Scheduler Eventi CSP

Alessio Portaro

Alma Mater Studiorum - Bologna

June 2022

Lo scopo

- Creare un sistema automatico per organizzare i vari eventi di una conferenza
- Massimizzare il gradimento di ogni partecipante alla conferenza

Tecnologie

(a) Minizinc

(b) Gecode

Parametri (1)

I parametri definiti nel file *data.dzn* sono:

- **Break,End** : Orario di inizio e fine della pausa
- **P** : Numero di persone partecipanti (ma anche l'ID piu' grande di un partecipante)
- **N** : Numero di lezioni (ID piu' grande di un evento)
- **X** : Numero di aule/sale disponibili
- **Y** : Numero di giorni a disposizione

Parametri (2)

Altri parametri necessari sono:

- **H** : Array lunghezza N che indica l'ID della persona che presenta la lezione
- **Len** : Array lunghezza N che indica la lunghezza di ogni evento
- **I** : Matrice NxP dove viene indicato l'interesse (dominio 1-5) che ogni persona ha nel partecipare ad ogni evento

Variabili decisionali (1)

Per ogni lezione della conferenza dobbiamo decidere :

- **D** : Il giorno in cui si tiene la lezione
- **R** : L'aula/sala in cui si tiene la lezione
- **M** : L'orario in cui inizia la lezione

Variabili decisionali (2)

Per ogni partecipante dobbiamo capire a quali lezioni puo' partecipare per costruire uno scheduling consistente. Per cui definiamo :

- **Choice** : Una matrice booleana di dimensione NxP dove scrivere quale persone partecipano ad ogni evento

Combinandola con **I** (interesse) possiamo infine definire :

- **Score** : Il grado di interesse complessivo all'evento da parte dei partecipanti

Modellazione dell'orario (1)

E' possibile modellare l'orario come una variabile nel dominio [0-420], ovvero ogni minuto a partire dalle 9 per le successive 7 ore disponibili ogni giorno.

Nota

Tuttavia con un dominio cosi' grande il solver non riesce a risolvere il problema.

Modellazione dell'orario (2)

La variabile **M** per cui non rappresenta direttamente un orario ma la posizione che un evento occupa in un ordinamento di tutti gli eventi nello stesso giorno e la stessa sala.

- Rimane tuttavia l'idea di modellare il tempo come descritto in precedenza, nel momento in cui si deve verificare la consistenza della soluzione (scheduling).

Vincoli

Nel programma Minizinc abbiamo due gruppi di vincoli (oltre ai diversi assert sui parametri):

- Vincoli che definiscono uno scheduling accettabile (*strong constraints*)
- Vincoli per definire quale tra tutte le soluzioni accettabili soddisfa maggiormente tutti i partecipanti alla conferenza (*soft constraints*)

Codice

Il programma sviluppato e' disponibile al repository Github

Test e risultati (1)

```
Running schedule.mzn, events10.dzn
15s 305msec

Command: minizinc --json-stream --push-working-directory $HOME/Documents/Lectures-Scheduler-CSP --param-file-no-push /tmp/mzn_ZxwjIG.mpc --pop-working-directory
 $HOME/Documents/Lectures-Scheduler-CSP/schedule.mzn $HOME/Documents/Lectures-Scheduler-CSP/events10.dzn
Configuration:
{
    "intermediate-solutions": false,
    "output-time": true,
    "parallel": 4,
    "solver": "org.gecode.gecode@6.3.0",
    "solver-statistics": true,
    "time-limit": 300000
}
Warning: multiple executables '../../../bin/fzn-gecode' found on the system, using '/home/alecio/Downloads/MiniZincIDE-2.6.3-bundle-linux-x86_64/bin/fzn-gecode'
Warning: multiple executables '../../../bin/fzn-gecode' found on the system, using '/home/alecio/Downloads/MiniZincIDE-2.6.3-bundle-linux-x86_64/bin/fzn-gecode'
246
7
[55, 55, 90, 120, 135, 140, 60, 90, 105, 140]
[1, 2, 3, 4, 1, 2, 5, 3, 4, 5]
[1, 1, 1, 1, 2, 2, 1, 2, 2, 2]
% time elapsed: 15s 213msec
----------
==========
%%%mzn-stat: failures=34539
%%%mzn-stat: initTime=0.084103
%%%mzn-stat: nodes=75527
%%%mzn-stat: peakDepth=113
%%%mzn-stat: propagations=6509149
%%%mzn-stat: propagators=10732
%%%mzn-stat: restarts=0
%%%mzn-stat: solutions=199
%%%mzn-stat: solveTime=14.027
%%%mzn-stat: variables=7631
%%%mzn-stat-end
Finished in 15s 305msec.
```

