Relazione progetto Asp/Clingo anno 2019/2020

Studenti:

Sanfilippo Paolo

Giuseppe Biondi

Antonio Surdo

Traccia:

Formulare e risolvere il seguente problema di soddisfacimento di vincoli: generazione del calendario del Master in Progettazione e Management del Multimedia per la Comunicazione dell'Università di Torino che, con la sua storia iniziata nel 1996, è il più longevo d'Europa. Le lezioni del master si svolgono il venerdì (8 ore) e il sabato (4 o 5 ore) nell'unica aula assegnata al Master, per 23 settimane. Inoltre, sono previste due settimane full-time, la 7a e la 16a, con lezioni dal lunedì al sabato (8 ore al giorno da lunedì a venerdì, 4 o 5 ore al sabato). Il calendario dovrà tener conto di alcuni vincoli, da considerarsi partizionati in rigidi (da soddisfare tassativamente) e auspicabili (da soddisfare ove possibile, e nel maggior numero possibile).

Predicati e costanti utilizzati:

- Settimane(1..24)
- Giorni_per_settimana_full(lunedi;martedi,mercoledi;giovedi)
- Numero_giorno(giornoX,numeroX)
- Insegnamenti(insegnamento1;……;insegnamentoN)
- Professori(professore1;….;professoreN)
- Inizio_ora(8;9;10;11;12;14;15;16)
- Settimana_full_time(7;16)
- Professori_insegnamento(insegnamentoX,professoreX)
- Ore_insegnamento(insegnamentoX;durataX)
- Insegnamento_successivo(insegnamentoX,insegnamentoY)
- insegnamento_successivo_4ore(insegnamentoX,insegnamentoY).

Modellazione problema e vincoli utilizzati:

- Ad ogni settimana associamo i giorni, attraverso il predicato: giorno_in_settimana(settimanaX,GiornoX):

```
2{giorno_in_settimana(S,G):giorni_settimana(G)}2:-settimane(S).
```

- Ad ogni settimana full time associamo i giorni in più:

```
4{giorno_in_settimana(S,G):giorni_per_settimana_full(G)} 4:-settimane(S),settimana_full_time(S).
```

- Ad ogni giorno della settimana, vengono associate delle ore, attraverso la creazione del predicato “inizio_ora_giorno_settimana(S,G,O)”, se il giorno è Sabato, vengono associate 8 ore, altrimenti 4:

```
1. 8{inizio_ora_giorno_settimana(S,G,O):inizio_ora(O)}8:-
   giorno_in_settimana(S,G),not G==sabato.
2. 4{inizio_ora_giorno_settimana(S,G,O):inizio_ora(O)}5:-
   giorno_in_settimana(S,G),G==sabato.
```

- Per associare ad ogni ora di un giorno di una settimana “inizio_ora_giorno_settimana(S,G,O)” un solo insegnamento, dapprima abbiamo creato tutti i possibili accoppiamenti tra gli insegnamenti e le ore, creando il predicato “lezione(S,G,O,I)”, poi attraverso un integrity constraint abbiamo fatto si che non possano esistere due lezioni nella stessa ora, dello stesso giorno, della stessa settimana con insegnamenti differenti.

```
1. {lezione(S,G,O,I):insegnamenti(I)}:-
   inizio_ora_giorno_settimana(S,G,O).
2. :-lezione(S,G,O,I1),lezione(S,G,O,I2),I1!=I2.
```

- Per soddisfare il vincolo secondo il quale un corso nello stesso giorno deve avere un minimo di 2 ore e un massimo di 4, abbiamo prima creato un predicato “conteggioOreGiorno(I,S,G,Conteggio)” che attraverso l’utilizzo dell’operatore “#count” memorizza il totale delle ore di un determinato insegnamento in un determinato giorno. In seguito con l’utilizzo di due integrity constraint abbiamo eliminato tutti i modelli che hanno conteggi <2 o >4:

```
1. conteggioOreGiorno(I,S,G,Conteggio):-
   Conteggio=#count{O:lezione(S,G,O,I)},lezione(S,G,_,I).
2. :-conteggioOreGiorno(I,S,G,Conteggio),Conteggio>4.
3. :-conteggioOreGiorno(I,S,G,Conteggio),Conteggio<2.
```

- Per far sì che la somma delle durate delle lezioni di un insegnamento sia uguale alle ore assegnate all'insegnamento, abbiamo utilizzato un approccio simile al vincolo precedente. Prima vengono conteggiate le ore di un determinato corso, e poi si eliminato tutti i modelli nel quale i conteggi non sono uguali alle ore assegnate all’insegnamento:

```
1. conteggioOreTotali(I,Conteggio) :-Conteggio = #count{ S,G,O :
   lezione(S,G,O,I)}, insegnamenti(I).
2. :-
   conteggioOreTotali(I,Conteggio),ore_insegnamento(I,O),Conteggi
   o!=O.
```

