Proposta di un calendario sportivo calcolato mediante CLINGO

Dipartimento di Informatica: Università di Torino

Autori: Alessandro Saracco, Leonardo Magliolo , Mattia Marra

Intelligenza artificiale e laboratorio

A.A. 2021/2022

Parametrizzazione

Per facilitare il debug del programma sono state aggiunte delle costanti per rendere parametrizzabile il problema trattato.

```
#const num_partite = 10. %10
#const num_giornate = 38. %38
#const num_derby = 4. %4 per il facoltativo
#const distanza_ritorno_minima = 10. %10 per il facoltativo
```

Rappresentazione del campionato

E' stato deciso di utilizzare, per la generazione del calendario sportivo, il campionato di calcio italiano dell'anno 2021/2022.
Per la definizione delle squadre e delle giornate sono stati utilizzati aggregati per rendere più compatta la sintassi descrittiva.
E' stata inoltre indicata la città d'appartenenza per ogni squadra per mezzo del predicato binario **in(squadra, città)**.

```
squadra(juventus;milan;inter;napoli;lazio;roma;
        fiorentina;atalanta;sassuolo;verona;
        torino;udinese;empoli;bologna;spezia;
        sampdoria;cagliari;genoa;venezia;salernitana).

in(juventus, torino).
%...
in(salernitana, salerno).

giornata(1..num_giornate).
```

(vincolo 1)

Rappresentazione delle partite

E' stato modellato il concetto di partita per mezzo del predicato ternario **partita(Giornata, Squadra1, Squadra2)**.
Per convenzione è stato scelto di indicare la squadra che gioca in casa all'interno del secondo parametro (**Squadra1**).

Per la generazione delle partite è stato utilizzato un aggregato, in modo tale che crei esattamente ***num_partite*** per ogni giornata del campionato.
E' stata inoltre rispettata la condizione tale per cui non possa esistere una partita avente una squadra che affronti se' stessa.

```
num_partite {partita(G, S1, S2):squadra(S1), squadra(S2), S1!=S2} num_partite :- giornata(G).
```

(vincolo 2)

Al più una partita giocata nella stessa giornata

E' stato introdotto il predicato **numeroPartiteGiornata(Giornata, Squadra, Conteggio)** per calcolare quante partite una determinata squadra ha giocato in una giornata.

Tale predicato è stato utilizzato per implementare il vincolo che impedisce a una squadra di giocare più volte nella stessa giornata.

```
numeroPartiteGiornata(G, S1, Conteggio):-giornata(G), squadra(S1),
                        Conteggio1 = #count{S2: partita(G, S1, S2)},
                        Conteggio2 = #count{S2: partita(G, S2, S1)},
                        Conteggio = Conteggio1+Conteggio2.

:-giornata(G), squadra(S), numeroPartiteGiornata(G,S,Conteggio), Conteggio > 1.
```

(vincolo 3)

Giocate di andata e di ritorno

E' stato introdotto un vincolo in modo tale che ogni squadra affronta due volte tutte le altre squadre, una volta in casa e una
volta fuori casa, ossia una volta nella propria città di riferimento e una volta in quella dell'altra squadra.

```
:-squadra(S1), squadra(S2), S1 != S2, Sum = #count{G: partita(G, S1, S2)}, Sum != 1.
```

Considerata la generica coppia di **squadre (Squadra1, Squadra2)**, è possibile istanziare **S1** come **Squadra1** ed **S2** come **Squadra2** e viceversa,
in questo modo si impone l'esistenza di una sola partita giocata in casa e fuori casa per ogni coppia di squadre
(sfruttando la convenzione per la dichiarazione delle partite descritta in precedenza).

