Pràctica Programació amb Restriccions

Festival de Música de Cambra + ScalAT

Ismael El Habri, Lluís Trilla

28 de gener de 2019

Índex

Capítol 1

Minizinc: Festival de música de cambra

1.1 Model

1.1.1 Dades

Apart de les dades donades, hem definit `maxMinuts`, que consisteix en la durada màxima que pot tenir el festival, sent aquesta donada per la suma de durades de totes les peces, pel cas en que totes les peces es toquin una darrere l'altre. Aquesta dada s'usarà per definir el domini de variables posteriors. Per tal de reduir-lo, sabent que les durades son de multiples de 5, decidim dividir-la entre 5 (fent que les peces comencin en minuts múltiples de 5).

```
int: nPeces;
int: nMusics;
int: nInstruments;
int: nPrecs;
int: pressupost;

set of int: Peces = 1..nPeces;
set of int: Musics = 1..nMusics;
set of int: Instruments = 1..nInstruments;
set of int: Precs = 1..nPrecs;

array [Peces] of int: durada;
array [Peces, Instruments] of int: requereix;
array [Musics, Instruments] of bool: saptocar;
array [Musics] of int: salari;

array[Precs] of int: pred;
array[Precs] of int: succ;

int: maxMinuts = sum(durada) div 5;
set of int: MaxMinuts = 0..maxMinuts;
```

1.1.2 Variables i dominis

Hem definit les següents variables:

- `instrumentsXMusics`: Taula que ens indica per cada peça i músic, quin instrument toca aquest músic en aquesta peça, si és 0, indica que aquest músic no toca en aquesta peça. Per tant el domini és de 0 a `nInstruments`.
- `minutInici`: Array que ens indica el minut en que comença cada peça. El temps, com ja s'ha dit, està dividit entre 5 (per reduir l'espai de cerca). EL domini és de 0 a maxMinuts.
- `minutsTocatsXmusic`: Variable auxiliar que ens diu els minuts tocats per cada músic. Domini de 0..durada màxima del event.
- `quanAcaba`: Variable auxiliar que ens diu quan acaba cada peça. Domini igual a la anterior.

```
array[Peces, Musics] of var 0..nInstruments: instrumentsXmusics;
array[Peces] of var MaxMinuts: minutInici;
array[Musics] of var 0..sum(durada): minutsTocatsXmusic :: is_defined_var;
array[Peces] of var 0..sum(durada): quanAcaba :: is_defined_var;
```

1.1.3 Restriccions

Restricció per tal de que cada músic no toqui més d'un instrument en la mateixa peça

No cal definir-la, posat que ve donada pel viewpoint, que només permet un instrument per músic i peça.

Restricció per evitar que un músic toqui instruments que no domina

Consisteix en mirar per cada peça p i músic m, el instrument que toca m en la peça p sigui 0, o el domini (`saptocar[m, instrument]` és `true`).

```
constraint forall( p in Peces, m in Musics)(
    saptocar[m, instrumentsXmusics[p,m]] \/ (instrumentsXmusics[p,m] == 0)
);
```

Restricció per a que es toquin els instruments que es requereixen en cada peça

Per cada peça, contem quants instruments de cada es toquen, i mirem que sigui igual amb el que requereix la peça.

```
constraint forall(p in Peces, i in Instruments)(
    count([instrumentsXmusics[p,m]|m in Musics], i, requereix[p,i])
) :: domain;
```

Restricció per evitar que les peces que es solapin usin els mateixos músics

Per cada músic i dos peces, mirem que si el músic toca en les dos peces, una de les dos comenci abans que acabi l'altre.

```
constraint forall (p in Peces, m in Musics where instrumentsXmusics[p,m] != 0 )(
    forall (p2 in p+1..nPeces where instrumentsXmusics[p2,m] != 0 )(
        ( ( (minutInici[p]*5+durada[p]) <= minutInici[p2]*5) \/ ((minutInici[p2]*5 +
            durada[p2]) <= minutInici[p]*5) )
        /\ ( ( (minutInici[p2]*5) > minutInici[p]*5) \/ ((minutInici[p2]*5) <
            minutInici[p]*5) )
)) :: domain;
```

Restricció per mantenir-nos dins del pressupost

Primer cal donar valor a la variable `minutsTocatsXmusic`. Amb això usem la funció builtin de FlatZinc `int_lin_le(array int of int: as, array int of var int: bs, int: c)` que aplica la restricció següent:

$$\sum as[i] * bs[i] \leq c$$

```
constraint forall(m in Musics)(
    minutsTocatsXmusic[m] = sum(p in Peces where instrumentsXmusics[p,m]!=0)(durada[p]) ::
        defines_var(minutsTocatsXmusic[m])
);
constraint int_lin_le(salari,minutsTocatsXmusic, pressupost*5) :: domain;
```

Restricció de precedències

Aquesta restricció és tan senzilla com mirar que cada precedència acabi abans de que comenci la peça que ha de precedir.

```
constraint forall(pr in Precs)(
```

```
    minutInici[pred[pr]]*5 + durada[pred[pr]] <= minutInici[succ[pr]]*5
) :: domain ;
```

Restricció per donar valor a `quanAcaba`

Sumem la durada al minut d'inici, multiplicat per 5. Els temps de `quanAcaba` són relatius al inici (minut 0), però en temps real (no en funció de 5).

```
constraint forall(pr in Precs)(
    minutInici[pred[pr]]*5 + durada[pred[pr]] <= minutInici[succ[pr]]*5
) :: domain ;
```

Restriccions implicades

Com a restriccions implicades, hem forçat una mica els límits de durada. Fent que la peça que acaba abans, acabi en un minut més gran o igual a la duració de la peça més curta, i que la peça que acabi més tard, acabi en un minut més gran o igual que la peça més llarga.

L'intenció és reduir una mica l'espai de cerca, en zones on no hi ha solucions.

```
constraint max(quanAcaba) >= max(durada) :: domain;
constraint min(quanAcaba) >= min(durada) :: domain;
```

Capítol 2

ScalAT