Universitat De Girona Montilivi
Facultat D'Enginyeria

Treball Minizinc i SAT

23/9/2024 - 21/11/2024

Nom: Oriol Tomàs Lara
Codi: u1980974

Nom: Pau Ferrer Font
Codi: u1978930

Índex

1. MINIZINC

1.1. Explicació dels models

El que hem fet en el nostre **viewpoint** és estructurar la informació dels partits utilitzant una matriu on representa els enfrontaments entre equips. Les files i columnes de la matriu representen cada un dels equips, i cada una de les caselles conté informació sobre el partit entre els dos equips corresponents. Concretament, a cada casella s'indica el dia i l'estadi on es juga el partit.

	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7	Team 8
Team 1	-----	d1 e4	d7 e5	d5 e6	d2 e2	d6 e3	d3 e1	d4 e7
Team 2	d1 e4	-----	d2 e1	d3 e2	d7 e3	d4 e5	d5 e7	d6 e6
Team 3	d7 e5	d2 e1	-----	d1 e7	d4 e6	d5 e4	d6 e2	d3 e3
Team 4	d5 e6	d3 e2	d1 e7	-----	d6 e4	d7 e1	d4 e3	d2 e5
Team 5	d2 e2	d7 e3	d4 e6	d6 e4	-----	d3 e7	d1 e5	d5 e1
Team 6	d6 e3	d4 e5	d5 e4	d7 e1	d3 e7	-----	d2 e6	d1 e2
Team 7	d3 e1	d5 e7	d6 e2	d4 e3	d1 e5	d2 e6	-----	d7 e4
Team 8	d4 e7	d6 e6	d3 e3	d2 e5	d5 e1	d1 e2	d7 e4	-----

En aquest model hi tenim les següents variables:

- **TeamVsTeam:** aquesta variable és on s'emmagatzema el viewpoint descrit anteriorment.
- **SeguidorsFora:** Variable per saber el nombre de seguidors que es queden fora. Aquesta serà utilitzada per minimitzar el nombre de seguidors que es queden fora els estadis.
- **MinKilometres:** Variable representada per un array on cada posició correspon a un equip de la competició. El valor ens indica els quilòmetres totals que haurà de recórrer aquell equip al llarg de la competició. D'aquesta manera calcularem dues coses:
  1. Sumar tots els valors per poder saber els quilòmetres totals que faran tots els equips en conjunt durant la competició
  2. Minimitzar la diferència entre l'equip que fa més i menys quilòmetres.

Els dominis de les variables han estat els següents:

- **TeamVsTeam:** rang d'1..nEquips. Pels valors de cada cel·la tindríem el següent:
  1. 1..nDies
  2. 1..nEstadis
- **SeguidorsFora:**
  - **Mínim:** màxim entre 0 | la diferencia entre la suma de tots els seguidors i la suma de les capacitats de tots els estadis multiplicada pel nombre de dies −1. En cas que la resta donés negativa voldria dir que ningú es queda fora perquè hi ha més espai que seguidors.
  - **Màxim:** El màxim seria el nombre de seguidors que hi ha pel nombre de dies. Això seria el cas que tots els seguidors es quedessin a fora
- **MinKilometres:**
  - **Mínim:** el mínim l'hem contemplat com el valor mínim de distància multiplicada per tots els estadis perquè els seguidors han de passar per tots els estadis.
  - **Màxim:** el màxim seria el contrari del mínim mirem quin és el valor màxim entre estadis i el multipliquem pel nombre d'estadis que hi ha.

Pel nostre model hem fet un total de quatre constraints i aquests ens permeten trobar totes les solucions de manera òptima per a tots els casos.

1. El primer constraint restringeix que tots els equips juguin cada dia i a cada estadi. Fem un all-diferent per cada partit que tenen els equips pel primer valor de la tupla (dia) i després pel segon valor (estadi).