- Abbiamo creato il predicato “lezione(S,G,O,I,P)” per associare ad ogni lezione, il relativo professore:

```
lezione(S,G,O,I,P):-lezione(S,G,O,I),professori_Insegnamento(I,P).
```

- Per verificare che un docente non superi le 4 ore di lezione in un giorno, abbiamo nuovamente seguito l'approccio #count e integrity constraint:

```
1. conteggioOreProf(S,G,P,Conteggio):-
   Conteggio=#count{O:lezione(S,G,O,_,P)},lezione(S,G,_,_,P).
2. :-conteggioOreProf(_,_,_,Conteggio),Conteggio>4.
```

- Per creare due blocchi liberi di due ore ciascuno per le lezioni di recupero, abbiamo dapprima creato il predicato "insegnamenti(recupero)", in seguito questo insegnamento viene associato a due settimane e a due giorni di queste settimane. Infine, attraverso un integrity constraint le ore in uno stesso giorno sono vincolate ad essere contigue:

```
1. insegnamenti_recupero(recupero).
2. 2{lezione_recupero(S,G,I):giorno_in_settimana(S,G)}2:-
   insegnamenti_recupero(I).
3. 2{lezione(S,G,O,I):inizio_ora(O)}2:-lezione_recupero(S,G,I).
4. :-lezione(S,G,O,I),lezione(S,G,O1,I),insegnamenti_recupero(I),|O-
   O1|!=1,O!=O1.
```

- Per imporre che il corso project management debba essere finito prima della prima settimana_full_time (la settima) è stato creato il seguente integrity constraint:

```
:-lezione(S,_,_,I),I==project_Management,S>6.
```

- Il primo giorno di lezione prevede che, nelle prime due ore, vi sia la presentazione del master, per far ciò asseriamo due predicati:

```
1. lezione(1,venerdi,8,presentazione_master).
2. lezione(1,venerdi,9,presentazione_master).
```

- Per soddisfare alcuni vincoli del problema, si è venuta a creare la necessità della creazione di due predicati: "prima_lezione(S,G,O,I)" e "ultima_lezione(S,G,O,I)" che come si capisce dai nomi rappresentano rispettivamente la prima e l'ultima lezione di un determinato insegnamento. I due predicati sono stati dedotti, in maniera speculare, attraverso l'utilizzo di due predicati ausiliari che vanno a cercare la prima(ultima) settimana e il primo(ultimo) giorno. Quindi, si va a cercare dapprima la prima(ultima) settimana di lezione, imponendo che non esista una lezione di quell'insegnamento con una settimana minore(maggiore). In seguito, si fa la medesima cosa per i giorni in quella settimana e per le ore nel giorno.

```
1. x(S,G,O,I):-lezione(S,G,O,I),lezione(S1,G1,O1,I),S1<S.
2. prima_settimana(S,G,O,I):- not x(S,G,O,I),lezione(S,G,O,I).
3. x2(S,G,O,I):-
   prima_settimana(S,G,O,I),prima_settimana(S,G1,O1,I),G1<G.
4. primo_giorno(S,G,O,I):- not x2(S,G,O,I),prima_settimana(S,G,O,I).
5. x3(S,G,O,I):-primo_giorno(S,G,O,I),
   primo_giorno(S,G,O1,I),O1<O.
6. prima_lezione(S,G,O,I):- not x3(S,G,O,I),primo_giorno(S,G,O,I).
```

```
1. y(S,G,O,I):-lezione(S,G,O,I),lezione(S1,G1,O1,I),S1>S.
2. ultima_settimana(S,G,O,I):- not y(S,G,O,I),lezione(S,G,O,I).
3. y2(S,G,O,I):-
   prima_settimana(S,G,O,I),prima_settimana(S,G1,O1,I),G1>G.
4. ultimo_giorno(S,G,O,I):- not y2(S,G,O,I),prima_settimana(S,G,O,I).
5. y3(S,G,O,I):-primo_giorno(S,G,O,I),
   ultimo_giorno(S,G,O1,I),O1>O.
6. ultima_lezione(S,G,O,I):- not y3(S,G,O,I),ultimo_giorno(S,G,O,I).
```

