con dati in ingresso maggiori risultano più veloci, mentre per $first\_fail, indomain\_median$ il caso con 3 corsi è più veloce.

with additional data	Finished in	annotations
orario10E.dzn	10s 57msec	first_fail, indomain_random
orario10E.dzn	12s 255msec	first_fail, indomain_min
orario10E.dzn	16s 426msec	first_fail, indomain_split
orario10E.dzn	10s 988msec	first_fail, indomain_median
orario10E.dzn	>1 min	input_order, indomain_random
orario10E.dzn	>1 min	dom_w_deg, indomain_random
orario10E.dzn	>1 min	dom_w_deg, indomain_split
orario10E.dzn	>1 min	most_constrained, indomain_random
orario10corsi.dzn	9s 574msec	first_fail, indomain_random
orario10corsi.dzn	>1 min	first_fail, indomain_min
orario10corsi.dzn	>1 min	first_fail, indomain_split
orario10corsi.dzn	14s 647msec	first_fail, indomain_median
orario10corsi.dzn	>1 min	input_order, indomain_random
orario10corsi.dzn	>1 min	dom_w_deg, indomain_random
orario10corsi.dzn	>1 min	dom_w_deg, indomain_split
orario10corsi.dzn	>1 min	most_constrained, indomain_random
orario3corsi.dzn	2s 494msec	first_fail, indomain_random
orario3corsi.dzn	393msec	first_fail, indomain_min
orario3corsi.dzn	391msec	first_fail, indomain_split
orario3corsi.dzn	501msec	first_fail, indomain_median
orario3corsi.dzn	>1 min	input_order, indomain_random
orario3corsi.dzn	371msec	dom_w_deg, indomain_random
orario3corsi.dzn	934msec	dom_w_deg, indomain_split
orario3corsi.dzn	395msec	most_constrained, indomain_random

Table 2: tempi di esecuzione