2. El segon constraint el que ens restringeix són les posicions fixes donades per l'usuari. Amb la funció **card**, el que farem serà mirar si cada posició fixa el dia i estadi per dos, un o cap equip.
   - Si hem de fixar un equip, el que farem serà buscar dins la matriu del viewpoint la posició on el valor sigui igual al dia i l'estadi actuals, fent així que fixi aquests valors per aquell equip.
   - Si hem de fixar dos equips, indexarem a la matriu pels dos equips i comproves que aquell valor sigui igual al dia estadi actual, fent així que quedi fixat.
   - En cas que no hi hagi res simplement passem posant true.

3. Aquest constraint ens serveix per calcular el nombre de persones que no podran assistir a un partit a causa de la capacitat limitada de l'estadi:

   - Per a cada partit, sumem el nombre de seguidors dels dos equips que hi participen, cosa que ens dona la demanda d'aficionats per aquell partit.
   - Seguidament, faríem la resta de l'estadi on es jugarà el partit entre la demanda prèviament calculada. Si el resultat és positiu, vol dir que hi ha més aficionats que capacitat disponible, i, per tant, obtindríem el nombre de persones que es queden fora. Si el resultat és negatiu o zero, significa que l'estadi pot acomodar a tots els aficionats, i en aquest cas assignem un valor de 0 (cap persona es queda fora).
   - Finalment, sumem els resultats de tots els partits per obtenir el total de persones que no poden assistir a causa de les limitacions d'espai.

4. Aquest constraint calcula el total de quilòmetres recorreguts per cada equip al llarg de la competició, sumant les distàncies entre els estadis on juguen en dies consecutius. El procés per trobar aquestes distàncies és el següent:

   - Es consideren els dies de dos en dos (el dia actual i el dia anterior) per a cada equip. Per aquests dies, es comprova si existeix un partit que coincideixi amb el dia que estem analitzant. Això permet identificar l'estadi on es juga el partit.
   - Un cop identificats els estadis dels partits en els dos dies consecutius, s'utilitza la taula de distàncies per calcular la distància entre aquests dos estadis.
   - Les distàncies trobades per a l'equip actual es van sumant progressivament i es guarden a la variable minKilometres a la posició corresponent a aquest equip.

   Aquest càlcul assegura que cada equip tingui associada la suma total dels quilòmetres que haurà de recórrer durant tota la competició.

1.2. Restriccions

Les restriccions **globals**, que hem utilitzat ha estat només el **all_diferent** per restringir que els equips juguin en dies i estadis diferents sense repetir.

Hem utilitzat reificacions per verificar si tenim a la taula d'equip dia donada per l'usuari si tenim algun equip fixat per un dia o usuari. De la mateixa manera, hem utilitzat reificacions per verificar si un equip juga en un determinat estadi durant dos dies consecutius. Per últim, mirem si la diferència a l'hora de calcular el nombre de seguidors que es queden a fora és positiva.

També, en el nostre cas, i si no establim les variables ja descrites, **no hem afegit** variables auxiliars. Això ha estat una decisió per intentar millorar l'eficiència del programa, ja que no tenir variables auxiliars suposava no afegir constraints extres.

1.3. Upper Bound o Lower Bound Funció Objectiu

En principi a la funció objectiu no hem afegit ni upper bounds ni lower bounds.

1.4. Restriccions implicades i Simetries

Pel cas de les restriccions implicades, amb el nostre viewpoint, el que ens hem trobat ha estat:

- Al haver establert que tots els partits es juguessin en dies i estadis diferents, implícitament estem fent que els equips no juguin mai més d'un cop al dia o al mateix estadi.
- També, ja que en estem emplenant la taula en format equip contra equip, forcem que tots els equips juguin tots contra tots.
- Al tenir una matriu equip vs equip, un equip ja no podrà jugar contra ell mateix, ja que aquest valor és la diagonal de la matriu i no es tractarà.

En el nostre model, la simetria més important és que no hi ha diferència entre considerar que l'equip 1 juga contra l'equip 2 o que l'equip 2 juga contra l'equip 1. Aquesta simetria es trenca de manera natural gràcies a la forma en què estructurem la informació a la matriu equip contra equip.