Figure: Output esecuzione con 10 eventi

Test e risultati (2)

Esperimenti con **Y** costante :

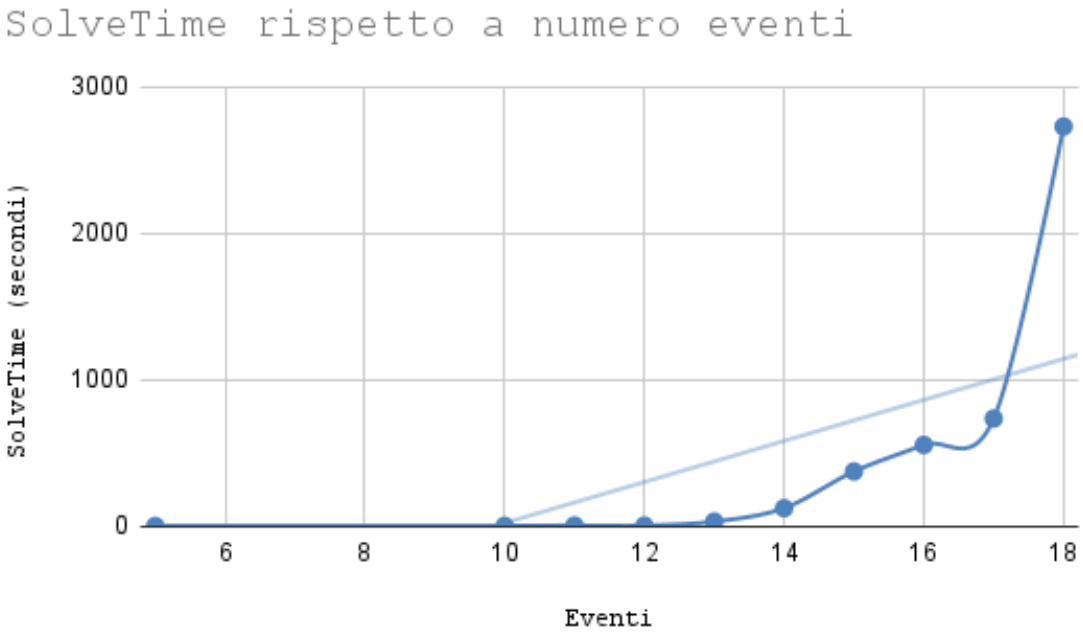


Figure: Y=5, N=10..18

Test e risultati (3)

Esperimenti con rapporto **N** su **Y** costante :