- La prima lezione dell'insegnamento "Accessibilità e usabilità nella progettazione multimediale" deve essere collocata prima che siano terminate le lezioni dell'insegnamento "Linguaggi di markup". Per fare ciò abbiamo usato il seguente integrity constraint:

```
:-
prima_lezione(S,_,_,accessibilita_e_usabilita_nella_progettazione
_multimediale),ultima_lezione(S2,_,_,linguaggi_di_markup),S>S2.
```

- Se un corso è successivo ad un altro, la prima lezione del corso deve essere successiva all'ultima lezione dell'altro corso. Quindi facciamo in modo che non esistano modelli in cui la prima lezione di un corso abbia una settimana <= all'ultima lezione di un altro corso, se il primo corso è successivo al secondo:

```
:-
insegnamento_successivo(I,I2),prima_lezione(S,_,_,I),ultima_lezione(S2,
_,_,I2),S<=S2.
```

- La distanza tra la prima e l'ultima lezione di ciascun insegnamento non deve superare le 6 settimane. Quindi, la differenza tra le settimane deve essere <=6:

```
:-prima_lezione(S,_,_,I),ultima_lezione(S1,_,_,I),S1-S>6.
```

- La prima lezione degli insegnamenti "Crossmedia: articolazione delle scritture multimediali" e "Introduzione al social media management" devono essere collocate nella seconda settimana full-time. Quindi eliminiamo tutti i modelli in cui, questo non è vero:

```
1. :-
   prima_lezione(S,_,_,crossmedia_articolazione_delle_scritture_mul
   timediali),S!=16.
2. :-
   prima_lezione(S,_,_,introduzione_al_social_media_management),
   S!=16.
```

- La distanza fra l'ultima lezione di "Progettazione e sviluppo di applicazioni web su dispositivi mobile I" e la prima di "Progettazione e sviluppo di applicazioni web su dispositivi mobile II" non deve superare le due settimane. Anche qui viene usato un integrity constraint, per eliminare i modelli dove la condizione non viene soddisfatta:

```
:-
ultima_lezione(S,_,_,progettazione_e_sviluppo_di_applicazioni_web_su_dispositivi_mobile_I),prima_lezione(S2,_,_,progettazione_e_sviluppo_di_applicazioni_web_su_dispositivi_mobile_II),S2-S>2.
```

Considerazioni e tempi esecuzione:

Siamo riusciti ad implementare tutti i vincoli rigidi e tutti i vincoli auspicabili tranne uno. Abbiamo ottenuto dei tempi di esecuzione di 23 minuti circa; questo risultato a nostro parere è più che positivo. Questa valutazione deriva dal paragone con tempi di esecuzione provenienti da precedenti versioni del modello, in cui avevamo utilizzato, per alcuni vincoli, scelte implementative diverse e da tempi di esecuzione ottenuti da un confronto con altri colleghi che hanno avuto modo di misurarsi con questo progetto.

Il vincolo auspicabile che ci ha dato problemi è quello dei corsi successivi di 4 ore. Nello specifico abbiamo provato ad implementarlo, ma dopo diverse ore di esecuzione non siamo riusciti ad ottenere una soluzione. Di seguito riportiamo l'implementazione di tale vincolo, anche se non è stata più utilizzata nel modello:

```
1. conteggioOrePrecedente(I,I1,Conteggio):-#count{S,G,O:
   lezione(S,G,O,I),prima_lezione(S1,G1,O1,I1),S<=S1,numero_giorno(G,X),numero_giorno(G1,X1),X<=X1,O<O1}=Conteggio,insegnamenti(I),insegnamenti(I1).
2. :-
   insegnamento_successivo_4ore(I,I1),conteggioOrePrecedente(I,I1,Conteggio),Conteggio<4.
```

Quello che abbiamo provato a fare è contare tutte le ore precedenti di un corso rispetto ad un altro e in seguito per tutti i corsi che sono successivi di 4 ore, imporre che tale conteggio non sia minore di 4. Nonostante concettualmente il vincolo sembri corretto, come detto, non siamo riusciti ad ottenere una soluzione dopo diverse ore, ciò ci ha fatto pensare che data le nostre scelte implementative e la conseguente modellazione del problema, il costo computazione del conteggio e dell'integrity constraint sia troppo oneroso.

Risultati:

Per mostrare i risultati visivamente, in modo da facilitare la verifica della correttezza della soluzione, abbiamo deciso di creare uno script Python che dapprima esegue il file “calendario_master.cl” che è il file contenente il nostro modello, e in seguito crea un file Excel dove è possibile consultare per ogni settimana il calendario delle lezioni. Per fare ciò abbiamo utilizzato la libreria di Python Pandas. Il file Excel risultante, che alleghiamo, è composto da 24 Sheet, ognuno di essi raffigura l’orario di una specifica settimana di master.

orario.xlsx