Com que aquesta matriu és simètrica, només cal que ens fixem en la part triangular inferior (els elements per sota de la diagonal principal). Això ens permet ignorar la informació redundat de la triangular superior, ja que els valors d'una són els mateixos que els de l'altra.

En fer això, aconseguim trencar la simetria més significativa del model, reduint considerablement la complexitat i el temps d'execució dels càlculs, ja que treballem amb menys dades i eliminem duplicacions innecessàries.

Pel cas de la primera restricció implícita, inicialment no ens vam adonar que ja estava garantida de manera natural en el model. Això ens va portar a definir un altre constraint que assegurava que cada equip no jugués més d'un partit per dia i no repetís estadi en un mateix dia. Quan ens vam adonar que aquest constraint era innecessari i el vam haver eliminar, el temps d'execució es va reduir considerablement. Aquesta simplificació va representar una millora molt significativa en l'eficiència del model.

Pel que fa a les simetries, com que les vam tractar des del principi del desenvolupament, no tenim una mesura exacta del guany de rendiment que vam obtenir en gestionar-les. Tot i això, és molt probable que aquesta optimització hagi tingut un impacte extremadament important, ja que les simetries solen introduir redundàncies que incrementen considerablement la complexitat computacional si no es gestionen adequadament.

1.5. Cerques i Resultats

Per saber quin solver utilitzar el que hem fet ha estat una taula amb les primeres execucions per veure quan tarda cada un i aixins poder dicidir quin solver utilitzar, aquests son els segons que tarda en certificar hem posat com a time out 5 min per fer aquesta taula.

	OR Tools	**Gcode**	**HiGHS**	**Chuffed**	**COIN-BC**
T0	17.501s	2.322s	+2m	32.404s	+2m
T1	22.150s	+2m	+2m	1m27s	+2m
T1Fix	1.234s	1.624s	2.73s	1.435s	43.778s
T2Fix	6m23s	+7m	+7m	+7m	+7m

El solver que hem utilitzat és **OR-Tools**, ja que com es veu en la taula es el que millor resultats ha donat. També gràcies a la seva capacitat per treballar amb múltiples fils (threads), hem pogut aprofitar al màxim la potència computacional dels nostres ordinadors i a més, dins d'aquest entorn, hem optat per utilitzar l'*optimization level O3 two-pass compilation with Gcode*, que ens ha permès millorar considerablement el temps d'execució, ja que redueix la mida de l'arbre de cerca.

La configuració per provar el programa ha estat:

- Solver: OrTools
- Optimitzation level: O3
- Number of threads: 16
- Random seed 123123

1.5.1. Millora

Hem estat mirant de fer que els quilometres a recorre per els equips estigues equilibrada i com podem veure en aquest exemple dona fins i tot a vegades millor resultats:

