18 settembre 2017 - laboratorio

Constraint Programming

Prof. Marco Gavanelli

18 settembre 2017

Descrizione problema

Ad un pranzo di matrimonio, bisogna decidere in quali tavoli far sedere ciascuno degli invitati. Il numero NI di invitati è riportato nel file `matrimonio.pl` in un fatto

```
num_invitati(NI).
```

Si ha poi un fatto

```
num_tavoli(NT).
```

che riporta il numero di tavoli NT.

Purtroppo, non tutti gli invitati vanno d'accordo fra di loro: sono stati invitati genitori che si sono separati ed altre persone che potrebbero scatenare delle risse se si trovassero allo stesso tavolo. Gli invitati che sono in conflitto sono riportati in un predicato

```
conflitto(I,J).
```

Se due persone I e J sono in conflitto, non possono sedere allo stesso tavolo.

Si desidera invece far sedere, per quanto possibile, le persone che si conoscono allo stesso tavolo. Se due persone I e J si conoscono, il predicato

```
conosce(I,J)
```

è vero.

Si calcoli una possibile assegnamento delle persone ai tavoli, massimizzando il numero di persone che si conoscono che si trovano allo stesso tavolo.

Per chi fa Applicazioni di Intelligenza Artificiale (punti 8)

Si risolva il problema usando ECLiPSe.

Suggerimento: il predicato nth1(N,L,X), definito nella libreria listut, dato un numero N e una lista L, fornisce in X l' N-esimo elemento della lista. N deve essere **ground** (non può essere una variabile).

Per chi fa Constraint Programming

Il file `matrimonio.pl` contiene anche un fatto

```
capacita(C)
```

che riporta la capacità di ciascuno dei tavoli (i tavoli sono tutti uguali), ovvero il numero di persone che vi si possono sedere.

CLP (punti 15)

Si risolva il problema usando ECLiPSe

ASP (10 punti)

Si risolva il problema in Aswer Set Programming.

MiniZinc

Per esercitarsi su MiniZinc, vengono forniti i dati di input in formato .mzn nel file matrimonio_input.mzn.

Il file contiene i seguenti dati:

- `int: num_tavoli` numero dei tavoli
- `int: num_invitati` numero invitati
- `int: capacita` capacità dei tavoli o numero di persone che possono sedere a ciascuno dei tavoli
- `array [1..num_conflitti,1..2] of 1..num_invitati : conflitto` array che contiene i conflitti: per ogni *i* fra 1 e num_conflitti (dove num_conflitti è una costante data di tipo intero), conflitto[i,1] e conflitto[i,2] sono gli identificativi di due invitati che sono in conflitto

- `array [1..num_conoscenze,1..2] of int : conosce` array che contiene le conoscenze, strutturato come l'array precedente, ovvero: per ogni *i* fra 1 e num_conoscenze, conosce[i,1] e conosce[i,2] sono gli identificativi di due invitati che si conoscono.