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 música, 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 música. 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])
);
```

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) )
));
```

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);
```

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
);
```

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
);
```

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) :: bounds;
constraint min(quanAcaba) >= min(durada) :: bounds;
```

1.1.4 Resolució

Volem minimitzar la durada del festival, així que necessitem minimitzar el element màxim de la variable `quanAcaba`, que representa la hora en que acaba la última peça que es toca, per tant quan acaba el festival en sí.

```
solve :: int_search(minutInici, anti_first_fail, indomain, complete) minimize
    max(quanAcaba);
```

1.2 Proves Fetes

1.2.1 Solvers

S'han provat els següents solvers, amb els següents resultats[1]:

- Gecode: Es penja amb totes les instàncies excepte la 0.
- Chuffed: Generalment Resol totes les instàncies fins a la 6 (inclosa).

[1] En temps raonable

- OSICBC: Generalment Resol totes les instàncies fins a la 5 (inclosa). Pero molt més lenta

La resta de solvers o no funcionaven amb el nostre model (Gecode Gist, FindMus) o requerien de components externs (Gurobi i CPLEX)

Per tant les proves es faran usant Chuffed, Posat que el OSCIBC és molt més lent, i el Gecode es penja amb molta facilitat.

1.2.2 Anotacions de cerca

Al fer la cerca, s'han fet les següents proves[2]:

Notar que quan es diu Penjat, es vol dir que el programa després de 20 minuts, encara no ha acabat.

1.2.2.1 Cap anotació

Instàncies	Chuffed
1	0.28s
2	0.28s
3	2.3s
4	1s
5	0.22s
6	Penjat
7	Penjat
8	Penjat
9	45.61s
10	Penjat

1.2.2.2 varchoiceannotation

int_search(minutInici, first_fail, indomain, complete)

Instàncies	Chuffed
1	0.27s
2	0.24s
3	1.7s
4	16s
5	0.25s
6	11s
7	Penjat
8	Penjat
9	Penjat
10	Penjat

[2]Proves fetes en un Ryzen 3 1200 @3.55GHz juntament amb 8Gb de RAM

int_search(minutInici, anti_first_fail, indomain, complete)

Instàncies	Chuffed
1	0.27s
2	0.23s
3	3.32s
4	1min24s
5	0.25s
6	11.22s
7	Penjat
8	Penjat
9	5.137s
10	Penjat

int_search(minutInici, most_constrained, indomain, complete)

Instàncies	Chuffed
1	0.27s
2	0.25s
3	1.68s
4	16.34s
5	0.31s
6	11.78s
7	Penjat
8	Penjat
9	Penjat
10	Penjat

Conclusió sobre varchoiceannotation
Ens quedem amb `anti_first_fail`, posat que és la que resol més instàncies. La part negativa és que el 4 és considerablement més lent.

1.2.2.3 assignmentannotation

int_search(minutInici, anti_first_fail, indomain, complete)
Aquest cas ja l'hem vist amb anterioritat

Instàncies	Chuffed
1	0.27s
2	0.23s
3	3.32s
4	1min24s
5	0.25s
6	11.22s
7	Penjat
8	Penjat
9	5.137s
10	Penjat

int_search(minutInici, anti_first_fail, indomain_min, complete)

Instàncies	Chuffed
1	0.24s
2	0.25s
3	3.3s
4	1min23s
5	0.25s
6	11.33s
7	Penjat
8	Penjat
9	5.28s
10	Penjat

int_search(minutInici, anti_first_fail, indomain_split, complete)

Instàncies	Chuffed
1	0.25s
2	0.25s
3	2.96s
4	1min46s
5	0.25s
6	8.17s
7	Penjat
8	Penjat
9	1min22s
10	Penjat

int_search(minutInici, anti_first_fail, indomain_median, complete)

Instàncies	Chuffed
1	0.26s
2	0.25s
3	3.7s
4	28.04s
5	0.25s
6	6.74s
7	Penjat
8	Penjat
9	6min39s
10	Penjat

Conclusió sobre assignmentannotation

Hem decidit usar una de les 3 primeres, que donen temps molt semblants en general (Les diferencies podrien ser considerades dintre del marge d'error). Així al final ens quedem amb `indomain`.

1.2.2.4 Conclusió sobre les anotacions de resolució

Generalment el nostre model es bastant ràpid, excepte la instància 4 i la 9. El cas més ràpid, és quan no li donem anotacions, que tenim que la 4 és quasi instància, mentre que la 9 es fa amb 45 segons. La resta d'instàncies es resolen més o menys en el mateix temps. Però no és capaç de resoldre la instància 6 en un temps prou veloç.

Amb anotacions podem fer la instància 6 a costa de augmentar el temps d'execució de la instància 4 (però reduint el temps de resolució de la instància).

Anotacions finals: `int_search(minutInici, anti_first_fail, indomain, complete)`

Capítol 2

ScalAT

2.1 Exercici 2

2.1.1 Pregunta 1

1. **Quines diferències observeu entre les tres configuracions pel que fa a la mida?**

   La configuració 1 és la més petita, mentre que la tres, és la més gran.

   Resultats amb 8 reines:

```
Conf1
N vars: 64
N clauses: 744
N decisions: 35
N propagations: 331
N conflicts: 20

Conf2
N vars: 174
N clauses: 728
N decisions: 82
N propagations: 2031
N conflicts: 55

Conf3
N vars: 1260
N clauses: 3987
N decisions: 88
N propagations: 4456
N conflicts: 14
```

2. **Quina de les tres configuracions és ràpida? Per què creieu que passa?**

   La configuració més ràpida és la 2. Pel que fa la diferència amb la primera, és evidentment el passar de constraints quadràtiques a constraints logarítmiques, fent que tinguem moltes menys clàusules. La configuració 3, la restricció implicada ens afegeix moltes clàusules i moltes variables. Això fa que carregui molt en memoria, fent-la molt més lenta.

Resultats amb 120 reines:

```
Conf1:
    Solving time: 1.781999945640564 seconds
    N vars: 14400
    N clauses: 2851480
    N decisions: 14516
    N propagations: 79872
    N conflicts: 297
    [success] Total time: 11 s, completed Jan 28, 2019 7:51:45 PM

Conf2:
    Solving time: 0.8429999947547913 seconds
    N vars: 18918
    N clauses: 392264
    N decisions: 21592
    N propagations: 334411
    N conflicts: 773
    [success] Total time: 6 s, completed Jan 28, 2019 7:52:52 PM

Conf3:
    Solving time: 34.29499816894531 seconds
    N vars: 1540644
    N clauses: 4963395
    N decisions: 44428
    N propagations: 45871621
    N conflicts: 891
    [success] Total time: 89 s, completed Jan 28, 2019 7:48:18 PM
```

3. **Quina de les tres configuracions és més ràpida? Per què creieu que passa? Fins a quin nombre de reines sou capaços de sol·lucionar amb la millor de les configuracions.**

   Hem pogut resoldre fins a 724 en menys d'un minut (Incloent Montatge i resolució). Aquí els resultats[1]:

```
Solving time: 28.125 seconds
N vars: 563504
N clauses: 20267392
N decisions: 592731
N propagations: 4342724
N conflicts: 774
[success] Total time: 59 s, completed Jan 28, 2019 7:41:58 PM
```

[1]Totes les proves d'aquest apartat s'han fet amb un Ryzen 3 1200 @3.55GHz amb 8GB de ram, dedicant-ne 4GB al `sbt` per tal de no quedar-nos sense memòria. La unitat de emmagatzematge (i també on hi ha la unitat de swap) és un SSD Samsung 970 Evo