	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7	Team 8	Team 9	Team 10	Team 11	Team 12	Team 13	Team 14
Team 1	-----	d1 e1	d2 e2	d3 e3	d4 e4	d5 e5	d6 e6	d13 e13	d12 e12	d11 e11	d10 e10	d9 e9	d7 e7	d8 e8
Team 2	d1 e1	-----	d10 e4	d9 e5	d11 e2	d6 e8	d13 e12	d8 e6	d7 e7	d5 e10	d4 e11	d12 e3	d3 e9	d2 e13
Team 3	d2 e2	d10 e4	-----	d1 e13	d12 e12	d13 e1	d9 e5	d5 e9	d6 e8	d7 e7	d11 e3	d4 e10	d8 e6	d3 e11
Team 4	d3 e3	d9 e5	d1 e13	-----	d5 e9	d12 e2	d7 e11	d2 e12	d13 e1	d10 e4	d8 e7	d11 e6	d6 e8	d4 e10
Team 5	d4 e4	d11 e2	d12 e12	d5 e9	-----	d3 e11	d8 e7	d10 e3	d2 e10	d9 e5	d6 e8	d13 e1	d1 e13	d7 e6
Team 6	d5 e5	d6 e8	d13 e1	d12 e2	d3 e11	-----	d4 e10	d9 e7	d1 e13	d8 e6	d7 e9	d2 e12	d11 e3	d10 e4
Team 7	d6 e6	d13 e12	d9 e5	d7 e11	d8 e7	d4 e10	-----	d1 e1	d11 e3	d12 e2	d3 e13	d10 e8	d2 e4	d5 e9
Team 8	d13 e13	d8 e6	d5 e9	d2 e12	d10 e3	d9 e7	d1 e1	-----	d3 e11	d6 e8	d12 e4	d7 e5	d4 e10	d11 e2
Team 9	d12 e12	d7 e7	d6 e8	d13 e1	d2 e10	d1 e13	d11 e3	d3 e11	-----	d4 e9	d5 e6	d8 e4	d10 e2	d9 e5
Team 10	d11 e11	d5 e10	d7 e7	d10 e4	d9 e5	d8 e6	d12 e2	d6 e8	d4 e9	-----	d2 e1	d3 e13	d13 e12	d1 e3
Team 11	d10 e10	d4 e11	d11 e3	d8 e7	d6 e8	d7 e9	d3 e13	d12 e4	d5 e6	d2 e1	-----	d1 e2	d9 e5	d13 e12
Team 12	d9 e9	d12 e3	d4 e10	d11 e6	d13 e1	d2 e12	d10 e8	d7 e5	d8 e4	d3 e13	d1 e2	-----	d5 e11	d6 e7
Team 13	d7 e7	d3 e9	d8 e6	d6 e8	d1 e13	d11 e3	d2 e4	d4 e10	d10 e2	d13 e12	d9 e5	d5 e11	-----	d12 e1
Team 14	d8 e8	d2 e13	d3 e11	d4 e10	d7 e6	d10 e4	d5 e9	d11 e2	d9 e5	d1 e3	d13 e12	d6 e7	d12 e1	-----

KMs TOTALS: 4824. [278, 297, 307, 384, 341, 367, 415, 337, 321, 406, 430, 368, 280, 293]
SEGUIDORS FORA: 812500

	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6	Team 7	Team 8	Team 9	Team 10	Team 11	Team 12	Team 13	Team 14
Team 1	-----	d1 e1	d2 e2	d3 e3	d4 e4	d5 e5	d6 e6	d8 e8	d10 e10	d13 e13	d11 e11	d9 e9	d12 e12	d7 e7
Team 2	d1 e1	-----	d6 e9	d2 e12	d10 e3	d3 e8	d13 e13	d7 e11	d9 e5	d8 e6	d4 e10	d5 e7	d11 e4	d12 e2
Team 3	d2 e2	d6 e9	-----	d1 e13	d8 e6	d13 e1	d5 e7	d12 e12	d4 e8	d11 e3	d9 e5	d10 e4	d7 e11	d3 e10
Team 4	d3 e3	d2 e12	d1 e13	-----	d9 e5	d12 e2	d4 e10	d5 e9	d8 e7	d7 e11	d6 e8	d13 e1	d10 e6	d11 e4
Team 5	d4 e4	d10 e3	d8 e6	d9 e5	-----	d6 e10	d3 e11	d2 e7	d11 e12	d5 e9	d12 e1	d7 e8	d13 e2	d1 e13
Team 6	d5 e5	d3 e8	d13 e1	d12 e2	d6 e10	-----	d10 e12	d11 e4	d7 e11	d4 e7	d1 e13	d8 e6	d9 e3	d2 e9
Team 7	d6 e6	d13 e13	d5 e7	d4 e10	d3 e11	d10 e12	-----	d9 e3	d12 e1	d1 e8	d7 e4	d11 e2	d2 e9	d8 e5
Team 8	d8 e8	d7 e11	d12 e12	d5 e9	d2 e7	d11 e4	d9 e3	-----	d13 e2	d6 e10	d3 e6	d1 e13	d4 e5	d10 e1
Team 9	d10 e10	d9 e5	d4 e8	d8 e7	d11 e12	d7 e11	d12 e1	d13 e2	-----	d2 e4	d5 e9	d3 e3	d1 e13	d6 e6
Team 10	d13 e13	d8 e6	d11 e3	d7 e11	d5 e9	d4 e7	d1 e8	d6 e10	d2 e4	-----	d10 e2	d12 e5	d3 e1	d9 e12
Team 11	d11 e11	d4 e10	d9 e5	d6 e8	d12 e1	d1 e13	d7 e4	d3 e6	d5 e9	d10 e2	-----	d2 e12	d8 e7	d13 e3
Team 12	d9 e9	d5 e7	d10 e4	d13 e1	d7 e8	d8 e6	d11 e2	d1 e13	d3 e3	d12 e5	d2 e12	-----	d6 e10	d4 e11
Team 13	d12 e12	d11 e4	d7 e11	d10 e6	d13 e2	d9 e3	d2 e9	d4 e5	d1 e13	d3 e1	d8 e7	d6 e10	-----	d5 e8
Team 14	d7 e7	d12 e2	d3 e10	d11 e4	d1 e13	d2 e9	d8 e5	d10 e1	d6 e6	d9 e12	d13 e3	d4 e11	d5 e8	-----