(vincolo 4)

Al più una partita giocata in casa per giornata

E' stato introdotto un vincolo in modo tale che due squadre della stessa città condividano la stessa struttura di gioco, quindi non possano giocare entrambe in casa nella stessa giornata.

```
:-in(S1, C), in(S2, C), partita(G, S1, S3) , partita(G, S2, S4), S1!=S2.
```

(vincolo 5)

Non più di due partite consecutive in casa

E' stato introdotto un vincolo in modo tale che ciascuna squadra non giochi mai più di due partite consecutive in casa o fuori casa.

```
:-giornata(G1), G2 = G1 + 1, G3 = G2 + 1, partita(G1, S1, S2), partita(G2, S1, S3), partita(G3, S1, S4).
:-giornata(G1), G2 = G1 + 1, G3 = G2 + 1, partita(G1, S2, S1), partita(G2, S3, S1), partita(G3, S4, S1).
```

E' stato deciso di dividere il vincolo in due vincoli indipendenti, uno per le partite in casa e uno per le partite fuori casa.
I vincoli sfruttano la proprietà delle partite tale per cui viene dichiarata prima la squadra che gioca in casa e poi quella che gioca fuori casa,
pertanto nel primo vincolo la squadra fissata **S1** è indicata dentro il secondo parametro di Partita, e nel secondo vincolo **S1** è indicata nel terzo parametro della stessa.

(vincolo 6)

Numero di derby

E' stato introdotto un vincolo che introduca la presenza di ***num_derby*** derby, ossia coppie di squadre che fanno riferimento alla medesima città.

```
:-Conteggio = #count{S1, S2: squadra(S1), squadra(S2), in(S1, C), in(S2, C),
                S1 != S2, partita(G, S1, S2)}, Conteggio/2 != num_derby.
```

Se **Squadra1** e **Squadra2** appartengono alla stessa città allora **partita(Giornata, Squadra1, Squadra2)** e **partita(Giornata, Squadra2, Squadra1)** rappresentano lo stesso derby; pertanto il conteggio delle partite all'interno del vincolo è stato diviso per due per evitare di contare due volte lo stesso derby.

(vincolo 7)

Distanza minima andata-ritorno

E' stato introdotto un vincolo in modo tale che la distanza tra una coppia di gare di andata e ritorno sia di almeno ***distanza_ritorno_minima*** giornate.
Supponendo che la distanza minima sia 10, allora se **SquadraA** vs **SquadraB** è programmata per la giornata 12, il ritorno **SquadraB** vs **SquadraA** verrà schedulato non prima dalla giornata 22.

```
:-partita(G1, S1, S2), partita(G2, S2, S1), G2>G1,
  Distanza = G2-G1, Distanza < distanza_ritorno_minima.
```

(vincolo 8)

Analisi delle prestazioni

Specifiche hardware:

- CPU: AMD Ryzen 7 5800X
- RAM: Corsair VENGEANCELPX16GB DDR4
- SSD: Samsung MZ-V8V1T0 980

Tutti i test prestazionali sono stati eseguiti utilizzando i parametri d'esecuzione:

- --time-limit=0, per impostare un tempo di computazione limite arbitrario
- --parallel-mode=16, per utilizzare tutti e 16 i threads di cui l'architettura dispone

Analisi delle prestazioni (2)

Sebbene i tempi di computazione rispetto al programma (formulato in relazione alla prima consegna con i punti facoltativi) eccedano le otto ore prima di trovare una soluzione, sono stati omessi selettivamente i vincoli indicati nelle slide come 4, 6, 8 al fine di ricavare soluzioni subottimali in tempi minori:

- Singolarmente i set di vincoli {4}, {6}, {8} sono stati eseguiti entro tempistiche inferiori ai 20 secondi.
- Anche modificando di una certa quantità i parametri indicati in consegna non è stato possibile ottenere una soluzione entro tempistiche accettabili per il set di vincoli {4,6}.
- Mantenendo i parametri del programma invariati come da consegna è stata trovata una soluzione per il set di vincoli {6,8} con tempistiche inferiori ai 40 secondi.
- Riducendo la distanza minima tra le partite di andata e di ritorno da 10 a 4 è stata trovata una soluzione per il set di vincoli {4,8} con tempistiche inferiori a 2 minuti.

In definitiva, il vincolo che si è rivelato più problematico al fine di calcolare un calendario accettabile per la consegna è stato il 4.