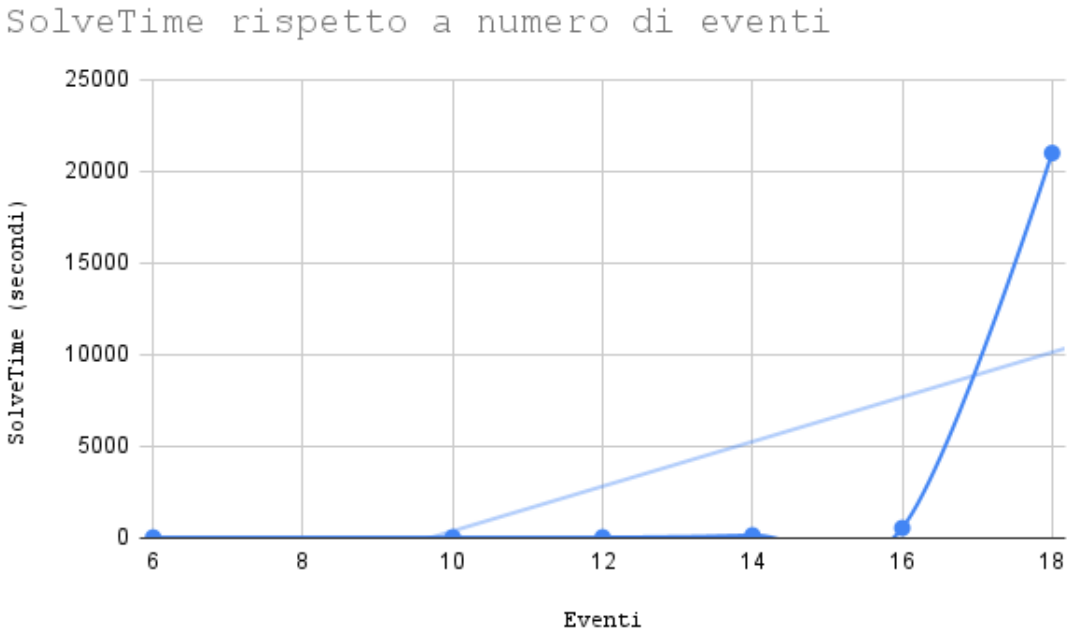


Figure: Y=N/2, N=10..18

Test e risultati (4)

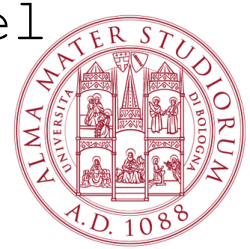
Minizinc permette di aggiungere delle **search annotations** per specificare la strategia di esplorazione dello spazio di ricerca.

Test e risultati (5)

Le esecuzioni con l'aggiunta delle annotations:

- **int_search(D, first_fail, indomain_min)** -> scelta della variabile con dominio piu' piccolo;
- **int_search(D, occurrence, indomain_min)** -> scelta della variabile con piu' vincoli;

hanno portato in entrambi i casi a risultati peggiori rispetto alla stessa versione del programma senza annotations.



Test e risultati (5)

In allegato le tabelle complete, contenenti i tempi di esecuzione di tutte le esecuzioni avviate.

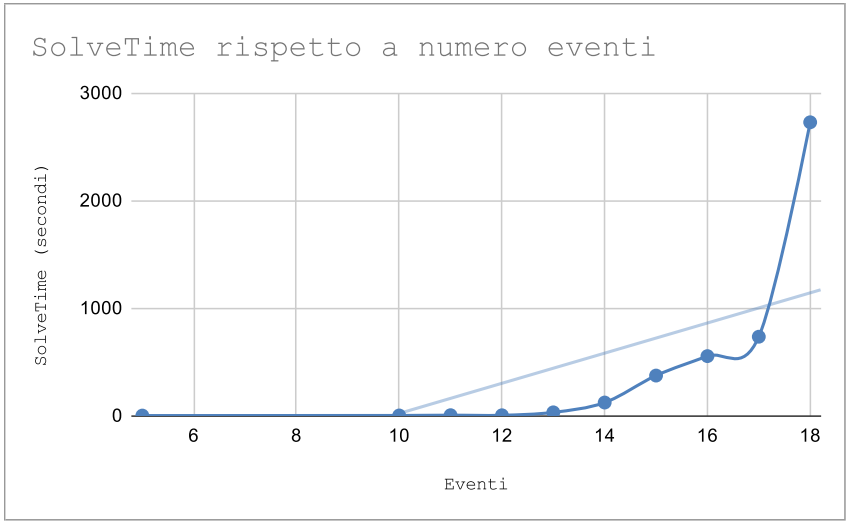
Funzione obiettivo senza annotations			
Eventi	Variabili	Nodi	Tempo
5	1504	300	0,01484
10	6154	15902	0,99616
11	7990	40232	3,57224
12	9556	36908	2,81889
13	11275	126730	30,0748
14	13150	312033	121,266
15	15184	799181	371,983
16	17380	757510	552,769
17	19741	850033	733,141
18	22270	3775604	2726,77