KMs TOTALS: 5335. [278, 358, 363, 362, 370, 377, 279, 351, 428, 403, 529, 462, 460, 315]
SEGUIDORS FORA: 822500
---

2. SCALAT

2.1. Model Escollit

2.1.1. Viewpoint

Per poder resoldre el problema del Minesweeper hem generat una matriu de n files i m columnes. Aquests valors venen donat per l'usuari i, per tant, generarem $n * m$ literals. Aquesta variable l'anomenarem **grid**.

També tindrem una **variable** auxiliar anomenada **emptyPositions**. Aquesta variable serà un array on guardarem aquelles posicions del tauler que continguin "-".

2.1.2. Restriccions

Les restriccions utilitzades serien les següent:

1. Per cada una de les caselles de tauler, mirarem si contenen un número, un guió "-" o una creu "X".
   - **Valor**: Afegim una restricció que a la casella ja no hi pot anar cap mina. Consqüentment, direm que les veïnes d'aquesta han de tenir el nombre de mines de la casella escollida.
   - **X**: En el cas de contenir una "X", restringirem la possibilitat de tenir una mina en aquella posició.
   - -: Afegirem el literal de la casella tractada a la llista de "emptyPositions".
2. En el cas de saber el nombre de mines, direm que a totes els literals guardats a "emptyPositions" han de tenir exactament el nombre de mines indicat.

Com que no hi ha simetries, no hem de fer cap restricció extra. D'aquesta manera, queden definides totes les normes per poder resoldre el buscamines.

2.2. Resultats

Dins el nostre programa, hem utilitzat un exactly-one quan el valor de la casella és igual a 1. D'aquesta manera hem pogut fer dues configuracions diferents:

- **Configuració 1(C1)**: Fem servir només codificacions quadràtiques, és a dir AMOQuad i EOQuad.
- **Configuració 2(C2)**: Fem servir només codificacions logarítmiques, és a dir AMOLog i EOLog.

Aquests serien els resultats obtinguts:

	nr1	**nr90**	**nr273**	**nr350**	**gran**
C1	0.0025434s	0.5384571s	0.0112351s	0.0307762s	12.4502524s
C2	0.0029448s	0.3239914s	0.012215s	0.0265301s	11.0067706s

Anotació: El fitxer "gran" conté un buscamines 200 x 200. Hem generat aquesta instància, ja que les proves donades eren ràpides de resoldre.

En general no es poden extreure bones conclusions amb les proves fetes, ja que les instàncies no són prou grans. Tot i això si observem el nombre de clàusules generades pel fitxer "gran", podem comprovar que es generen menys clàusules per les funcions logarítmiques (757.357) que per les codificacions quadràtiques (766.294). Així doncs, podem afirmar que per instàncies més grans i complicades, les codificacions logarítmiques seran més ràpides que les quadràtiques.

La instància més gran que vam intentar resoldre es diu "mesGran" i és un 400 x 400 on tenim 100.000 mines indicades. En el cas que eliminem les mines indicades, es mostra el resultat i tarda 33 segons amb la codificació logarítmica. El fitxer està a dins el repositori de GitHub.