Assegnamento articoli

Francesco Autera

January 24, 2020

1 Il Problema

Il Problema assegnatomi prevedeva di assegnare a dei revisori un set di articoli da revisionare. La risoluzione del problema é stata sviluppata come un problema di soddisfacimento dei vincoli ed é stato usato per la risoluzione la libreria Numberjack in python.

2 Implementazione

2.1 Generalitá

Come scritto in precedenza ho usato di un linguaggio python 2.7 per la risoluzione del problema, con il solver Mistral. Il mio programma per funzionare utilizza:

1. **1 dataset1,dataset2** queste sono cartelle, che vedremo meglio nella sezione successiva ,contengono gli input per risolvere il problema
2. **2 Revisorisched.py** file python che presenta il codice per la risoluzione del problema e tutti gli input vengono presi dalle cartelle dataset1 o dataset2 a seconda di quale problema si vuole svolgere.

2.2 Cartella Dataset

Nella cartella dataset(1 o 2) troviamo vari file di tipo .npy. Mostro ora cosa contengono questi file:

1. **articoli:** contiene tutti i nomi degli articoli sottomessi
2. **autori:** contiene tutti i nomi degli autori degli articoli
3. **revisori:** contiene tutti i nomi dei revisori; all'interno della parentesi troviamo il corrispondente autore che lui é(ho considerato che ogni revisore é anche autore)
4. **settori:**contiene tutti i nomi delle aree scientifiche

5. **autoriarticoli:**contiene una matrice booleana di (articoli,autori) dove 1 se quel determinato articolo é stato scritto dal quell'autore altrimenti 0

6. **settorearticoli:**contiene una matrice booleana di(articoli,settori) dove 1 se quell'articolo appartiene al quel settore altrimenti 0

7. **settorerevisori:**contiene una matrice booleana di(revisori,settori) dove 1 se quel revisore appartiene al quel settore altrimenti 0

8. **autoriconflitto:**contiene una matrice booleana di (revisori,autori) dove 1 se quel revisore é in conflitto con quell'autore(essendo revisore anche un autore lui é in conflitto con se stesso)

2.3 Revisoresched.py

Ora andiamo ad analizzare il file .py e andiamo ad analizzare i vincoli che ci hanno permesso di risolvere il problema. Gli elementi descritti nella sezione precedente vengono importati con gli stessi nomi all'interno del file attraverso un metodo della libreria numpy. Per i file .npy ho usato il metodo numpy.load(filename,..) dopo che ho salvato la matrice con il metodo numpy.save. Per i file .txt ho usato il metodo numpy.loadtxt(filename,..). La **variabile** in gioco é **revisorisched**; questa é una matrice di(revisori,articoli) in cui ogni variabile puo assumere valore(0,1), dove 1 se il revisore puo revisionare quell'articolo altrimenti 0.

2.4 I vincoli:

I vincoli in questione che risolvono il problema sono:

[$Sum(row) <= R$ for row in revisorisched.row]

Questo vincolo controlla che la somma degli articoli revisionati da un revisore sia minore di R.

[$Sum(col) == L$ for col in revisorisched.col]

Questo vincolo controlla che la somma dei revisori che revisionano un determinato articolo sia esattamente L.

```
for i in range(len(revisori)) do
    for j in range(len(settori)) do
        if (settorerevisori[i][j] == 0) then
            for k in range(len(articoli)) do
                if (settorearticoli[k][j] == 1) then
                    model.add(revisorisched[i][k] == 0)
                end if
            end for
        end if
    end for
end for
```

Questa parte di codice mi aggiunge 0 alla matrice revisorisched alla posizione dell'i-esimo revisore e al k-esimo articolo se il revisore non appartiene al settore ($settorearticoli[i][j] == 0$) dell'articolo in questione ($settorerevisori[k][j] == 1$).

```
1: for i in range(len(revisori)) do
2:     for j in range(len(settori)) do
3:         if (settorerevisori[i][j] == 1) then
4:             for k in range(len(articoli)) do
5:                 if (settorearticoli[k][j] == 1) then
6:                     for p in range(len(autori)) do
7:                         if (autoriconflitto[i][p]) then
8:                             if (autoriarticoli[k][p]) then
9:                                 model.add(revisorisched[i][k]== 0)
10:                            end if
11:                        end if
12:                    end for
13:                end if
14:            end for
15:        end if
16:    end for
17: end for
```

Questa seconda parte di codice mi aggiunge 0 alla matrice revisorisched se il revisore appartiene allo stesso settore dell'articolo ma questo é stato scritto da un autore che é in conflitto con il revisore o che lo ha scritto lui stesso.

L'idea della mia implementazione é stata quella di trovare quegli articoli che non possono essere revisionati dai revisori e quelli rimanenti(sono quelli che possono revisionare) soddisfano i due vincoli iniziali allora il programma mette 1 nei restanti spazi della matrice revisorisched.

3 Risultati e Conclusioni

I risultati che mostro sono realizzati con il dataset2 che ha come R=4 e L=2 con 8 articoli, 5 revisori,12 autori, 3 settori.

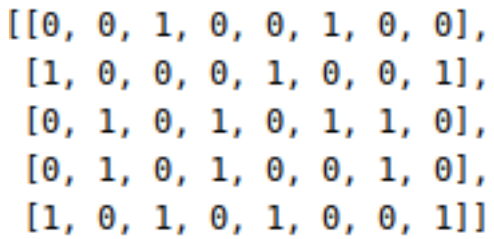

Figure 1: revisorisched risultante

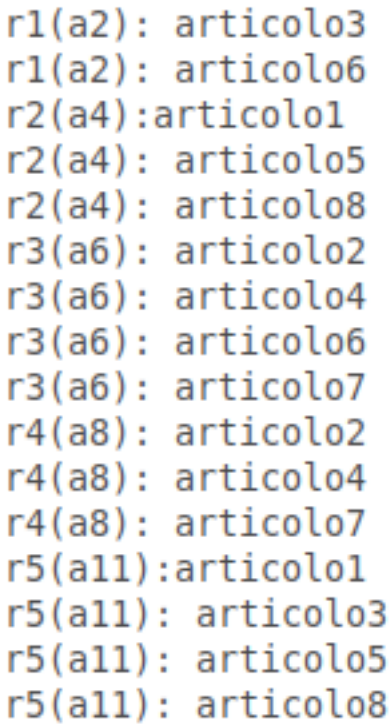

Figure 2: forma letterale della matrice soprastante

Per stampare in forma letterale ho usato una matrice di appoggio che produceva lo stesso risultato della matrice revisorisched. Se il solver non dava soluzione stampava "INSODDISFACIBILE". Il problema é stato implementato e risolto su una piattaforma con le seguenti specifiche: 8GB RAM,Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz,Linux Mint.