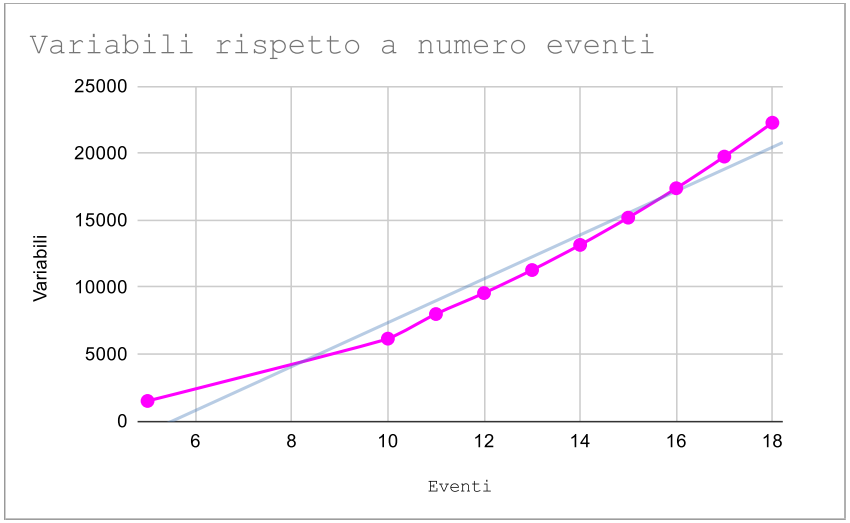
x	13,43
x	3,59
x	0,79
x	10,67
x	4,03
x	3,07
x	1,49
x	1,33
x	3,72

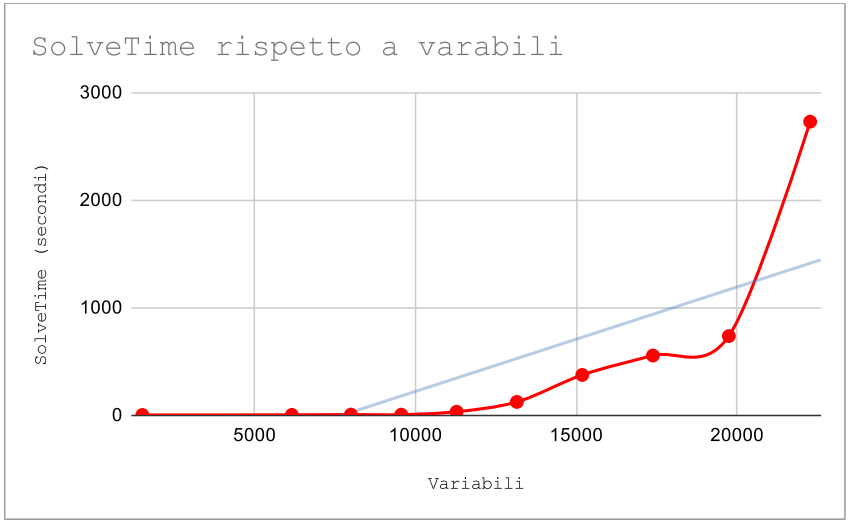
Numero di giorni (Y)	5

Numero di stanze (X)	1

Thread paralleli solver	4







Funzione obiettivo senza annotations				
Eventi	Giorni (Y)	Variabili	Nodi	Tempo
6	3	2170	2005	0,058183
10	5	6154	39888	8,03025
12	6	9556	52337	8,23176
14	7	13150	394092	115,754
16	8	17380	1206356	518,772
18	9	22270	26487195	20999,8

x	34,5043
x	0,5125
x	7,0309
x	2,2408
x	20,2399

Numero di stanze (X)	1

Thread paralleli solver	4